### Student Information


- **Name: Vinal Dalcy Dsouza**
- **Email: dsouza.vi@northeastern.edu**
- **Student ID: 002310102**
- **AWS Notebook ARN: arn:aws:sagemaker:us-east-1:058264117601:notebook-instance/my-flight-notebook**


# 1. Problem Formulation and Understanding (4 points)

## a. Business Problem and Goal (Open Answer)

*(Clearly articulate the business problem and the intended goal.)*

**Answer:**

Flight delays can cause significant discomfort to the travelers and also damage the reputation of the air carrier agency. Our goal is to improve customer experience by predicting weather-related delays at the busiest US airports and notifying customers at the time of booking. This would allow the travellers to plan their iternary, take informed decision, and avoiding any inconvenience that could be caused due to the delay.

## b. Appropriateness of Machine Learning (Open Answer)

*(Explain why ML is appropriate for this scenario.)*

**Answer:**

Yes, ML would be an appropriate solution here due to various factors. Firstly, since we have a large volume of data that already exists, it helps ML model to learn from it and predict the output better. Secondly, flight delays, especially those caused by weather, are influenced by multiple factors such as departure time, origin, destination, distance, seasonality, and airline. Traditional statistical rule based models might not be able to identify the hidden patterns whereas ML is good at predicting patterns if trained properly. Also, as we get more data, ML model can be retrained and fine-tuned to improve predictions.

## c. Success Metrics (Open Answer)

*(Define suitable success metrics.)*

**Answer:**

- ROC-AUC (Area under the ROC curve): Measures the model's ability to distinguish between delayed and non-delayed flights. Larger the area under the curve, better is the model.
- Accuracy : Overall proportion of correct predictions.
- Precision : Ensures that when a delay is predicted, it's likely to be accurate.
- Sensitivity : Ensures that most actual delays are identified.
- Specificity : Correctly identifies non-delayed flights.

- False Positive Rate : Percentage of non-delayed flights incorrectly predicted as delayed.
- False Negative Rate : Percentage of missed delays.

### d. Type of ML Problem (Open Answer)

*(Identify the ML problem type accurately.)*

**Answer:**

Since we need to identify if there is delay or no delay, this is a Binary Classification problem type.

1: Flight delayed due to weather

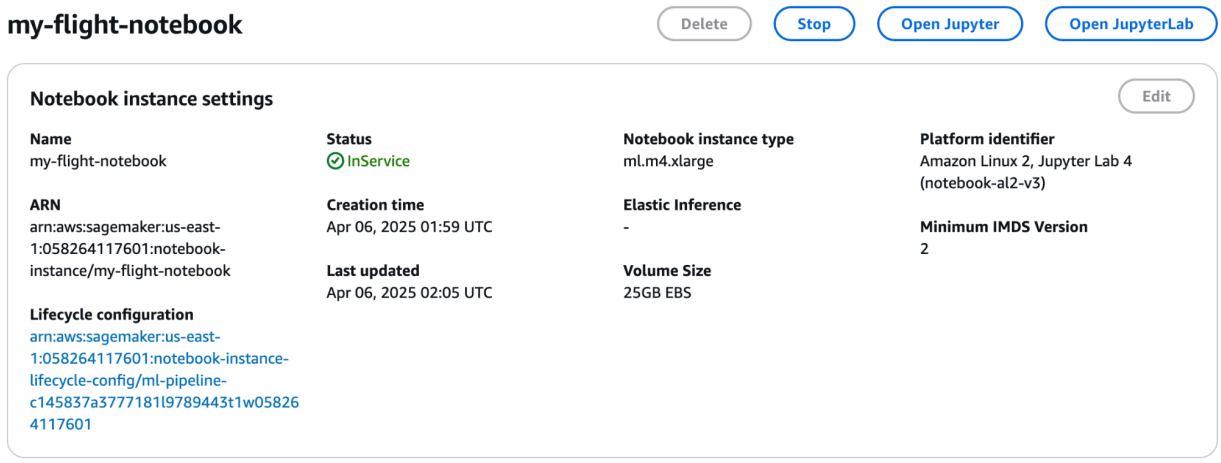
0: No delay or delay not caused by weather

# 2. Data Collection and Setup (2 points)

### a. Instance Configuration (Screenshot and Short Explanation)

*(Provide a screenshot clearly showing your AWS notebook instance configuration.)*

**Screenshot:**




**Explanation:**

A notebook instance (my-flight-notebook) of type ml.m4.xlarge is created using Amazon Sagemaker AI. The volume size of this instance is set to 25 GB. The image also shows the ARN - a unique identifier for the mentioned resource.

### b. Data Loading and Extraction (Screenshot or Code Snippet and Explanation)

*(Demonstrate the loading and extraction of CSV files.)*

**Screenshot/Code Snippet:**

[2]:
```
# download the files

zip_path = '/home/ec2-user/SageMaker/project/data/FlightDelays/'
base_path = '/home/ec2-user/SageMaker/project/data/FlightDelays/'
csv_base_path = '/home/ec2-user/SageMaker/project/data/csvFlightDelays/'

!mkdir -p {zip_path}
!mkdir -p {csv_base_path}
!aws s3 cp s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/ {zip_path} --recursive
```

```
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_2.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_2.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_12.zip to ../project/data/Fl
ightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_12.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_1.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_1.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_4.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_4.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_3.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_3.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_10.zip to ../project/data/Fl
ightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_10.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_7.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_7.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_6.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_6.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_9.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_9.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_5.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_5.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_11.zip to ../project/data/Fl
ightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_11.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_1.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_1.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_12.zip to ../project/data/Fl
ightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_12.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_8.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_8.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_10.zip to ../project/data/Fl
ightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_10.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_11.zip to ../project/data/Fl
ightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_11.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_3.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_3.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_2.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_2.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_5.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_5.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_4.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_4.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_7.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_7.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_9.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_9.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_6.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_6.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_8.zip to ../project/data/Fli
ghtDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_8.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_11.zip to ../project/data/Fl
ightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_11.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_1.zip to ../project/data/Fli
```

[3]:
```
zip_files = [str(file) for file in list(Path(base_path).iterdir()) if '.zip' in str(file)]
len(zip_files)
```

[3]:
```
60
```

Extract comma-separated values (CSV) files from the .zip files.

[4]:
```
def zip2csv(zipFile_name , file_path):
    """
    Extract csv from zip files
    zipFile_name: name of the zip file
    file_path : name of the folder to store csv
    """

    try:
        with ZipFile(zipFile_name, 'r') as z:
            print(f'Extracting {zipFile_name} ')
            z.extractall(path=file_path)
    except:
        print(f'zip2csv failed for {zipFile_name}')

for file in zip_files:
    zip2csv(file, csv_base_path)

print("Files Extracted")
```

```
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_3.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_9.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_9.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_3.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_9.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_3.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_3.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_9.zip
```

```
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_11.zip
Files Extracted
```

[5]:
```
csv_files = [str(file) for file in list(Path(csv_base_path).iterdir()) if '.csv' in str(file)]
len(csv_files)
```

[5]:
```
60
```

### Load sample CSV file

Before you combine all the CSV files, examine the data from a single CSV file. By using pandas, read the `On_Time_Reporting_Carrier_On_Time_Performance_(1987_present)_2018_9.csv` file first. You can use the built-in `read_csv` function in Python (pandas.read_csv documentation).

[7]:
```
df_temp = pd.read_csv(f"{csv_base_path}On_Time_Reporting_Carrier_On_Time_Performance_(1987_present)_2018_9.csv")
```

**Question**: Print the row and column length in the dataset, and print the column names.

**Hint**: To view the rows and columns of a DataFrame, use the `<DataFrame>.shape` function. To view the column names, use the `<DataFrame>.columns` function.

[8]:
```
df_shape = df_temp.shape
print(f'Rows and columns in one CSV file is {df_shape}')
```

```
Rows and columns in one CSV file is (585749, 110)
```

**Question**: Print the first 10 rows of the dataset.

**Hint**: To print `x` number of rows, use the built-in `head(x)` function in pandas.

```
[9]: df_temp.head(10)
```

[9]:

| | Year | Quarter | Month | DayofMonth | DayOfWeek | FlightDate | Reporting_Airline | DOT_ID_Reporting_Airline | IATA_CODE_Reporting_Airline | Tail_Number | ... | Div4TailNum | Div5Airport | Div5AirportID |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2018 | 3 | 9 | 3 | 1 | 2018-09-03 | 9E | 20363 | 9E | N908XJ | ... | NaN | NaN | NaN |
| 1 | 2018 | 3 | 9 | 9 | 7 | 2018-09-09 | 9E | 20363 | 9E | N315PQ | ... | NaN | NaN | NaN |
| 2 | 2018 | 3 | 9 | 10 | 1 | 2018-09-10 | 9E | 20363 | 9E | N582CA | ... | NaN | NaN | NaN |
| 3 | 2018 | 3 | 9 | 13 | 4 | 2018-09-13 | 9E | 20363 | 9E | N292PQ | ... | NaN | NaN | NaN |
| 4 | 2018 | 3 | 9 | 14 | 5 | 2018-09-14 | 9E | 20363 | 9E | N600LR | ... | NaN | NaN | NaN |
| 5 | 2018 | 3 | 9 | 16 | 7 | 2018-09-16 | 9E | 20363 | 9E | N316PQ | ... | NaN | NaN | NaN |
| 6 | 2018 | 3 | 9 | 17 | 1 | 2018-09-17 | 9E | 20363 | 9E | N916XJ | ... | NaN | NaN | NaN |
| 7 | 2018 | 3 | 9 | 20 | 4 | 2018-09-20 | 9E | 20363 | 9E | N371CA | ... | NaN | NaN | NaN |
| 8 | 2018 | 3 | 9 | 21 | 5 | 2018-09-21 | 9E | 20363 | 9E | N601LR | ... | NaN | NaN | NaN |
| 9 | 2018 | 3 | 9 | 23 | 7 | 2018-09-23 | 9E | 20363 | 9E | N906XJ | ... | NaN | NaN | NaN |

10 rows × 110 columns

**Explanation:**

To get started, I created dedicated folders on my local SageMaker environment to store the raw zipped files and the extracted CSV files. Using the AWS CLI, I copied the dataset from an S3 bucket into the FlightDelays directory. Once the download was completed, I listed all .zip files present in the directory.

I then wrote a function called zip2csv() which takes the name of a ZIP file and the destination folder as inputs. This function extracts all CSV files from each ZIP archive and saves them into the specified location. I ran this function in a loop for each ZIP file collected earlier.

After the extraction process was completed, I printed a confirmation message and cross-checked the output to ensure that all files had been properly unpacked and were ready for further processing. I also loaded one of the csv files and stored the data in a dataframe. First 5 records of the dataframe is displayed in the above screenshot.

# 3. Exploratory Data Analysis (EDA) and Visualization (5 points)

## a. Dataset Structure and Data Types (Open Answer – 1 point)

*(Describe the dataset structure, columns, and data types.)*

**Answer:**

The flight delay dataset used for this project is quite large and detailed. It contains 5,85,749 rows and 110 columns, which makes it suitable for building and training predictive models. The data is from the year 2018, but it specifically covers 9 months of that year — from 1st September to 30th September 2018.

This dataset includes flight records from 17 different airlines, giving a good variety of carriers and helping capture airline-specific patterns in delays. It also covers a wide network of airports, with 329 unique origin airports and 329 unique destination airports, indicating a broad representation of the US domestic aviation network.

The airport codes, like 'DFW' (Dallas/Fort Worth), 'LGA' (LaGuardia), and 'MSN' (Madison), represent different airports across the country. Because of this diversity in airports and airlines, the dataset enables us to conduct detailed analysis and model building for flight delay prediction.

We can analyse how different factors such as the origin and destination airports, airline, and date of travel impact the chances of delay. This variety in data provides a strong foundation for developing machine learning models that can predict whether a flight is likely to be delayed or not.

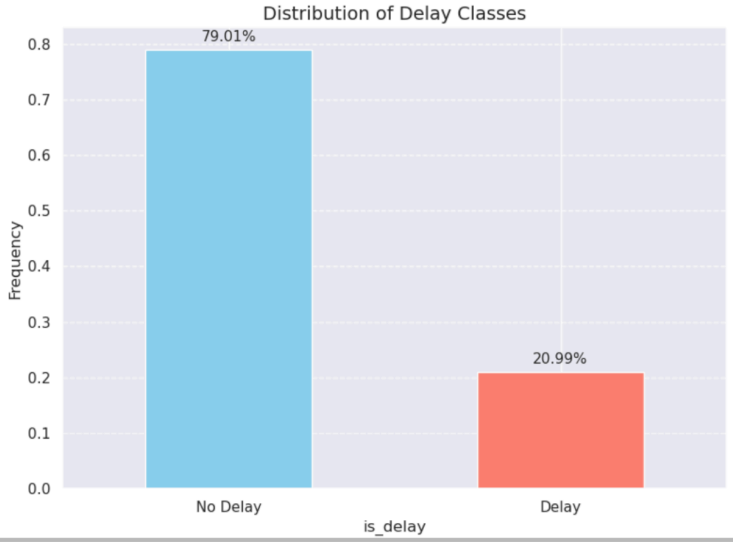
```
[24]: print("The #rows and #columns are ", df_temp.shape[0] , " and ", df_temp.shape[1])
      print("The years in this dataset are: ", df_temp.Year.unique())
      print("The months covered in this dataset are: ", df_temp.Month.unique())
      print("The date range for data is :" , min(df_temp.FlightDate.unique()), " to ", max(df_temp.FlightDate.unique()))
      print("The airlines covered in this dataset are: ", list(df_temp.Reporting_Airline.unique()))
      print("The Origin airports covered are: ", list(df_temp.Origin.unique()))
      print("The Destination airports covered are: ", list(df_temp.Dest.unique()))

      The #rows and #columns are  585749  and  110
      The years in this dataset are:  [2018]
      The months covered in this dataset are:  [9]
      The date range for data is : 2018-09-01  to  2018-09-30
      The airlines covered in this dataset are:  ['9E', 'B6', 'WN', 'YV', 'YX', 'EV', 'AA', 'AS', 'DL', 'HA', 'UA', 'F9', 'G4', 'MQ', 'NK', 'OH', 'OO']
      The Origin airports covered are:  ['DFW', 'LGA', 'MSN', 'MSP', 'ATL', 'BDL', 'VLD', 'JFK', 'RDU', 'CHS', 'DTW', 'GRB', 'PVD', 'SHV', 'FNT', 'PIT', 'RIC', 'RST', 'RSW', 'CV
      G', 'LIT', 'ORD', 'JAX', 'TRI', 'BOS', 'CWA', 'DCA', 'CHO', 'AVP', 'IND', 'GRR', 'BTR', 'MEM', 'TUL', 'CLE', 'STL', 'BTV', 'OMA', 'MGM', 'TVC', 'SAV', 'GSP', 'EWR', 'OAJ',
      'BNA', 'MCI', 'TLH', 'ROC', 'LEX', 'PWM', 'BUF', 'AGS', 'CLT', 'GSO', 'BWI', 'SAT', 'PHL', 'TYS', 'ACK', 'DSM', 'GNV', 'AVL', 'BGR', 'MHT', 'ILM', 'MOT', 'IAH', 'SBN', 'SY
      R', 'ORF', 'MKE', 'XNA', 'MSY', 'PBI', 'ABE', 'HPN', 'EVV', 'ALB', 'LNK', 'AUS', 'PHF', 'CHA', 'GTR', 'BMI', 'BQK', 'CID', 'CAK', 'ATW', 'ABY', 'CAE', 'SRQ', 'MLI', 'BHM',
      'IAD', 'CSG', 'CMH', 'MCO', 'MBS', 'FLL', 'SDF', 'TPA', 'MVY', 'LAS', 'LGB', 'SFO', 'SAN', 'LAX', 'RNO', 'PDX', 'ANC', 'ABQ', 'SLC', 'DEN', 'PHX', 'OAK', 'SMF', 'SJU', 'SE
      A', 'HOU', 'STX', 'BUR', 'SWF', 'SJC', 'DAB', 'BQN', 'PSE', 'ORH', 'HYA', 'STT', 'ONT', 'HRL', 'ICT', 'ISP', 'LBB', 'MAF', 'MDW', 'OKC', 'PNS', 'SNA', 'TUS', 'AMA', 'BOI',
      'CRP', 'DAL', 'ECP', 'ELP', 'GEG', 'LFT', 'MFE', 'MDT', 'JAN', 'COS', 'MOB', 'VPS', 'MTJ', 'DRO', 'GPT', 'BFL', 'MRY', 'SBA', 'PSP', 'FSD', 'BRO', 'RAP', 'COU', 'STS', 'PI
      A', 'FAT', 'SBP', 'FSM', 'HSV', 'BIS', 'DAY', 'BZN', 'MIA', 'EYW', 'MYR', 'HHH', 'GJT', 'FAR', 'SGF', 'HOB', 'CLL', 'LRD', 'AEX', 'ERI', 'MLU', 'LCH', 'ROA', 'LAW', 'MHK',
      'GRK', 'SAF', 'GRI', 'JLN', 'ROW', 'FWA', 'CRW', 'LAN', 'OGG', 'HNL', 'KOA', 'EGE', 'LIH', 'MLB', 'JAC', 'FAI', 'RDM', 'ADQ', 'BET', 'BRW', 'SCC', 'KTN', 'YAK', 'CDV', 'JN
      U', 'SIT', 'PSG', 'WRG', 'OME', 'OTZ', 'ADK', 'FCA', 'FAY', 'PSC', 'BIL', 'MSO', 'ITO', 'PPG', 'MFR', 'EUG', 'GUM', 'SPN', 'DLH', 'TTN', 'BKG', 'SFB', 'PIE', 'PGD', 'AZA',
      'SMX', 'RFD', 'SCK', 'OWB', 'HTS', 'BLV', 'IAG', 'USA', 'GFK', 'BLI', 'ELM', 'PBG', 'LCK', 'GTF', 'OGD', 'IDA', 'PVU', 'TOL', 'PSM', 'CKB', 'HGR', 'SPI', 'STC', 'ACT', 'TY
      R', 'ABI', 'AZO', 'CMI', 'BPT', 'GCK', 'MQT', 'ALO', 'TXK', 'SPS', 'SWO', 'DBQ', 'SUX', 'SJT', 'GGG', 'LSE', 'LBE', 'ACY', 'LYH', 'PGV', 'HVN', 'EWN', 'DHN', 'PIH', 'IMT',
      'WYS', 'CPR', 'SCE', 'HLN', 'SUN', 'ISN', 'CMX', 'EAU', 'LWB', 'SHD', 'LBF', 'HYS', 'SLN', 'EAR', 'VEL', 'CNY', 'GCC', 'RKS', 'PUB', 'LBL', 'MKG', 'PAH', 'CGI', 'UIN', 'BF
      F', 'DVL', 'JMS', 'LAR', 'SGU', 'PRC', 'ASE', 'RDD', 'ACV', 'OTH', 'COD', 'LWS', 'ABR', 'APN', 'ESC', 'PLN', 'BJI', 'BRD', 'BTM', 'CDC', 'CIU', 'EKO', 'TWF', 'HIB', 'BGM',
      'RHI', 'ITH', 'INL', 'FLG', 'YUM', 'MEI', 'PIB', 'HDN']
      The Destination airports covered are:  ['CVG', 'PWM', 'RDU', 'MSP', 'MSN', 'SHV', 'CLT', 'PIT', 'RIC', 'IAH', 'ATL', 'JFK', 'DCA', 'DTW', 'LGA', 'TYS', 'PVD', 'FNT', 'LIT',
      'BUF', 'ORD', 'TRI', 'IND', 'BGR', 'AVP', 'BWI', 'LEX', 'BDL', 'GRR', 'CWA', 'TUL', 'MEM', 'AGS', 'EWR', 'MGM', 'PHL', 'SYR', 'OMA', 'STL', 'TVC', 'ORF', 'CLE', 'ABY', 'BO
      S', 'OAJ', 'TLH', 'BTR', 'SAT', 'JAX', 'BNA', 'CHO', 'VLD', 'ROC', 'DFW', 'GNV', 'ACK', 'PBI', 'CHS', 'GRB', 'MOT', 'MKE', 'DSM', 'ILM', 'GSO', 'MCI', 'SBN', 'BTV', 'MVY',
      'XNA', 'RST', 'EVV', 'HPN', 'RSW', 'MDT', 'ROA', 'GSP', 'MCO', 'CSG', 'SAV', 'PHF', 'ALB', 'CHA', 'ABE', 'BMI', 'MSY', 'IAD', 'GTR', 'CID', 'CAK', 'ATW', 'AUS', 'BQK', 'ML
      I', 'CAE', 'CMH', 'AVL', 'MBS', 'FLL', 'SDF', 'TPA', 'LNK', 'SRQ', 'MHT', 'BHM', 'LAS', 'SFO', 'SAN', 'RNO', 'LGB', 'ANC', 'PDX', 'SJU', 'ABQ', 'SLC', 'DEN', 'LAX', 'PHX',
      'OAK', 'SMF', 'SEA', 'STX', 'BUR', 'DAB', 'SJC', 'SWF', 'HOU', 'BQN', 'PSE', 'ORH', 'HYA', 'STT', 'ONT', 'DAL', 'ECP', 'ELP', 'HRL', 'MAF', 'MDW', 'OKC', 'PNS', 'SNA', 'AM
      A', 'BOI', 'GEG', 'ICT', 'LBB', 'TUS', 'ISP', 'CRP', 'MFE', 'LFT', 'VPS', 'JAN', 'COS', 'MOB', 'DRO', 'GPT', 'BFL', 'COU', 'SBP', 'MTJ', 'SBA', 'PSP', 'FSD', 'FSM', 'BRO',
      'PIA', 'STS', 'FAT', 'RAP', 'MRY', 'HSV', 'BIS', 'DAY', 'BZN', 'MIA', 'EYW', 'MYR', 'HHH', 'GJT', 'FAR', 'MLU', 'LRD', 'CLL', 'LCH', 'FWA', 'GRK', 'SGF', 'HOB', 'LAW', 'MH
      K', 'SAF', 'JLN', 'ROW', 'GRI', 'AEX', 'CRW', 'LAN', 'ERI', 'HNL', 'KOA', 'OGG', 'EGE', 'LIH', 'JAC', 'MLB', 'RDM', 'BET', 'ADQ', 'BRW', 'SCC', 'FAI', 'JNU', 'CDV', 'YAK',
      'SIT', 'KTN', 'WRG', 'PSG', 'OME', 'OTZ', 'ADK', 'FCA', 'BIL', 'PSC', 'FAY', 'MSO', 'ITO', 'PPG', 'MFR', 'DLH', 'EUG', 'GUM', 'SPN', 'TTN', 'BKG', 'AZA', 'SFB', 'LCK', 'BL
      I', 'SCK', 'PIE', 'RFD', 'PVU', 'PBG', 'BLV', 'PGD', 'SPI', 'USA', 'TOL', 'IDA', 'ELM', 'HTS', 'HGR', 'SMX', 'OGD', 'GFK', 'STC', 'GTF', 'IAG', 'CKB', 'OWB', 'PSM', 'ABI',
      'TYR', 'ALO', 'SUX', 'AZO', 'ACT', 'CMI', 'BPT', 'TXK', 'SWO', 'SPS', 'DBQ', 'SJT', 'GGG', 'LSE', 'MQT', 'GCK', 'LBE', 'ACY', 'LYH', 'PGV', 'HVN', 'EWN', 'DHN', 'PIH', 'WY
      S', 'SCE', 'IMT', 'HLN', 'ASE', 'SUN', 'ISN', 'EAR', 'SGU', 'VEL', 'SHD', 'LWB', 'MKG', 'SLN', 'HYS', 'BFF', 'PUB', 'LBL', 'CMX', 'EAU', 'PAH', 'UIN', 'RKS', 'CGI', 'CNY',
      'JMS', 'DVL', 'LAR', 'GCC', 'LBF', 'PRC', 'RDD', 'ACV', 'OTH', 'COD', 'LWS', 'ABR', 'APN', 'PLN', 'BJI', 'CPR', 'BRD', 'BTM', 'CDC', 'CIU', 'ESC', 'EKO', 'ITH', 'HIB', 'BG
      M', 'TWF', 'RHI', 'INL', 'FLG', 'YUM', 'MEI', 'PIB', 'HDN']
```

## b. Descriptive Statistics and Visualizations (Screenshots – 1 point)

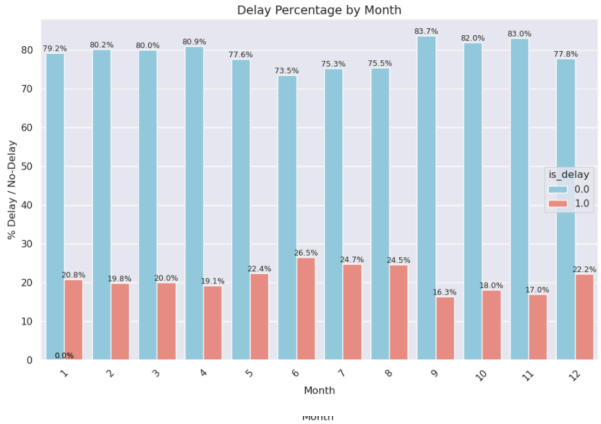
*(Generate and present descriptive statistics and visualizations. Include personalized titles or color schemes.)*

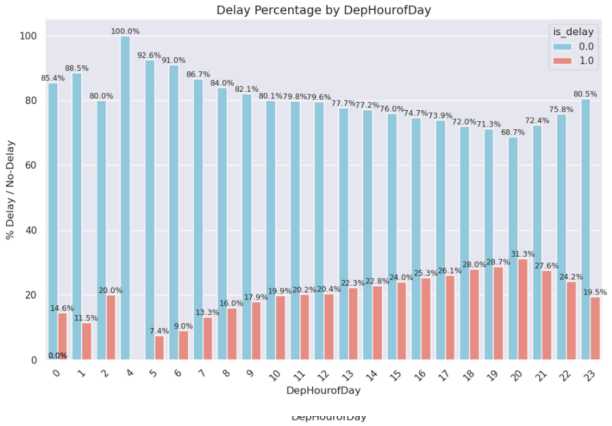
**Screenshots:**

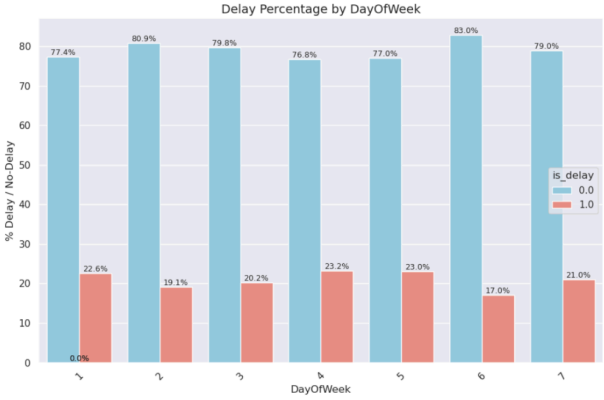
[34]:
```
class_dist = (data.groupby('is_delay').size() / len(data))
ax = class_dist.plot(kind='bar', color=['skyblue', 'salmon'], figsize=(8, 6))

plt.ylabel('Frequency', fontsize=12)
plt.title('Distribution of Delay Classes', fontsize=14)
plt.xticks(ticks=[0, 1], labels=['No Delay', 'Delay'], rotation=0, fontsize=11)

for i, v in enumerate(class_dist):
    plt.text(i, v + 0.01, f'{v:.2%}', ha='center', va='bottom', fontsize=11)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```

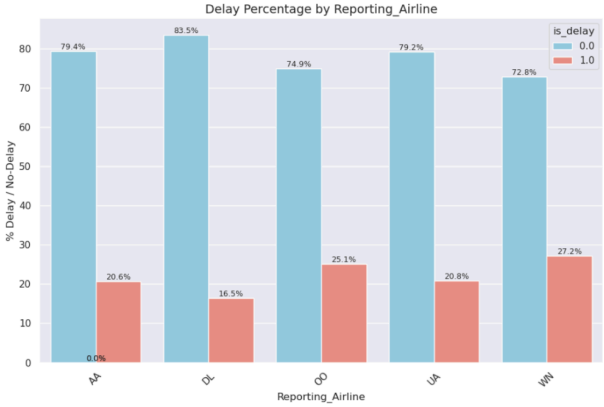


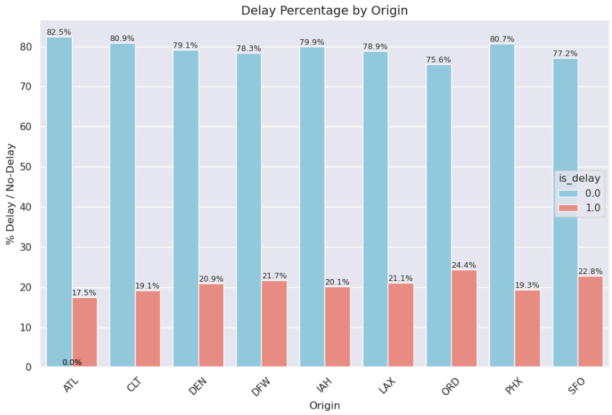

:
```
viz_columns = ['Month', 'DepHourofDay', 'DayOfWeek', 'Reporting_Airline', 'Origin', 'Dest']
fig, axes = plt.subplots(3, 2, figsize=(20, 20), squeeze=False)

for idx, column in enumerate(viz_columns):
    ax = axes[idx // 2, idx % 2]
    temp = data.groupby(column)['is_delay'].value_counts(normalize=True).rename('percentage')\
              .mul(100).reset_index().sort_values(column)

    sns.barplot(x=column, y="percentage", hue="is_delay", data=temp, ax=ax,
                palette={0: 'skyblue', 1: 'salmon'})

    ax.set_title(f'Delay Percentage by {column}', fontsize=14)
    ax.set_ylabel('% Delay / No-Delay', fontsize=12)
    ax.tick_params(axis='x', rotation=45)

    # Annotate bars
    for p in ax.patches:
        height = p.get_height()
        ax.annotate(f'{height:.1f}%', (p.get_x() + p.get_width() / 2, height),
                    ha='center', va='bottom', fontsize=9)

plt.tight_layout()
plt.show()
```

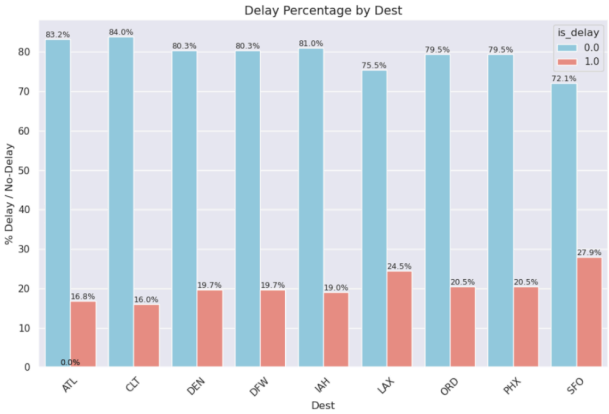

Delay Percentage by Month
% Delay / No-Delay
Month
is_delay
0.0
1.0


Delay Percentage by DepHourofDay
% Delay / No-Delay
DepHourofDay
is_delay
0.0
1.0

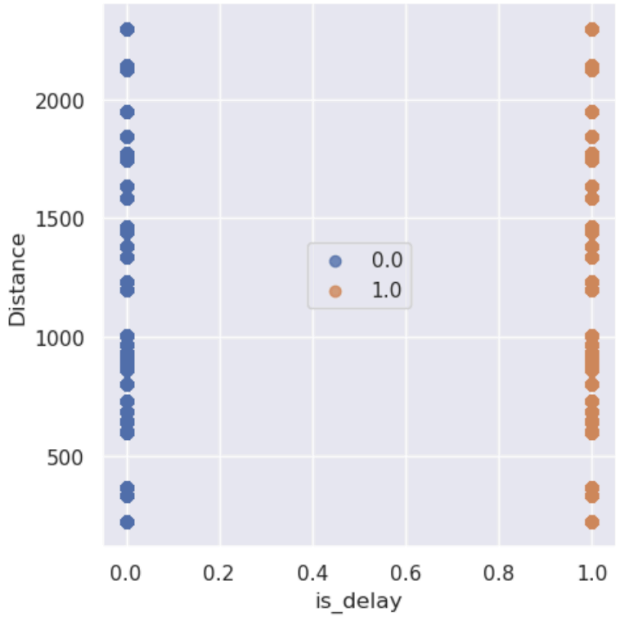

Delay Percentage by DayOfWeek
% Delay / No-Delay
DayOfWeek
is_delay
0.0
1.0


Delay Percentage by Reporting_Airline
% Delay / No-Delay
Reporting_Airline
is_delay
0.0
1.0


Delay Percentage by Origin
% Delay / No-Delay
Origin
is_delay
0.0
1.0


Delay Percentage by Dest
% Delay / No-Delay
Dest
is_delay
0.0
1.0

```
[35]: sns.lmplot( x="is_delay", y="Distance", data=data, fit_reg=False, hue='is_delay', legend=False)
      plt.legend(loc='center')
      plt.xlabel('is_delay')
      plt.ylabel('Distance')
      plt.show()
```




## c. Interpretation of EDA Results (Open Answer – 2 points for each correctly answered question)

- Which months have the most delays?
- What time of day has the most delays?
- What day of the week has the most delays?
- Which airline has the most delays?
- Which origin and destination airports have the most delays?
- Is flight distance a factor in delays?

**Answers:**

1. Month of June has most delays i.e. 26.5%.
2. Around 20th hour(8pm) the flight would be delayed 31.3% standing highest as compared to the other time of the day.
3. Wednesday has most number of delays - 23.2%
4. WN airlines has most number of delays - 27.2%
5. Origin airport: ORD, Destination airport : SFO
6. From the scatter plot, there doesn't seem to be a strong corelation. Hence distance does not seem to be a factor.

# 4. Data Preprocessing and Feature Engineering (4 points)

## a. Handling Missing Data (Open Answer)

*(Explain your approach to handling missing data.)*

**Answer:**

It is essential to remove any noise in the dataset in the data preprocessing step. Around 1.3% of arrival delay details and airtime were missing. While compared to the whole dataset, its significantly low.

```
[26]: data.isnull().sum(axis=0)
```

```
[26]: Year                    0
      Quarter                 0
      Month                   0
      DayofMonth              0
      DayOfWeek               0
      FlightDate              0
      Reporting_Airline       0
      Origin                  0
      OriginState             0
      Dest                    0
      DestState               0
      CRSDepTime              0
      Cancelled               0
      Diverted                0
      Distance                0
      DistanceGroup           0
      ArrDelay            22540
      ArrDelayMinutes     22540
      is_delay            22540
      AirTime             22540
      dtype: int64
```

The arrival delay details and airtime are missing for 22,540 out of 1,658,130 rows, which is 1.3 percent. You can either remove or impute these rows. The documentation doesn't mention any information about missing rows.

All the rows that had null for is_delay field were removed since we would only want '1' or '0' in the is_delay column. The below screenshot shows the code used to do the same.

```
[27]: ### Remove null columns
      data = data[~data.is_delay.isnull()]
      data.isnull().sum(axis = 0)
```

```
[27]: Year                0
      Quarter             0
      Month               0
      DayofMonth          0
      DayOfWeek           0
      FlightDate          0
      Reporting_Airline   0
      Origin              0
      OriginState         0
      Dest                0
      DestState           0
      CRSDepTime          0
      Cancelled           0
      Diverted            0
      Distance            0
      DistanceGroup       0
      ArrDelay            0
      ArrDelayMinutes     0
      is_delay            0
      AirTime             0
      dtype: int64
```

Get the hour of the day in 24-hour-time format from CRSDepTime.

```
[28]: data['DepHourofDay'] = (data['CRSDepTime']//100)
```

```
[30]: data['DepHourofDay'].head(5)
```

```
[30]: 0    10
      1     7
      2     8
      3     6
      4    15
      Name: DepHourofDay, dtype: int64
```

## b. Encoding Categorical Variables (Open Answer)

*(Describe your method for encoding categorical variables.)*

**Answer:**

One hot encoding method is used.

1. Initially the data is copied on to data_origin. Copy() is used to achieve this since we do not want any changes done to data to be impacted to data_origin.
2. Categorical columns like Quarter, Month, DayofMonth, DayofWeek, ReportingAirline, Origin, Dest, DepartureDay were identified.

3. get_dummies() from Pandas was used on the categorical columns and result returned into a new data frame data_dummies.
4. If the corresponding value in the column is True then it is replaced with '1' and False then '0'.
5. Original data frame data now is concatenated with the data_dummies using concat().
6. Also we do not need the existing categorical columns(one that existed before encoding) anymore so those columns are dropped.

```
[38]: data_orig = data.copy()
data = data[['is_delay', 'Quarter', 'Month', 'DayofMonth', 'DayOfWeek',
       'Reporting_Airline', 'Origin', 'Dest','Distance','DepHourofDay']]
categorical_columns = ['Quarter', 'Month', 'DayofMonth', 'DayOfWeek',
       'Reporting_Airline', 'Origin', 'Dest', 'DepHourofDay']
for c in categorical_columns:
    data[c] = data[c].astype('category')
```

To use one-hot encoding, use the `get_dummies` function in pandas for the categorical columns that you selected. Then, you can concatenate those generated features to your original dataset by using the `concat` function in pandas. For encoding categorical variables, you can also use *dummy encoding* by using a keyword `drop_first=True`. For more information about dummy encoding, see Dummy variable (statistics).

For example:

```
pd.get_dummies(df[['column1','columns2']], drop_first=True)
```

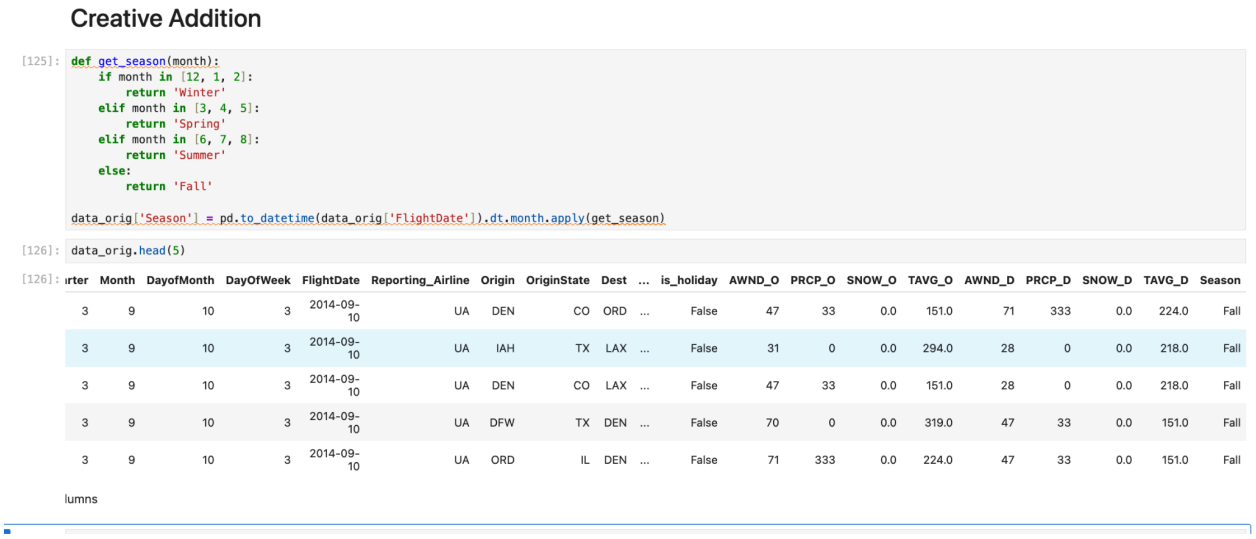
```
[39]: data_dummies = pd.get_dummies(data[['Quarter', 'Month', 'DayofMonth', 'DayOfWeek',
       'Reporting_Airline', 'Origin', 'Dest', 'DepHourofDay']], drop_first=True) # Enter your code here
data_dummies = data_dummies.replace({True: 1, False: 0})
data = pd.concat([data, data_dummies], axis = 1)
data.drop(categorical_columns,axis=1, inplace=True)
```

## c. Creation of Additional Features (Open Answer)

*(Explain additional meaningful features created, such as is_holiday and weather conditions, clearly stating your rationale.)*

**Answer:**

Weather conditions and travel demand vary by season to season which could potentially influence delays. Categorizing which season it belongs to might add value to the prediction.



Creative Addition

```
[125]: def get_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    else:
        return 'Fall'

data_orig['Season'] = pd.to_datetime(data_orig['FlightDate']).dt.month.apply(get_season)
```

```
[126]: data_orig.head(5)
```

[126]:

| ırter | Month | DayofMonth | DayOfWeek | FlightDate | Reporting_Airline | Origin | OriginState | Dest | ... | is_holiday | AWND_O | PRCP_O | SNOW_O | TAVG_O | AWND_D | PRCP_D | SNOW_D | TAVG_D | Season |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 9 | 10 | 3 | 2014-09-10 | UA | DEN | CO | ORD | ... | False | 47 | 33 | 0.0 | 151.0 | 71 | 333 | 0.0 | 224.0 | Fall |
| 3 | 9 | 10 | 3 | 2014-09-10 | UA | IAH | TX | LAX | ... | False | 31 | 0 | 0.0 | 294.0 | 28 | 0 | 0.0 | 218.0 | Fall |
| 3 | 9 | 10 | 3 | 2014-09-10 | UA | DEN | CO | LAX | ... | False | 47 | 33 | 0.0 | 151.0 | 28 | 0 | 0.0 | 218.0 | Fall |
| 3 | 9 | 10 | 3 | 2014-09-10 | UA | DFW | TX | DEN | ... | False | 70 | 0 | 0.0 | 319.0 | 47 | 33 | 0.0 | 151.0 | Fall |
| 3 | 9 | 10 | 3 | 2014-09-10 | UA | ORD | IL | DEN | ... | False | 71 | 333 | 0.0 | 224.0 | 47 | 33 | 0.0 | 151.0 | Fall |

lumns

### d. Personalized Documentation of Preprocessing Decisions (Open Answer)

*(Include a personalized comment clearly documenting your preprocessing decisions.)*

**Answer:**

The dataset initially had 110 columns, all of which were not important to the issue in hand. I only focused on the fields that I thought were important, getting rid of the rest. Additionally since is_delay was a field that played major role in training the model, noise around this field was removed. Also ensured to correctly encode the categorical columns.

# 5. Model Training and Evaluation (3 points)

### a. Baseline Model Establishment (Open Answer)

*(Justify your choice of baseline algorithm - Linear Learner.)*

**Answer:**

I chose Linear Learner as the baseline algorithm because it is simple, easy to understand, and works well for binary classification problems like mine. Since my data is structured, Linear Learner helps me get quick and interpretable results, which is useful for understanding how the features are contributing to the prediction. Also, SageMaker's version is optimised for performance and automatically selects the best model based on cross-entropy loss, so I don't have to manually tune everything.

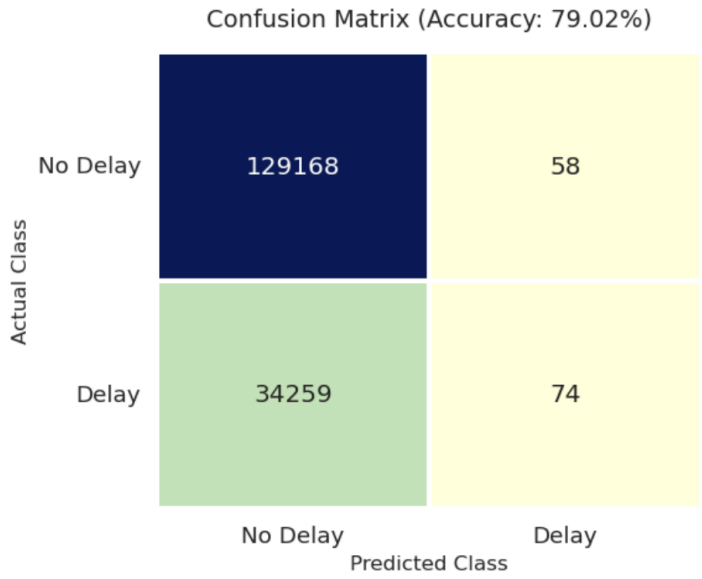
### b. Model Training and Evaluation (Screenshots with Annotations)

*(Include confusion matrix, accuracy, and ROC curves with personalized annotations.)*

**Screenshots:**

- **Confusion Matrix, Accuracy**: The confusion matrix shows that the model correctly predicted the majority of on-time flights but could barely identify Delays. Model only successfully identified delays 74 times whereas it gave incorrect results 34259 times, making accuracy 79.02% .

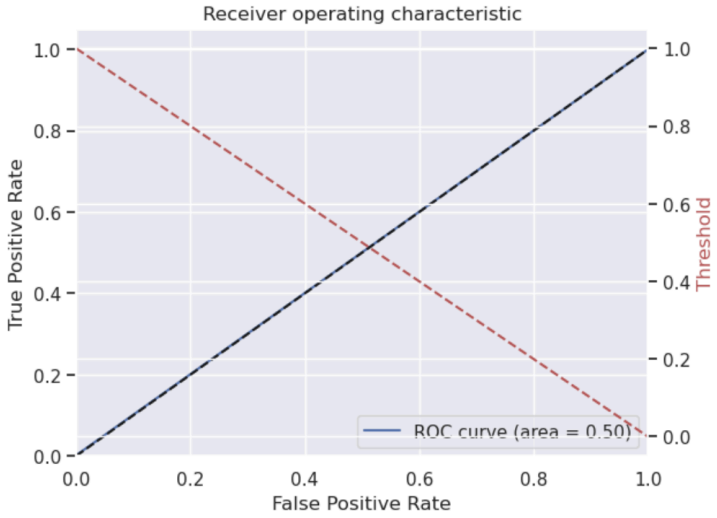
```
[72]: plot_confusion_matrix(test_labels,target_predicted)
```




- **ROC**: The area under the curve is only around 0.50. Hence indicating that this model needs to be trained further to ensure better results. More AUC the better here.

```
[62]: plot_roc(test_labels, target_predicted)
```

```
Sensitivity or TPR:  0.2155360731657589 %
Specificity or TNR:  99.95511739123705 %
Precision:  56.060606060606055 %
Negative Predictive Value:  79.03712360870603 %
False Positive Rate:  0.04488260876294244 %
False Negative Rate:  99.78446392683425 %
False Discovery Rate:  43.93939393939394 %
Accuracy:  79.01858045109105 %
Validation AUC 0.5008532673220141
Figure(640x480)
```




```
<Figure size 640x480 with 0 Axes>
```

### c. Interpretation of Model Performance (Open Answer)

*(Provide a detailed interpretation identifying strengths and weaknesses.)*

**Answer:**

Looking at the confusion matrix and ROC curve, I feel the model is doing okay in terms of overall accuracy, which is around 79%. It is able to correctly predict most of the "No Delay" cases, which is why the specificity is very high (almost 99.96%). Also, the false positive rate is very low, so the model is not raising unnecessary delay alarms, which is a good thing.

But at the same time, there are some major weaknesses also. The sensitivity is very low (around 21.5%), which means the model is missing out on a lot of actual "Delay" cases. Out of more than 34,000 actual delay cases, it could predict only 74 correctly. This is a big problem if delay prediction is important for the use case. Also, the AUC value is 0.50, which basically means the model is doing no better than random guessing. So even though the accuracy is looking good, it is mostly because the data is imbalanced, with a lot more "No Delay" cases. So overall, the model is biased towards the majority class and not learning much about the "Delay" class.

## 6. Hyperparameter Optimization and Advanced Modeling (2 points)

### a. Hyperparameter Tuning Process (Screenshots and Open Answer)

*(Explain your hyperparameter optimization process, chosen ranges, and rationale.)*

**Screenshot:**

```
sage_client = boto3.Session().client('sagemaker')
tuning_job_name = tuner.latest_tuning_job.job_name
print(f'tuning job name:{tuning_job_name}')
tuning_job_result = sage_client.describe_hyper_parameter_tuning_job(HyperParameterTuningJobName=tuning_job_name)
best_training_job = tuning_job_result['BestTrainingJob']
best_training_job_name = best_training_job['TrainingJobName']
print(f"best training job: {best_training_job_name}")

best_estimator = tuner.best_estimator()

tuner_df = sagemaker.HyperparameterTuningJobAnalytics(tuning_job_name).dataframe()
tuner_df.head()
```

```
[04/06/25 06:06:43] INFO     Found credentials from IAM Role:                                credentials.py:1132
                             BaseNotebookInstanceEc2InstanceRole

tuning job name:sagemaker-xgboost-250406-0517
best training job: sagemaker-xgboost-250406-0517-009-affc0152

2025-04-06 05:58:27 Starting - Found matching resource for reuse
2025-04-06 05:58:27 Downloading - Downloading the training image
2025-04-06 05:58:27 Training - Training image download completed. Training in progress.
2025-04-06 05:58:27 Uploading - Uploading generated training model
2025-04-06 05:58:27 Completed - Resource reused by training job: sagemaker-xgboost-250406-0517-010-6fded35c
```

| | alpha | eta | min_child_weight | num_round | subsample | TrainingJobName | TrainingJobStatus | FinalObjectiveValue | TrainingStartTime | TrainingEndTime | TrainingElapsedTimeSeconds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 30.776656 | 0.462859 | 7.743142 | 36.0 | 0.513018 | sagemaker-xgboost-250406-0517-010-6fded35c | Completed | 0.72957 | 2025-04-06 05:58:30+00:00 | 2025-04-06 06:00:59+00:00 | 149.0 |
| 1 | 26.839969 | 0.237453 | 4.482515 | 104.0 | 0.751361 | sagemaker-xgboost-250406-0517-009-affc0152 | Completed | 0.73942 | 2025-04-06 05:52:04+00:00 | 2025-04-06 05:57:29+00:00 | 325.0 |
| 2 | 325.979799 | 0.259279 | 7.860463 | 123.0 | 0.776748 | sagemaker-xgboost-250406-0517-008-e50ae693 | Completed | 0.72634 | 2025-04-06 05:44:26+00:00 | 2025-04-06 05:50:56+00:00 | 390.0 |
| 3 | 515.951723 | 0.438011 | 3.158364 | 104.0 | 0.744704 | sagemaker-xgboost-250406-0517-007-dcf160d1 | Completed | 0.72098 | 2025-04-06 05:38:58+00:00 | 2025-04-06 05:43:53+00:00 | 295.0 |
| 4 | 469.385209 | 0.313702 | 9.945242 | 38.0 | 0.892781 | sagemaker-xgboost-250406-0517-006-95014a84 | Completed | 0.71601 | 2025-04-06 05:36:03+00:00 | 2025-04-06 05:38:27+00:00 | 144.0 |

**Explanation:**

To improve the model's performance and make it more generalised, I used Amazon SageMaker's HyperparameterTuner with the XGBoost algorithm. The main aim of this process was to find the best set of hyperparameters that maximise the validation AUC score, especially because my dataset is imbalanced. I selected a few important hyperparameters and defined their search ranges thoughtfully.

- alpha (L1 regularisation term): I kept it between 0 and 1000 to control overfitting by penalising large weights.
- eta (learning rate): I set the range between 0.1 and 0.5. A smaller value might lead to better generalisation, but I didn't go too low to avoid very slow training.
- min_child_weight: I chose the range 3 to 10 to manage the model complexity and prevent overfitting on noise or small leaf nodes.
- subsample: I set it between 0.5 and 1.0 to introduce randomness during training, helping to reduce overfitting.
- num_round: I varied the number of boosting rounds from 10 to 150 to let the model train just enough without going too far and overfitting.

I kept the maximum number of tuning jobs to 10 due to time and budget limitations, and ran them sequentially (max_parallel_jobs=1). Once tuning was complete, I extracted the best training job and model using the built-in methods, and compared its performance with the baseline model.

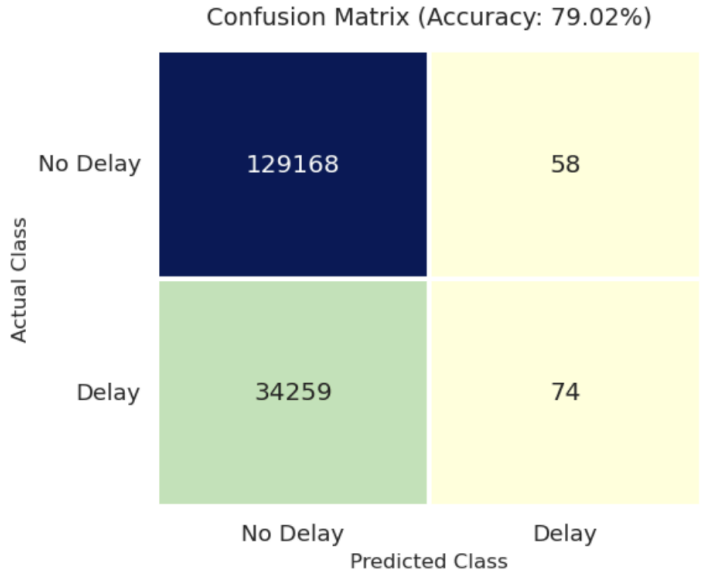
## b. Comparison of Optimized and Baseline Model (Screenshots and Open Answer)

*(Evaluate and compare your optimized model with the baseline.)*

**Screenshots:**

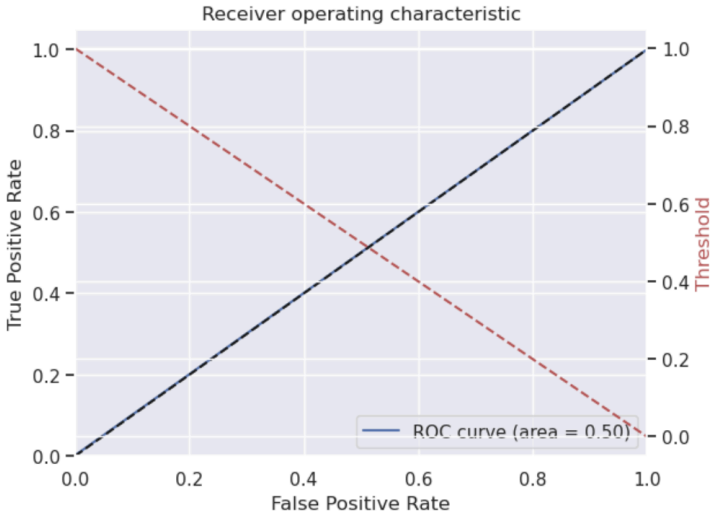
**Initial Baseline Model with Linear Learner**

[72]:
```
plot_confusion_matrix(test_labels,target_predicted)
```




[62]:
```
plot_roc(test_labels, target_predicted)
```

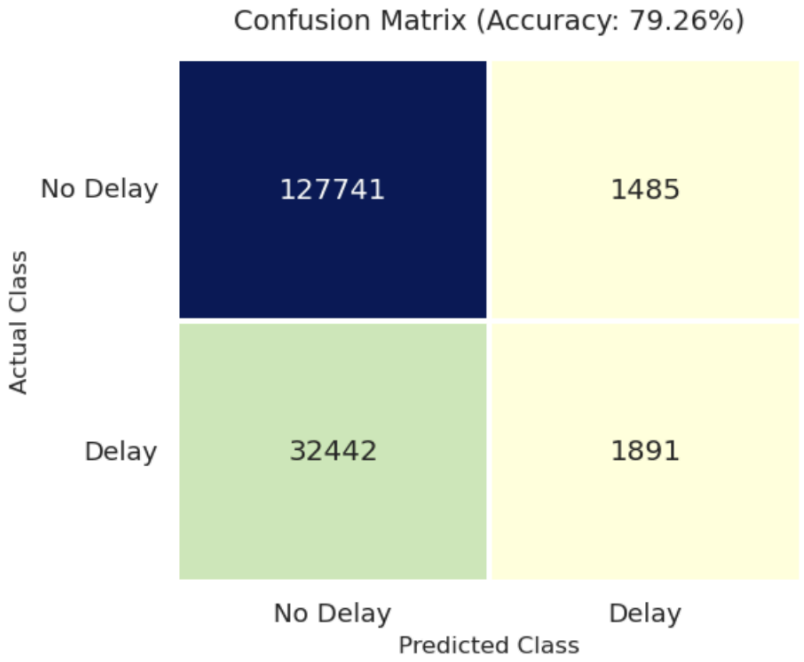
```
Sensitivity or TPR:  0.2155360731657589 %
Specificity or TNR:  99.95511739123705 %
Precision:  56.060606060606055 %
Negative Predictive Value:  79.03712360870603 %
False Positive Rate:  0.04488260876294244 %
False Negative Rate:  99.78446392683425 %
False Discovery Rate:  43.93939393939394 %
Accuracy:  79.01858045109105 %
Validation AUC 0.5008532673220141
Figure(640x480)
```




```
<Figure size 640x480 with 0 Axes>
```

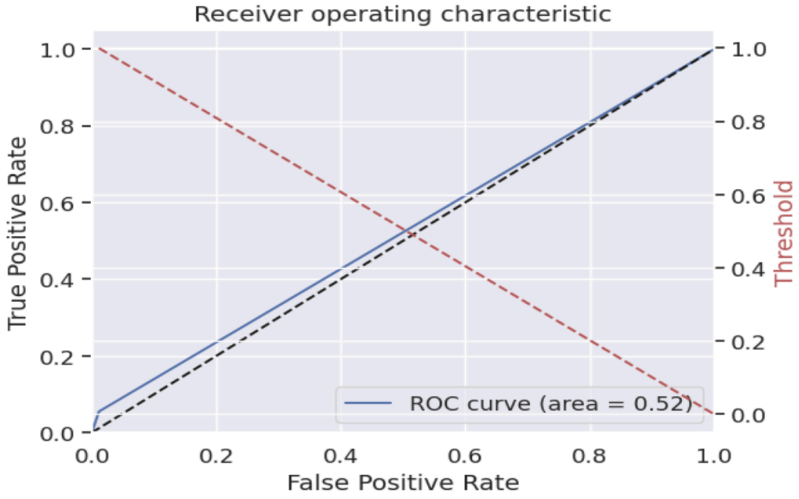
## New Baseline Model after Feature Engineering

[101]:
```
plot_confusion_matrix(test_labels, target_predicted)
```



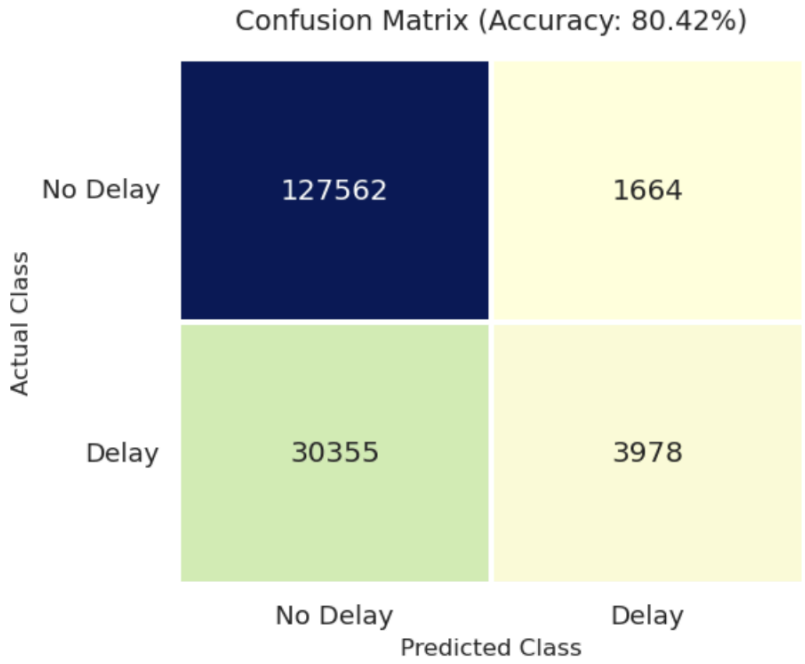

[102]:
```
plot_roc(test_labels, target_predicted)
```

```
Sensitivity or TPR:  5.507820464276352 %
Specificity or TNR:  98.85085044805226 %
Precision:  56.01303317535545 %
Negative Predictive Value:  79.74691446657886 %
False Positive Rate:  1.1491495519477506 %
False Negative Rate:  94.49217953572365 %
False Discovery Rate:  43.986966824644554 %
Accuracy:  79.25702651642527 %
Validation AUC 0.521793354561643
Figure(640x480)
```



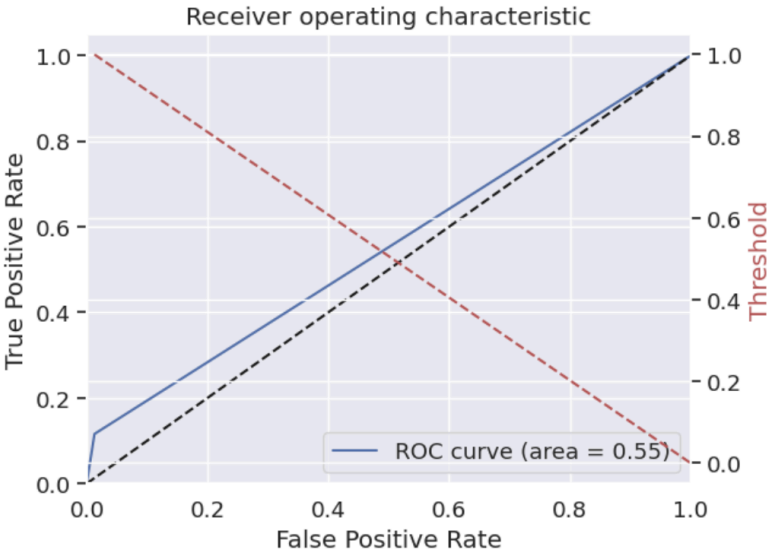
## Optimised Model:

[122]:
```
plot_confusion_matrix(test_labels, target_predicted)
```




[123]:
```
plot_roc(test_labels, target_predicted)
```

```
Sensitivity or TPR:  11.586520257478227 %
Specificity or TNR:  98.71233343135283 %
Precision:  70.50691244239631 %
Negative Predictive Value:  80.77787698601165 %
False Positive Rate:  1.2876665686471762 %
False Negative Rate:  88.41347974252177 %
False Discovery Rate:  29.493087557603687 %
Accuracy:  80.42357803606038 %
Validation AUC 0.5514942684441553
Figure(640x480)
```




```
<Figure size 640x480 with 0 Axes>
```

**Comparison:**
The optimized flight delay prediction model shows notable improvements over the baseline Linear Learner model. While accuracy increased modestly from 79.02% to 80.42%, the most significant enhancement is in delay detection capability, with sensitivity jumping from a mere 0.22% to 11.59% - resulting in 3,978 correctly identified delays compared to just 74 previously. Precision also improved substantially from 56.06% to 70.51%, meaning greater reliability when the model predicts a delay. The ROC AUC value rose from 0.50 to 0.55, indicating better discriminative ability between classes. These gains came with a reasonable tradeoff of slightly increased false positives (1.29% vs 0.04%), which is acceptable considering the operational and customer service benefits of correctly identifying more delays. Despite these improvements, the model still misses 88.41% of actual delays, suggesting scope for further refinement through advanced techniques like ensemble methods or deep learning approaches.

## Penalties

- Failure to clearly include individualization (name or student ID, personalized comments, unique visual formatting) may result in up to 2 points deducted.
- Identical or nearly identical code cells or results among multiple students may result in additional point deductions for suspected copying or plagiarism.