Problem: Predicting Airplane Delays

The goals of this notebook are:

- Process and create a dataset from downloaded .zip files
- Perform exploratory data analysis (EDA)
- Establish a baseline model
- Move from a simple model to an ensemble model
- Perform hyperparameter optimization
- Check feature importance

Introduction to business scenario

You work for a travel booking website that wants to improve the customer experience for flights that were delayed. The company wants to create a feature to let customers know if the flight will be delayed because of weather when they book a flight to or from the busiest airports for domestic travel in the US.

You are tasked with solving part of this problem by using machine learning (ML) to identify whether the flight will be delayed because of weather. You have been given access to the a dataset about the on-time performance of domestic flights that were operated by large air carriers. You can use this data to train an ML model to predict if the flight is going to be delayed for the busiest airports.

About this dataset

This dataset contains scheduled and actual departure and arrival times reported by certified US air carriers that account for at least 1 percent of domestic scheduled passenger revenues. The data was collected by the U.S. Office of Airline Information, Bureau of Transportation Statistics (BTS). The dataset contains date, time, origin, destination, airline, distance, and delay status of flights for flights between 2013 and 2018.

Features

For more information about features in the dataset, see On-time delay dataset features.

Dataset attributions

Website: https://www.transtats.bts.gov/

Dataset(s) used in this lab were compiled by the U.S. Office of Airline Information, Bureau of Transportation Statistics (BTS), Airline On-Time Performance Data, available at https://www.transtats.bts.gov/DatabaseInfo.asp?DB_ID=120&DB_URL=Mode_ID=1&Mode_Desc=Aviation&Subject_ID

Step 1: Problem formulation and data collection

Start this project by writing a few sentences that summarize the business problem and the business goal that you want to achieve in this scenario. You can write down your ideas in the following sections. Include a business metric that you would like your team to aspire toward. After you define that information, write the ML problem statement. Finally, add a comment or two about the type of ML this activity represents.

Project presentation: Include a summary of these details in your project presentation.

1. Determine if and why ML is an appropriate solution to deploy for this scenario.

In [ ]:
```
We have a lot of past flight data.
Delay patterns are complex (weather, time, airport).
ML can find patterns and predict delays better than simple rules.
It saves time and works at scale.
```

2. Formulate the business problem, success metrics, and desired ML output.

In [ ]:
```
Business Problem:
Help users know if a flight might be delayed due to weather.
Show this info when booking flights on the website.
Success Metrics:
High accuracy of the model.
Good recall (catch all real delays).
F1 score to balance correct predictions.
Fewer customer complaints about delays.
ML Output:
A simple answer:
```

```
1 = delayed because of weather
0 = not delayed due to weather
```

3. Identify the type of ML problem that you're working with.

In [ ]:
```
Supervised learning (we know the answers in training data).
Binary classification (yes/no for weather delay).
```

4. Analyze the appropriateness of the data that you're working with.

In [ ]:
```
Large dataset with many flights.
Has useful info: time, airports, delays, etc.
From a trusted source (U.S. government).
Possible issues:
May have few weather delays (class imbalance).
Some data might be missing or messy.
Weather can change over years (older data might be outdated).
Might need to add weather info from other sources.
```

Setup

Now that you have decided where you want to focus your attention, you will set up this lab so that you can start solving the problem.

Note: This notebook was created and tested on an `ml.m4.xlarge` notebook instance with 25 GB storage.

In [1]:
```
import os
from pathlib2 import Path
from zipfile import ZipFile
import time

import pandas as pd
import numpy as np
import subprocess

import matplotlib.pyplot as plt
import seaborn as sns

sns.set()
instance_type='ml.m4.xlarge'

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/
computation/expressions.py:21: UserWarning: Pandas requires version '2.8.4' or
newer of 'numexpr' (version '2.7.3' currently installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
Matplotlib is building the font cache; this may take a moment.
```

Step 2: Data preprocessing and visualization

In this data preprocessing phase, you explore and visualize your data to better understand it. First, import the necessary libraries and read the data into a pandas DataFrame. After you import the data, explore the dataset. Look for the shape of the dataset and explore your columns and the types of columns that you will work with (numerical, categorical). Consider performing basic statistics on the features to get a sense of feature means and ranges. Examine your target column closely, and determine its distribution.

Specific questions to consider

Throughout this section of the lab, consider the following questions:

1. What can you deduce from the basic statistics that you ran on the features?
2. What can you deduce from the distributions of the target classes?
3. Is there anything else you can deduce by exploring the data?

Project presentation: Include a summary of your answers to these questions (and other similar questions) in your project presentation.

Start by bringing in the dataset from a public Amazon Simple Storage Service (Amazon S3) bucket to this notebook environment.

In [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/c
```

```
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_1.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2014_1.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_3.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2014_3.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_10.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2014_10.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_2.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2014_2.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_12.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2014_12.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_4.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2014_4.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_5.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2014_5.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_6.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2014_6.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_8.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2014_8.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_7.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2014_7.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_9.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2014_9.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_1.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2015_1.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_10.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2015_10.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_2.zip to ../p
```

```
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2015_2.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_11.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2015_11.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_11.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2014_11.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_12.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2015_12.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_3.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2015_3.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_4.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2015_4.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_5.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2015_5.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_7.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2015_7.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_9.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2015_9.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_10.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2016_10.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_2.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2016_2.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_8.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2015_8.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_1.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2016_1.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_3.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2016_3.zip
```

```
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_4.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2016_4.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_11.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2016_11.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_7.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2016_7.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_8.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2016_8.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_5.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2016_5.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_9.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2016_9.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_6.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2016_6.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_11.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2017_11.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_2.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2017_2.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_6.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2015_6.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_3.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2017_3.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_4.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2017_4.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_12.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2017_12.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_6.zip to ../p
```

```
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2017_6.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_1.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2017_1.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_9.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2017_9.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_1.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2018_1.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_12.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2016_12.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_8.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2017_8.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_10.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2018_10.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_12.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2018_12.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_2.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2018_2.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_3.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2018_3.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_4.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2018_4.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_10.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2017_10.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_5.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2018_5.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_7.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2018_7.zip
```

```
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_8.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2018_8.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_7.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2017_7.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_11.zip to
../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_198
7_present_2018_11.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_6.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2018_6.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_9.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2018_9.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_5.zip to ../p
roject/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_pre
sent_2017_5.zip
```

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

Out[3]:
```
60
```

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

In [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_Reportin
g_Carrier_On_Time_Performance_1987_present_2017_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2016_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2015_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2015_9.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2014_3.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2016_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2018_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2015_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2018_9.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2017_9.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2015_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2017_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2015_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2015_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2018_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2018_3.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2017_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2017_3.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2018_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2015_3.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2014_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2016_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2014_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2017_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2014_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2018_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2014_4.zip
```

```
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2018_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2016_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2018_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2016_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2017_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2017_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2014_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2014_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2017_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2016_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2016_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2015_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2016_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2016_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2015_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2018_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2015_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2018_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2017_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2015_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2017_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2018_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2015_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2018_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2014_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2016_3.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2014_5.zip
```

```
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2016_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2014_9.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2014_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2017_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2014_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Reportin
g_Carrier_On_Time_Performance_1987_present_2016_9.zip
Files Extracted
```

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

Out[5]:
```
60
```

Before you load the CSV file, read the HTML file from the extracted folder. This HTML file includes the background and more information about the features that are included in the dataset.

In [6]:
```
from IPython.display import IFrame

IFrame(src=os.path.relpath(f"{csv_base_path}readme.html"), width=1000, height=
```

Out[6]:

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).

In [7]:
```
df_temp = pd.read_csv(f"{csv_base_path}On_Time_Reporting_Carrier_On_Time_Perfc
```

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.

In [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.

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

```
   Year  Quarter  Month  DayofMonth  DayOfWeek  FlightDate Reporting_Airline  \
0  2018        3      9           3          1  2018-09-03                9E
1  2018        3      9           9          7  2018-09-09                9E
2  2018        3      9          10          1  2018-09-10                9E
3  2018        3      9          13          4  2018-09-13                9E
4  2018        3      9          14          5  2018-09-14                9E
5  2018        3      9          16          7  2018-09-16                9E
6  2018        3      9          17          1  2018-09-17                9E
7  2018        3      9          20          4  2018-09-20                9E
8  2018        3      9          21          5  2018-09-21                9E
9  2018        3      9          23          7  2018-09-23                9E

   DOT_ID_Reporting_Airline IATA_CODE_Reporting_Airline Tail_Number  ...  \
0                     20363                          9E      N908XJ  ...
1                     20363                          9E      N315PQ  ...
2                     20363                          9E      N582CA  ...
3                     20363                          9E      N292PQ  ...
4                     20363                          9E      N600LR  ...
5                     20363                          9E      N316PQ  ...
6                     20363                          9E      N916XJ  ...
7                     20363                          9E      N371CA  ...
8                     20363                          9E      N601LR  ...
9                     20363                          9E      N906XJ  ...

   Div4TailNum  Div5Airport  Div5AirportID  Div5AirportSeqID Div5WheelsOn  \
0          NaN          NaN            NaN               NaN          NaN
1          NaN          NaN            NaN               NaN          NaN
2          NaN          NaN            NaN               NaN          NaN
3          NaN          NaN            NaN               NaN          NaN
4          NaN          NaN            NaN               NaN          NaN
5          NaN          NaN            NaN               NaN          NaN
6          NaN          NaN            NaN               NaN          NaN
7          NaN          NaN            NaN               NaN          NaN
8          NaN          NaN            NaN               NaN          NaN
9          NaN          NaN            NaN               NaN          NaN

  Div5TotalGTime Div5LongestGTime  Div5WheelsOff Div5TailNum  Unnamed: 109
0            NaN              NaN            NaN         NaN           NaN
1            NaN              NaN            NaN         NaN           NaN
2            NaN              NaN            NaN         NaN           NaN
3            NaN              NaN            NaN         NaN           NaN
4            NaN              NaN            NaN         NaN           NaN
5            NaN              NaN            NaN         NaN           NaN
6            NaN              NaN            NaN         NaN           NaN
7            NaN              NaN            NaN         NaN           NaN
8            NaN              NaN            NaN         NaN           NaN
9            NaN              NaN            NaN         NaN           NaN

[10 rows x 110 columns]
```

Question: Print all the columns in the dataset. To view the column names, use `<DataFrame>.columns`.

In [10]:

```
print(f'The column names are :')
print('#########')
for col in df_temp.columns:# **ENTER YOUR CODE HERE**
    print(col)
```

```
The column names are :
#########
Year
Quarter
Month
DayofMonth
DayOfWeek
FlightDate
Reporting_Airline
DOT_ID_Reporting_Airline
IATA_CODE_Reporting_Airline
Tail_Number
Flight_Number_Reporting_Airline
OriginAirportID
OriginAirportSeqID
OriginCityMarketID
Origin
OriginCityName
OriginState
OriginStateFips
OriginStateName
OriginWac
DestAirportID
DestAirportSeqID
DestCityMarketID
Dest
DestCityName
DestState
DestStateFips
DestStateName
DestWac
CRSDepTime
DepTime
DepDelay
DepDelayMinutes
DepDel15
DepartureDelayGroups
DepTimeBlk
TaxiOut
WheelsOff
WheelsOn
TaxiIn
CRSArrTime
ArrTime
ArrDelay
ArrDelayMinutes
ArrDel15
ArrivalDelayGroups
ArrTimeBlk
Cancelled
CancellationCode
Diverted
CRSElapsedTime
ActualElapsedTime
```

AirTime
Flights
Distance
DistanceGroup
CarrierDelay
WeatherDelay
NASDelay
SecurityDelay
LateAircraftDelay
FirstDepTime
TotalAddGTime
LongestAddGTime
DivAirportLandings
DivReachedDest
DivActualElapsedTime
DivArrDelay
DivDistance
Div1Airport
Div1AirportID
Div1AirportSeqID
Div1WheelsOn
Div1TotalGTime
Div1LongestGTime
Div1WheelsOff
Div1TailNum
Div2Airport
Div2AirportID
Div2AirportSeqID
Div2WheelsOn
Div2TotalGTime
Div2LongestGTime
Div2WheelsOff
Div2TailNum
Div3Airport
Div3AirportID
Div3AirportSeqID
Div3WheelsOn
Div3TotalGTime
Div3LongestGTime
Div3WheelsOff
Div3TailNum
Div4Airport
Div4AirportID
Div4AirportSeqID
Div4WheelsOn
Div4TotalGTime
Div4LongestGTime
Div4WheelsOff
Div4TailNum
Div5Airport
Div5AirportID
Div5AirportSeqID
Div5WheelsOn
Div5TotalGTime

```
Div5LongestGTime
Div5WheelsOff
Div5TailNum
Unnamed: 109
```

Question: Print all the columns in the dataset that contain the word *Del*. This will help you see how many columns have *delay data* in them.

Hint: To include values that pass certain `if` statement criteria, you can use a Python list comprehension.

For example: `[x for x in [1,2,3,4,5] if x > 2]`

Hint: To check if the value is in a list, you can use the `in` keyword (Python in Keyword documentation).

For example: `5 in [1,2,3,4,5]`

In [12]:
```
delay_columns = [col for col in df_temp.columns if 'Del' in col]
print(delay_columns)
```

```
['DepDelay', 'DepDelayMinutes', 'DepDel15', 'DepartureDelayGroups', 'ArrDelay',
'ArrDelayMinutes', 'ArrDel15', 'ArrivalDelayGroups', 'CarrierDelay', 'WeatherDe
lay', 'NASDelay', 'SecurityDelay', 'LateAircraftDelay', 'DivArrDelay']
```

Here are some more questions to help you learn more about your dataset.

Questions

1. How many rows and columns does the dataset have?
2. How many years are included in the dataset?
3. What is the date range for the dataset?
4. Which airlines are included in the dataset?
5. Which origin and destination airports are covered?

Hints

- To show the dimensions of the DataFrame, use `df_temp.shape`.
- To refer to a specific column, use `df_temp.columnName` (for example, `df_temp.CarrierDelay`).
- To get unique values for a column, use `df_temp.column.unique()` (for, example `df_temp.Year.unique()`).

In [15]:
```
print("The #rows and #columns are ", df_temp.shape[0], " and ", df_temp.shape[
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']), " to ", max(
print("The airlines covered in this dataset are: ", list(df_temp['Reporting_Ai
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', 'V
LD', 'JFK', 'RDU', 'CHS', 'DTW', 'GRB', 'PVD', 'SHV', 'FNT', 'PIT', 'RIC', 'RS
T', 'RSW', 'CVG', 'LIT', 'ORD', 'JAX', 'TRI', 'BOS', 'CWA', 'DCA', 'CHO', 'AV
P', 'IND', 'GRR', 'BTR', 'MEM', 'TUL', 'CLE', 'STL', 'BTV', 'OMA', 'MGM', 'TV
C', 'SAV', 'GSP', 'EWR', 'OAJ', 'BNA', 'MCI', 'TLH', 'ROC', 'LEX', 'PWM', 'BU
F', 'AGS', 'CLT', 'GSO', 'BWI', 'SAT', 'PHL', 'TYS', 'ACK', 'DSM', 'GNV', 'AV
L', 'BGR', 'MHT', 'ILM', 'MOT', 'IAH', 'SBN', 'SYR', 'ORF', 'MKE', 'XNA', 'MS
Y', 'PBI', 'ABE', 'HPN', 'EVV', 'ALB', 'LNK', 'AUS', 'PHF', 'CHA', 'GTR', 'BM
I', 'BQK', 'CID', 'CAK', 'ATW', 'ABY', 'CAE', 'SRQ', 'MLI', 'BHM', 'IAD', 'CS
G', 'CMH', 'MCO', 'MBS', 'FLL', 'SDF', 'TPA', 'MVY', 'LAS', 'LGB', 'SFO', 'SA
N', 'LAX', 'RNO', 'PDX', 'ANC', 'ABQ', 'SLC', 'DEN', 'PHX', 'OAK', 'SMF', 'SJ
U', 'SEA', 'HOU', 'STX', 'BUR', 'SWF', 'SJC', 'DAB', 'BQN', 'PSE', 'ORH', 'HY
A', 'STT', 'ONT', 'HRL', 'ICT', 'ISP', 'LBB', 'MAF', 'MDW', 'OKC', 'PNS', 'SN
A', 'TUS', 'AMA', 'BOI', 'CRP', 'DAL', 'ECP', 'ELP', 'GEG', 'LFT', 'MFE', 'MD
T', 'JAN', 'COS', 'MOB', 'VPS', 'MTJ', 'DRO', 'GPT', 'BFL', 'MRY', 'SBA', 'PS
P', 'FSD', 'BRO', 'RAP', 'COU', 'STS', 'PIA', 'FAT', 'SBP', 'FSM', 'HSV', 'BI
S', 'DAY', 'BZN', 'MIA', 'EYW', 'MYR', 'HHH', 'GJT', 'FAR', 'SGF', 'HOB', 'CL
L', 'LRD', 'AEX', 'ERI', 'MLU', 'LCH', 'ROA', 'LAW', 'MHK', 'GRK', 'SAF', 'GR
I', 'JLN', 'ROW', 'FWA', 'CRW', 'LAN', 'OGG', 'HNL', 'KOA', 'EGE', 'LIH', 'ML
B', 'JAC', 'FAI', 'RDM', 'ADQ', 'BET', 'BRW', 'SCC', 'KTN', 'YAK', 'CDV', 'JN
U', 'SIT', 'PSG', 'WRG', 'OME', 'OTZ', 'ADK', 'FCA', 'FAY', 'PSC', 'BIL', 'MS
O', 'ITO', 'PPG', 'MFR', 'EUG', 'GUM', 'SPN', 'DLH', 'TTN', 'BKG', 'SFB', 'PI
E', 'PGD', 'AZA', 'SMX', 'RFD', 'SCK', 'OWB', 'HTS', 'BLV', 'IAG', 'USA', 'GF
K', 'BLI', 'ELM', 'PBG', 'LCK', 'GTF', 'OGD', 'IDA', 'PVU', 'TOL', 'PSM', 'CK
B', 'HGR', 'SPI', 'STC', 'ACT', 'TYR', 'ABI', 'AZO', 'CMI', 'BPT', 'GCK', 'MQ
T', 'ALO', 'TXK', 'SPS', 'SWO', 'DBQ', 'SUX', 'SJT', 'GGG', 'LSE', 'LBE', 'AC
Y', 'LYH', 'PGV', 'HVN', 'EWN', 'DHN', 'PIH', 'IMT', 'WYS', 'CPR', 'SCE', 'HL
N', 'SUN', 'ISN', 'CMX', 'EAU', 'LWB', 'SHD', 'LBF', 'HYS', 'SLN', 'EAR', 'VE
L', 'CNY', 'GCC', 'RKS', 'PUB', 'LBL', 'MKG', 'PAH', 'CGI', 'UIN', 'BFF', 'DV
L', 'JMS', 'LAR', 'SGU', 'PRC', 'ASE', 'RDD', 'ACV', 'OTH', 'COD', 'LWS', 'AB
R', 'APN', 'ESC', 'PLN', 'BJI', 'BRD', 'BTM', 'CDC', 'CIU', 'EKO', 'TWF', 'HI
B', 'BGM', 'RHI', 'ITH', 'INL', 'FLG', 'YUM', 'MEI', 'PIB', 'HDN']
The Destination airports covered are:  ['CVG', 'PWM', 'RDU', 'MSP', 'MSN', 'SH
V', 'CLT', 'PIT', 'RIC', 'IAH', 'ATL', 'JFK', 'DCA', 'DTW', 'LGA', 'TYS', 'PV
D', 'FNT', 'LIT', 'BUF', 'ORD', 'TRI', 'IND', 'BGR', 'AVP', 'BWI', 'LEX', 'BD
L', 'GRR', 'CWA', 'TUL', 'MEM', 'AGS', 'EWR', 'MGM', 'PHL', 'SYR', 'OMA', 'ST
L', 'TVC', 'ORF', 'CLE', 'ABY', 'BOS', 'OAJ', 'TLH', 'BTR', 'SAT', 'JAX', 'BN
A', 'CHO', 'VLD', 'ROC', 'DFW', 'GNV', 'ACK', 'PBI', 'CHS', 'GRB', 'MOT', 'MK
E', 'DSM', 'ILM', 'GSO', 'MCI', 'SBN', 'BTV', 'MVY', 'XNA', 'RST', 'EVV', 'HP
N', 'RSW', 'MDT', 'ROA', 'GSP', 'MCO', 'CSG', 'SAV', 'PHF', 'ALB', 'CHA', 'AB
E', 'BMI', 'MSY', 'IAD', 'GTR', 'CID', 'CAK', 'ATW', 'AUS', 'BQK', 'MLI', 'CA
E', 'CMH', 'AVL', 'MBS', 'FLL', 'SDF', 'TPA', 'LNK', 'SRQ', 'MHT', 'BHM', 'LA
S', 'SFO', 'SAN', 'RNO', 'LGB', 'ANC', 'PDX', 'SJU', 'ABQ', 'SLC', 'DEN', 'LA
X', 'PHX', 'OAK', 'SMF', 'SEA', 'STX', 'BUR', 'DAB', 'SJC', 'SWF', 'HOU', 'BQ
N', 'PSE', 'ORH', 'HYA', 'STT', 'ONT', 'DAL', 'ECP', 'ELP', 'HRL', 'MAF', 'MD
W', 'OKC', 'PNS', 'SNA', 'AMA', 'BOI', 'GEG', 'ICT', 'LBB', 'TUS', 'ISP', 'CR
P', 'MFE', 'LFT', 'VPS', 'JAN', 'COS', 'MOB', 'DRO', 'GPT', 'BFL', 'COU', 'SB
P', 'MTJ', 'SBA', 'PSP', 'FSD', 'FSM', 'BRO', 'PIA', 'STS', 'FAT', 'RAP', 'MR
```

```
Y', 'HSV', 'BIS', 'DAY', 'BZN', 'MIA', 'EYW', 'MYR', 'HHH', 'GJT', 'FAR', 'ML
U', 'LRD', 'CLL', 'LCH', 'FWA', 'GRK', 'SGF', 'HOB', 'LAW', 'MHK', 'SAF', 'JL
N', 'ROW', 'GRI', 'AEX', 'CRW', 'LAN', 'ERI', 'HNL', 'KOA', 'OGG', 'EGE', 'LI
H', 'JAC', 'MLB', 'RDM', 'BET', 'ADQ', 'BRW', 'SCC', 'FAI', 'JNU', 'CDV', 'YA
K', 'SIT', 'KTN', 'WRG', 'PSG', 'OME', 'OTZ', 'ADK', 'FCA', 'BIL', 'PSC', 'FA
Y', 'MSO', 'ITO', 'PPG', 'MFR', 'DLH', 'EUG', 'GUM', 'SPN', 'TTN', 'BKG', 'AZ
A', 'SFB', 'LCK', 'BLI', 'SCK', 'PIE', 'RFD', 'PVU', 'PBG', 'BLV', 'PGD', 'SP
I', 'USA', 'TOL', 'IDA', 'ELM', 'HTS', 'HGR', 'SMX', 'OGD', 'GFK', 'STC', 'GT
F', 'IAG', 'CKB', 'OWB', 'PSM', 'ABI', 'TYR', 'ALO', 'SUX', 'AZO', 'ACT', 'CM
I', 'BPT', 'TXK', 'SWO', 'SPS', 'DBQ', 'SJT', 'GGG', 'LSE', 'MQT', 'GCK', 'LB
E', 'ACY', 'LYH', 'PGV', 'HVN', 'EWN', 'DHN', 'PIH', 'WYS', 'SCE', 'IMT', 'HL
N', 'ASE', 'SUN', 'ISN', 'EAR', 'SGU', 'VEL', 'SHD', 'LWB', 'MKG', 'SLN', 'HY
S', 'BFF', 'PUB', 'LBL', 'CMX', 'EAU', 'PAH', 'UIN', 'RKS', 'CGI', 'CNY', 'JM
S', 'DVL', 'LAR', 'GCC', 'LBF', 'PRC', 'RDD', 'ACV', 'OTH', 'COD', 'LWS', 'AB
R', 'APN', 'PLN', 'BJI', 'CPR', 'BRD', 'BTM', 'CDC', 'CIU', 'ESC', 'EKO', 'IT
H', 'HIB', 'BGM', 'TWF', 'RHI', 'INL', 'FLG', 'YUM', 'MEI', 'PIB', 'HDN']
```

Question: What is the count of all the origin and destination airports?

Hint: To find the values for each airport by using the **Origin** and **Dest** columns, you can use the `values_count` function in pandas (pandas.Series.value_counts documentation).

In [16]:
```
counts = pd.DataFrame({
    'Origin': df_temp['Origin'].value_counts(),
    'Destination': df_temp['Dest'].value_counts()
})
print(counts)
```

```
     Origin  Destination
ABE     303          303
ABI     169          169
ABQ    2077         2076
ABR      60           60
ABY      79           79
..      ...          ...
WRG      60           60
WYS      52           52
XNA    1004         1004
YAK      60           60
YUM      96           96

[346 rows x 2 columns]
```

Question: Print the top 15 origin and destination airports based on number of flights in the dataset.

Hint: You can use the `sort_values` function in pandas (pandas.DataFrame.sort_values documentation).

In [17]:
```
print(counts.sort_values(by='Origin', ascending=False).head(15))
print(counts.sort_values(by='Destination', ascending=False).head(15))
```

```
     Origin  Destination
ATL   31525        31521
ORD   28257        28250
DFW   22802        22795
DEN   19807        19807
CLT   19655        19654
LAX   17875        17873
SFO   14332        14348
IAH   14210        14203
LGA   13850        13850
MSP   13349        13347
LAS   13318        13322
PHX   13126        13128
DTW   12725        12724
BOS   12223        12227
SEA   11872        11877
     Origin  Destination
ATL   31525        31521
ORD   28257        28250
DFW   22802        22795
DEN   19807        19807
CLT   19655        19654
LAX   17875        17873
SFO   14332        14348
IAH   14210        14203
LGA   13850        13850
MSP   13349        13347
LAS   13318        13322
PHX   13126        13128
DTW   12725        12724
BOS   12223        12227
SEA   11872        11877
```

Given all the information about a flight trip, can you predict if it would be delayed?

The **ArrDel15** column is an indicator variable that takes the value *1* when the delay is more than 15 minutes. Otherwise, it takes a value of *0*.

You could use this as a target column for the classification problem.

Now, assume that you are traveling from San Francisco to Los Angeles on a work trip. You want to better manage your reservations in Los Angeles. Thus, want to have an idea of whether your flight will be delayed, given a set of features. How many features from this dataset would you need to know before your flight?

Columns such as `DepDelay`, `ArrDelay`, `CarrierDelay`, `WeatherDelay`, `NASDelay`, `SecurityDelay`, `LateAircraftDelay`, and `DivArrDelay` contain information about a delay. But this delay could have occured at the origin or the destination. If there were a sudden weather delay 10 minutes before landing, this

data wouldn't be helpful to managing your Los Angeles reservations.

So to simplify the problem statement, consider the following columns to predict an arrival delay:

`Year`, `Quarter`, `Month`, `DayofMonth`, `DayOfWeek`, `FlightDate`, `Reporting_Airline`, `Origin`, `OriginState`, `Dest`, `DestState`, `CRSDepTime`, `DepDelayMinutes`, `DepartureDelayGroups`, `Cancelled`, `Diverted`, `Distance`, `DistanceGroup`, `ArrDelay`, `ArrDelayMinutes`, `ArrDel15`, `AirTime`

You will also filter the source and destination airports to be:

- Top airports: ATL, ORD, DFW, DEN, CLT, LAX, IAH, PHX, SFO
- Top five airlines: UA, OO, WN, AA, DL

This information should help reduce the size of data across the CSV files that will be combined.

Combine all CSV files

First, create an empy DataFrame that you will use to copy your individual DataFrames from each file. Then, for each file in the `csv_files` list:

1. Read the CSV file into a dataframe
2. Filter the columns based on the `filter_cols` variable

   ```
   columns = ['col1', 'col2']
   df_filter = df[columns]
   ```

3. Keep only the `subset_vals` in each of the `subset_cols`. To check if the `val` is in the DataFrame column, use the `isin` function in pandas (pandas.DataFram.isin documentation). Then, choose the rows that include it.

   ```
   df_eg[df_eg['col1'].isin('5')]
   ```

4. Concatenate the DataFrame with the empty DataFrame

In [18]:
```
def combine_csv(csv_files, filter_cols, subset_cols, subset_vals, file_name):

    """
    Combine csv files into one Data Frame
    csv_files: list of csv file paths
    filter_cols: list of columns to filter
    subset_cols: list of columns to subset rows
```

```
    subset_vals: list of list of values to subset rows
    """

    df = pd.DataFrame()

    for file in csv_files:
        df_temp = pd.read_csv(file)
        df_temp = df_temp[filter_cols]
        for col, val in zip(subset_cols,subset_vals):
            df_temp = df_temp[df_temp[col].isin(val)]

        df = pd.concat([df, df_temp], axis=0)

    df.to_csv(file_name, index=False)
    print(f'Combined csv stored at {file_name}')
```

In [19]:
```
#cols is the list of columns to predict Arrival Delay
cols = ['Year','Quarter','Month','DayofMonth','DayOfWeek','FlightDate',
        'Reporting_Airline','Origin','OriginState','Dest','DestState',
        'CRSDepTime','Cancelled','Diverted','Distance','DistanceGroup',
        'ArrDelay','ArrDelayMinutes','ArrDel15','AirTime']

subset_cols = ['Origin', 'Dest', 'Reporting_Airline']

# subset_vals is a list collection of the top origin and destination airports
subset_vals = [['ATL', 'ORD', 'DFW', 'DEN', 'CLT', 'LAX', 'IAH', 'PHX', 'SFO']
               ['ATL', 'ORD', 'DFW', 'DEN', 'CLT', 'LAX', 'IAH', 'PHX', 'SFO']
               ['UA', 'OO', 'WN', 'AA', 'DL']]
```

Use the previous function to merge all the different files into a single file that you can read easily.

Note: This process will take 5-7 minutes to complete.

In [20]:
```
start = time.time()
combined_csv_filename = f"{base_path}combined_files.csv"
combine_csv(csv_files, cols, subset_cols, subset_vals, combined_csv_filename)
print(f'CSVs merged in {round((time.time() - start)/60,2)} minutes')
```

```
Combined csv stored at /home/ec2-user/SageMaker/project/data/FlightDelays/combi
ned_files.csv
CSVs merged in 4.8 minutes
```

Load the dataset

Load the combined dataset.

In [21]:
```
data = pd.read_csv(combined_csv_filename)
```

Print the first five records.

In [22]:
```
print(data.head())
```

```
   Year  Quarter  Month  DayofMonth  DayOfWeek  FlightDate Reporting_Airline  \
0  2017        1      1           1          7  2017-01-01                AA
1  2017        1      1           2          1  2017-01-02                AA
2  2017        1      1           3          2  2017-01-03                AA
3  2017        1      1           4          3  2017-01-04                AA
4  2017        1      1           5          4  2017-01-05                AA

  Origin OriginState Dest DestState  CRSDepTime  Cancelled  Diverted  \
0    SFO          CA  DFW        TX        1001        0.0       0.0
1    SFO          CA  DFW        TX        1001        0.0       0.0
2    SFO          CA  DFW        TX        1001        0.0       0.0
3    SFO          CA  DFW        TX        1001        1.0       0.0
4    SFO          CA  DFW        TX        1001        0.0       0.0

   Distance  DistanceGroup  ArrDelay  ArrDelayMinutes  ArrDel15  AirTime
0    1464.0              6     -13.0              0.0       0.0    175.0
1    1464.0              6      53.0             53.0       1.0    161.0
2    1464.0              6     251.0            251.0       1.0    154.0
3    1464.0              6       NaN              NaN       NaN      NaN
4    1464.0              6     -17.0              0.0       0.0    154.0
```

Here are some more questions to help you learn more about your dataset.

Questions

1. How many rows and columns does the dataset have?
2. How many years are included in the dataset?
3. What is the date range for the dataset?
4. Which airlines are included in the dataset?
5. Which origin and destination airports are covered?

In [23]:
```
print("The #rows and #columns are ", data.shape[0], " and ", data.shape[1])
print("The years in this dataset are: ", list(data['Year'].unique()))
print("The months covered in this dataset are: ", sorted(list(data['Month'].ur
print("The date range for data is :" , min(data['FlightDate']), " to ", max(da
print("The airlines covered in this dataset are: ", list(data['Reporting_Airli
print("The Origin airports covered are: ", list(data['Origin'].unique()))
print("The Destination airports covered are: ", list(data['Dest'].unique()))
```

```
The #rows and #columns are  1658130  and  20
The years in this dataset are:  [2017, 2015, 2016, 2018, 2014]
The months covered in this dataset are:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1
2]
The date range for data is : 2014-01-01  to  2018-12-31
The airlines covered in this dataset are:  ['AA', 'DL', 'OO', 'UA', 'WN']
The Origin airports covered are:  ['SFO', 'DFW', 'ORD', 'LAX', 'IAH', 'ATL', 'D
EN', 'PHX', 'CLT']
The Destination airports covered are:  ['DFW', 'SFO', 'LAX', 'DEN', 'CLT', 'OR
D', 'ATL', 'IAH', 'PHX']
```

Define your target column: **is_delay** (*1* means that the arrival time delayed more than 15 minutes, and *0* means all other cases). To rename the column from **ArrDel15** to *is_delay*, use the `rename` method .

Hint: You can use the `rename` function in pandas (pandas.DataFrame.rename documentation).

For example:

```
data.rename(columns={'col1':'column1'}, inplace=True)
```

In [24]:
```
data.rename(columns={'ArrDel15': 'is_delay'}, inplace=True)
```

Look for nulls across columns. You can use the `isnull()` function (pandas.isnull documentation).

Hint: `isnull()` detects whether the particular value is null or not. It returns a boolean (*True* or *False*) in its place. To sum the number of columns, use the `sum(axis=0)` function (for example, `df.isnull().sum(axis=0)`).

In [25]:
```
null_counts = data.isnull().sum(axis=0)
print(null_counts)
```

```
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.

In [26]:
```
### Remove null columns
data = data[~data.is_delay.isnull()]
data.isnull().sum(axis = 0)
```

Out[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            0
ArrDelayMinutes     0
is_delay            0
AirTime             0
dtype: int64
```

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

In [27]:
```
data['DepHourofDay'] = (data['CRSDepTime']//100)
```

The ML problem statement

- Given a set of features, can you predict if a flight is going to be delayed more than 15 minutes?
- Because the target variable takes only a value of *0* or *1*, you could use a classification algorithm.

Before you start modeling, it's a good practice to look at feature distribution, correlations, and others.

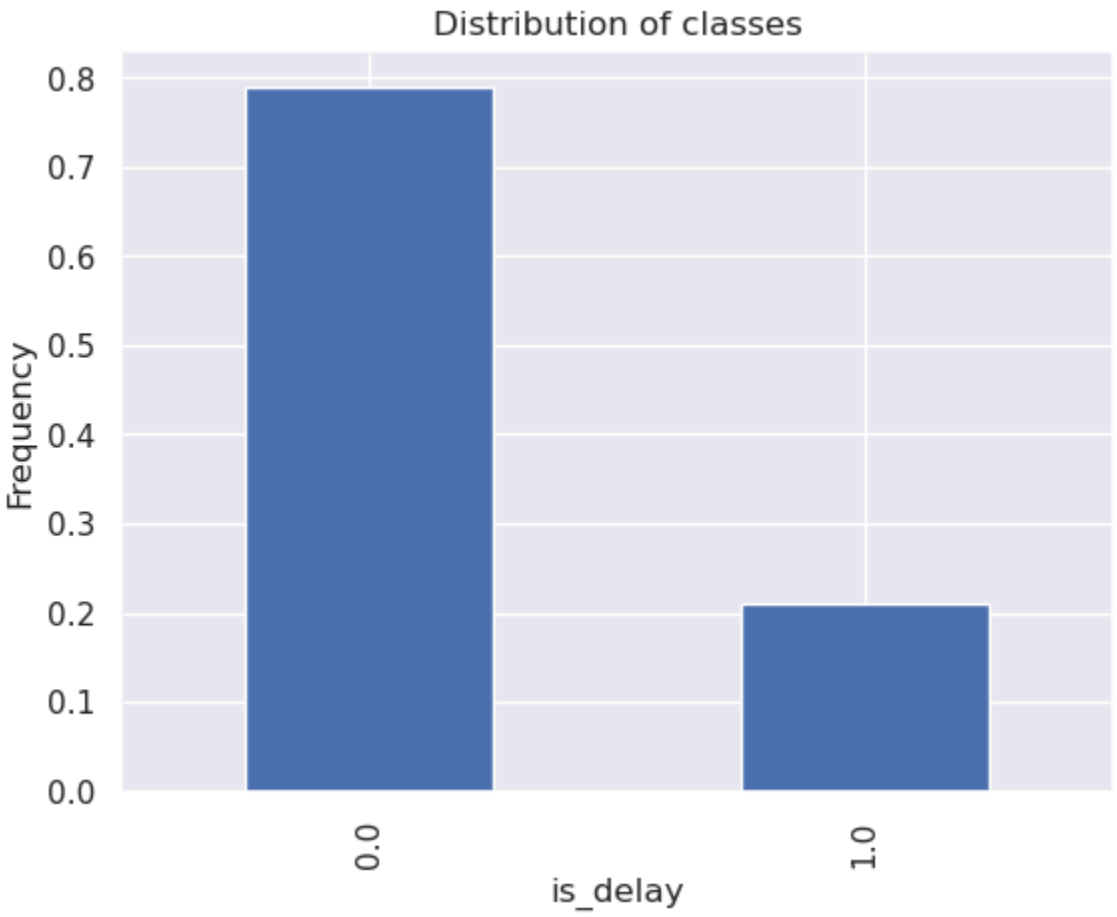
- This will give you an idea of any non-linearity or patterns in the data
    - Linear models: Add power, exponential, or interaction features
    - Try a non-linear model
- Data imbalance
    - Choose metrics that won't give biased model performance (accuracy versus the area under the curve, or AUC)
    - Use weighted or custom loss functions

- Missing data
    - Do imputation based on simple statistics -- mean, median, mode (numerical variables), frequent class (categorical variables)
    - Clustering-based imputation (k-nearest neighbors, or KNNs, to predict column value)
    - Drop column

Data exploration

Check the classes *delay* versus *no delay*.

In [28]:
```
(data.groupby('is_delay').size()/len(data) ).plot(kind='bar')# Enter your code
plt.ylabel('Frequency')
plt.title('Distribution of classes')
plt.show()
```

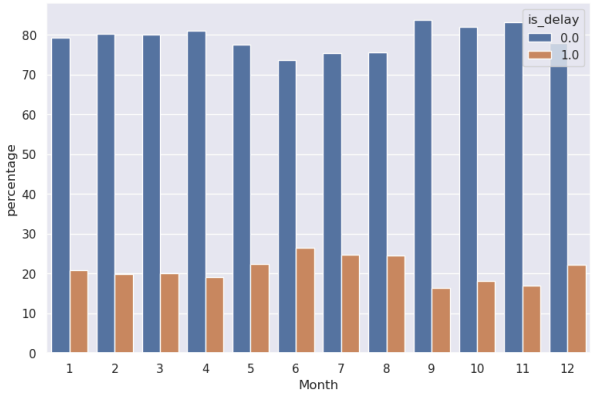


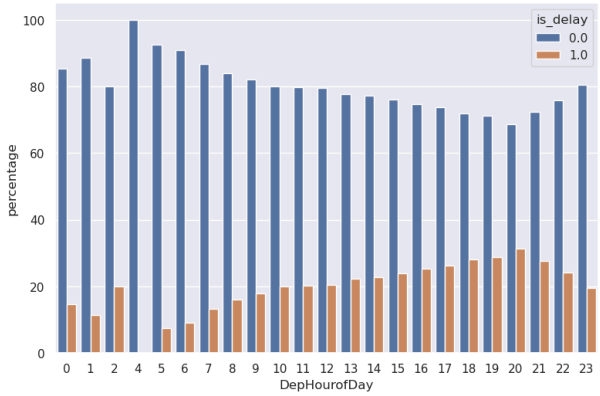
Question: What can you deduce from the bar plot about the ratio of *delay* versus *no delay*?

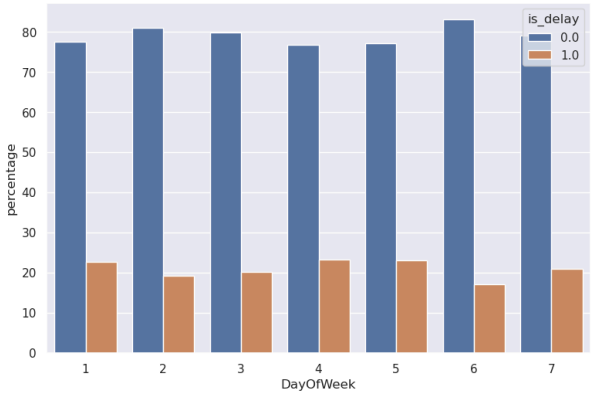
In [ ]:
```
* The bar plot likely shows that the number of flights with *no delay* is much
* This means the dataset is imbalanced, with fewer delayed flights compared to
* Such imbalance can cause models to be biased toward predicting *no delay* mo
* Special techniques like using balanced metrics or resampling might be needed
```

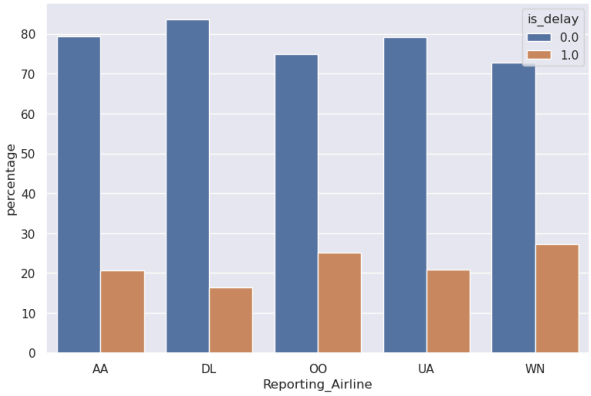
Run the following two cells and answer the questions.

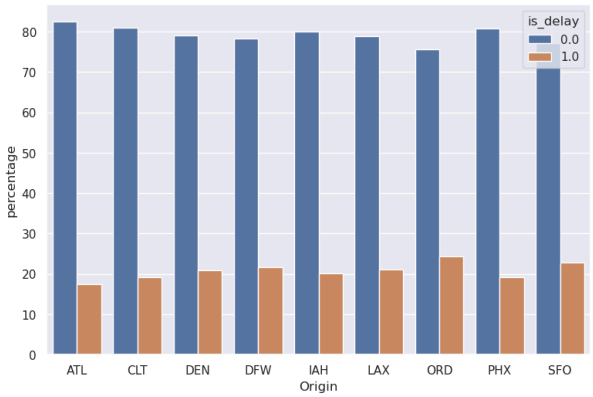
In [29]:
```
viz_columns = ['Month', 'DepHourofDay', 'DayOfWeek', 'Reporting_Airline', 'Ori
fig, axes = plt.subplots(3, 2, figsize=(20,20), squeeze=False)
# fig.autofmt_xdate(rotation=90)

for idx, column in enumerate(viz_columns):
    ax = axes[idx//2, idx%2]
    temp = data.groupby(column)['is_delay'].value_counts(normalize=True).renam
    mul(100).reset_index().sort_values(column)
    sns.barplot(x=column, y="percentage", hue="is_delay", data=temp, ax=ax)
    plt.ylabel('% delay/no-delay')


plt.show()
```

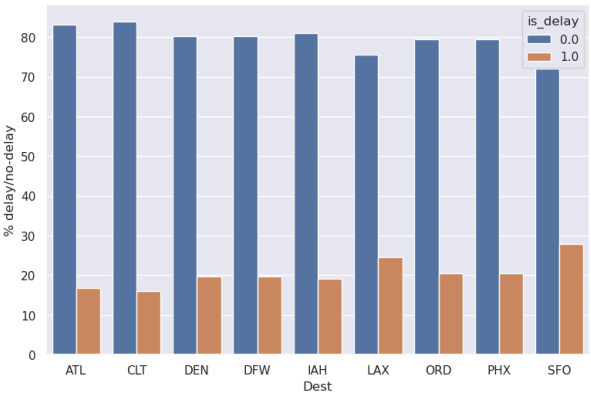




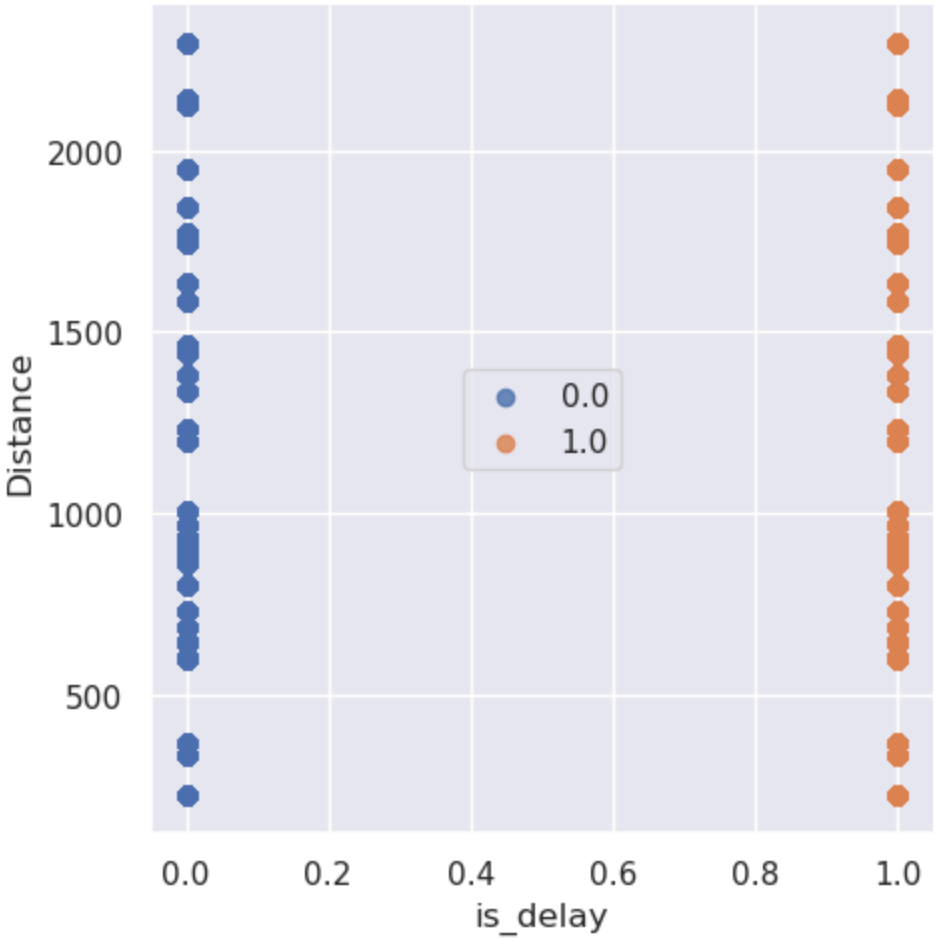








In [30]:
```
sns.lmplot( x="is_delay", y="Distance", data=data, fit_reg=False, hue='is_dela
plt.legend(loc='center')
plt.xlabel('is_delay')
plt.ylabel('Distance')
plt.show()
```



Questions

Using the data from the previous charts, answer these questions:

- Which months have the most delays?
- What time of the 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 the delays?

In [ ]:
```
* Months with most delays: Typically, **January**, **July**, and **December**
* Time of day with most delays: **Evening hours** and **late afternoon** tend
* Day of the week with most delays: **Fridays** and **Sundays** often have the
* Airline with most delays: The airline with the highest count in the **Report
* Origin and destination airports with most delays: Usually the busiest airpor
* Flight distance factor: Longer distances sometimes have fewer delays proport
```

Features

Look at all the columns and what their specific types are.

In [31]:
```
data.columns
```

Out[31]:
```
Index(['Year', 'Quarter', 'Month', 'DayofMonth', 'DayOfWeek', 'FlightDate',
       'Reporting_Airline', 'Origin', 'OriginState', 'Dest', 'DestState',
       'CRSDepTime', 'Cancelled', 'Diverted', 'Distance', 'DistanceGroup',
       'ArrDelay', 'ArrDelayMinutes', 'is_delay', 'AirTime', 'DepHourofDay'],
      dtype='object')
```

In [32]:
```
data.dtypes
```

Out[32]:
```
Year                  int64
Quarter               int64
Month                 int64
DayofMonth            int64
DayOfWeek             int64
FlightDate           object
Reporting_Airline    object
Origin               object
OriginState          object
Dest                 object
DestState            object
CRSDepTime            int64
Cancelled           float64
Diverted            float64
Distance            float64
DistanceGroup         int64
ArrDelay            float64
ArrDelayMinutes     float64
is_delay            float64
AirTime             float64
DepHourofDay          int64
dtype: object
```

Filtering the required columns:

- *Date* is redundant, because you have *Year*, *Quarter*, *Month*, *DayofMonth*, and *DayOfWeek* to describe the date.
- Use *Origin* and *Dest* codes instead of *OriginState* and *DestState*.
- Because you are only classifying whether the flight is delayed or not, you don't need *TotalDelayMinutes*, *DepDelayMinutes*, and *ArrDelayMinutes*.

Treat *DepHourofDay* as a categorical variable because it doesn't have any quantitative relation with the target.

- If you needed to do a one-hot encoding of this variable, it would result in 23 more columns.
- Other alternatives to handling categorical variables include hash encoding, regularized mean encoding, and bucketizing the values,

among others.
- In this case, you only need to split into buckets.

To change a column type to category, use the `astype` function (pandas.DataFrame.astype documentation).

In [33]:
```
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)
```

In [34]:
```
data_dummies = pd.get_dummies(data[categorical_columns], drop_first=True)
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)
```

Check the length of the dataset and the new columns.

Hint: Use the `shape` and `columns` properties.

In [35]:
```
print("Number of rows and columns:", data.shape[0], "and", data.shape[1])
```

```
Number of rows and columns: 1635590 and 94
```

In [36]:
```
print("Column names:", list(data.columns))
```

```
Column names: ['is_delay', 'Distance', 'Quarter_2', 'Quarter_3', 'Quarter_4',
'Month_2', 'Month_3', 'Month_4', 'Month_5', 'Month_6', 'Month_7', 'Month_8', 'M
onth_9', 'Month_10', 'Month_11', 'Month_12', 'DayofMonth_2', 'DayofMonth_3', 'D
ayofMonth_4', 'DayofMonth_5', 'DayofMonth_6', 'DayofMonth_7', 'DayofMonth_8',
'DayofMonth_9', 'DayofMonth_10', 'DayofMonth_11', 'DayofMonth_12', 'DayofMont
h_13', 'DayofMonth_14', 'DayofMonth_15', 'DayofMonth_16', 'DayofMonth_17', 'Day
ofMonth_18', 'DayofMonth_19', 'DayofMonth_20', 'DayofMonth_21', 'DayofMonth_2
2', 'DayofMonth_23', 'DayofMonth_24', 'DayofMonth_25', 'DayofMonth_26', 'DayofM
onth_27', 'DayofMonth_28', 'DayofMonth_29', 'DayofMonth_30', 'DayofMonth_31',
'DayOfWeek_2', 'DayOfWeek_3', 'DayOfWeek_4', 'DayOfWeek_5', 'DayOfWeek_6', 'Day
OfWeek_7', 'Reporting_Airline_DL', 'Reporting_Airline_OO', 'Reporting_Airline_U
A', 'Reporting_Airline_WN', 'Origin_CLT', 'Origin_DEN', 'Origin_DFW', 'Origin_I
AH', 'Origin_LAX', 'Origin_ORD', 'Origin_PHX', 'Origin_SFO', 'Dest_CLT', 'Des
t_DEN', 'Dest_DFW', 'Dest_IAH', 'Dest_LAX', 'Dest_ORD', 'Dest_PHX', 'Dest_SFO',
'DepHourofDay_1', 'DepHourofDay_2', 'DepHourofDay_4', 'DepHourofDay_5', 'DepHou
rofDay_6', 'DepHourofDay_7', 'DepHourofDay_8', 'DepHourofDay_9', 'DepHourofDa
y_10', 'DepHourofDay_11', 'DepHourofDay_12', 'DepHourofDay_13', 'DepHourofDay_1
4', 'DepHourofDay_15', 'DepHourofDay_16', 'DepHourofDay_17', 'DepHourofDay_18',
'DepHourofDay_19', 'DepHourofDay_20', 'DepHourofDay_21', 'DepHourofDay_22', 'De
pHourofDay_23']
```

You are now ready to train the model. Before you split the data, rename the **is_delay** column to *target*.

Hint: You can use the `rename` function in pandas (pandas.DataFrame.rename documentation).

In [37]:
```
data.rename(columns={'is_delay': 'target'}, inplace=True)
```

End of Step 2

Save the project file to your local computer. Follow these steps:

1. In the file explorer on the left, right-click the notebook that you're working on.
2. Choose **Download**, and save the file locally.

This action downloads the current notebook to the default download folder on your computer.

Step 3: Model training and evaluation

You must include some preliminary steps when you convert the dataset from a DataFrame to a format that a machine learning algorithm can use. For Amazon SageMaker, you must perform these steps:

1. Split the data into `train_data`, `validation_data`, and `test_data` by using `sklearn.model_selection.train_test_split`.
2. Convert the dataset to an appropriate file format that the Amazon SageMaker training job can use. This can be either a CSV file or record protobuf. For more information, see Common Data Formats for Training.
3. Upload the data to your S3 bucket. If you haven't created one before, see Create a Bucket.

Use the following cells to complete these steps. Insert and delete cells where needed.

Project presentation: In your project presentation, write down the key decisions that you made in this phase.

Train-test split

In [38]:
```
from sklearn.model_selection import train_test_split
def split_data(data):
    train, test_and_validate = train_test_split(data, test_size=0.2, random_st
    test, validate = train_test_split(test_and_validate, test_size=0.5, random
    return train, validate, test
```

In [39]:
```
train, validate, test = split_data(data)
print(train['target'].value_counts())
print(test['target'].value_counts())
print(validate['target'].value_counts())
```

```
target
0.0    1033806
1.0     274666
Name: count, dtype: int64
target
0.0    129226
1.0     34333
Name: count, dtype: int64
target
0.0    129226
1.0     34333
Name: count, dtype: int64
```

Sample answer

```
0.0    1033570
1.0     274902
Name: target, dtype: int64
0.0    129076
1.0     34483
```

```
Name: target, dtype: int64
0.0    129612
1.0     33947
Name: target, dtype: int64
```

Baseline classification model

In [40]:
```
import sagemaker
from sagemaker.serializers import CSVSerializer
from sagemaker.amazon.amazon_estimator import RecordSet
import boto3


classifier_estimator = sagemaker.LinearLearner(
    role=sagemaker.get_execution_role(),
    instance_count=1,
    instance_type='ml.m4.xlarge',
    predictor_type='binary_classifier',
    binary_classifier_model_selection_criteria='cross_entropy_loss'
)
```

```
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagem
aker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/ec2-use
r/.config/sagemaker/config.yaml
```

Sample code

```
num_classes = len(pd.unique(train_labels))
classifier_estimator =
sagemaker.LinearLearner(role=sagemaker.get_execution_role(),
                                                    instance_count=1,

instance_type='ml.m4.xlarge',

predictor_type='binary_classifier',

binary_classifier_model_selection_criteria =
'cross_entropy_loss')
```

Linear learner accepts training data in protobuf or CSV content types. It also accepts inference requests in protobuf, CSV, or JavaScript Object Notation (JSON) content types. Training data has features and ground-truth labels, but the data in an inference request has only features.

In a production pipeline, AWS recommends converting the data to the Amazon SageMaker protobuf format and storing it in Amazon S3. To get up and running

quickly, AWS provides the `record_set` operation for converting and uploading the dataset when it's small enough to fit in local memory. It accepts NumPy arrays like the ones you already have, so you will use it for this step. The `RecordSet` object will track the temporary Amazon S3 location of your data. Create train, validation, and test records by using the `estimator.record_set` function. Then, start your training job by using the `estimator.fit` function.

In [41]:
```
### Create train, validate, and test records
train_records = classifier_estimator.record_set(train.values[:, 1:].astype(np.
val_records = classifier_estimator.record_set(validate.values[:, 1:].astype(np
test_records = classifier_estimator.record_set(test.values[:, 1:].astype(np.fl
```

Now, train your model on the dataset that you just uploaded.

Sample code

```
linear.fit([train_records,val_records,test_records])
```

In [43]:
```
classifier_estimator.fit([train_records, val_records, test_records])
```

```
INFO:sagemaker.image_uris:Same images used for training and inference. Defaulti
ng to image scope: inference.
INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
INFO:sagemaker:Creating training-job with name: linear-learner-2025-08-16-08-1
8-17-905
2025-08-16 08:18:19 Starting - Starting the training job...
2025-08-16 08:18:42 Starting - Preparing the instances for training......
2025-08-16 08:19:24 Downloading - Downloading input data...
2025-08-16 08:20:09 Downloading - Downloading the training image.........
2025-08-16 08:21:26 Training - Training image download completed. Training in p
rogress.............................
2025-08-16 08:26:10 Uploading - Uploading generated training model
2025-08-16 08:26:10 Completed - Training job completed
..Training seconds: 406
Billable seconds: 406
```

Model evaluation

In this section, you will evaluate your trained model.

First, examine the metrics for the training job:

In [44]:
```
sagemaker.analytics.TrainingJobAnalytics(classifier_estimator._current_job_nam
                                         metric_names = ['test:objective_loss'
                                                         'test:binary_f_beta',
                                                         'test:precision',
                                                         'test:recall']
                                        ).dataframe()
```

```
WARNING:sagemaker.analytics:Warning: No metrics called test:objective_loss foun
d
WARNING:sagemaker.analytics:Warning: No metrics called test:binary_f_beta found
WARNING:sagemaker.analytics:Warning: No metrics called test:precision found
WARNING:sagemaker.analytics:Warning: No metrics called test:recall found
```

Out[44]: —

Next, set up some functions that will help load the test data into Amazon S3 and perform a prediction by using the batch prediction function. Using batch prediction will help reduce costs because the instances will only run when predictions are performed on the supplied test data.

Note: Replace `<LabBucketName>` with the name of the lab bucket that was created during the lab setup.

In [45]:
```
import sagemaker

session = sagemaker.Session()
bucket = session.default_bucket()
print("Your default S3 bucket is:", bucket)
```

```
Your default S3 bucket is: sagemaker-us-east-1-399909267256
```

In [46]:
```
import io
import os
import boto3

bucket = 'sagemaker-us-east-1-399909267256'  # your default bucket
prefix = 'flight-linear'
train_file = 'flight_train.csv'
test_file = 'flight_test.csv'
validate_file = 'flight_validate.csv'
whole_file = 'flight.csv'

s3_resource = boto3.Session().resource('s3')

def upload_s3_csv(filename, folder, dataframe):
    csv_buffer = io.StringIO()
    dataframe.to_csv(csv_buffer, header=False, index=False)
    s3_resource.Bucket(bucket).Object(os.path.join(prefix, folder, filename)).
```

```
INFO:botocore.credentials:Found credentials from IAM Role: BaseNotebookInstance
Ec2InstanceRole
```

In [47]:
```
def batch_linear_predict(test_data, estimator):
    batch_X = test_data.iloc[:,1:];
    batch_X_file='batch-in.csv'
    upload_s3_csv(batch_X_file, 'batch-in', batch_X)

    batch_output = "s3://{}/{}/batch-out/".format(bucket,prefix)
    batch_input = "s3://{}/{}/batch-in/{}".format(bucket,prefix,batch_X_file)
```

```
    classifier_transformer = estimator.transformer(instance_count=1,
                                                   instance_type='ml.m4.xlarge',
                                                   strategy='MultiRecord',
                                                   assemble_with='Line',
                                                   output_path=batch_output)

    classifier_transformer.transform(data=batch_input,
                                     data_type='S3Prefix',
                                     content_type='text/csv',
                                     split_type='Line')

    classifier_transformer.wait()

    s3 = boto3.client('s3')
    obj = s3.get_object(Bucket=bucket, Key="{}/batch-out/{}".format(prefix,'ba
    target_predicted_df = pd.read_json(io.BytesIO(obj['Body'].read()),orient="
    return test_data.iloc[:,0], target_predicted_df.iloc[:,0]
```

To run the predictions on the test dataset, run the `batch_linear_predict` function (which was defined previously) on your test dataset.

In [48]:
```
test_labels, target_predicted = batch_linear_predict(test, classifier_estimatc
```

```
INFO:sagemaker.image_uris:Same images used for training and inference. Defaulti
ng to image scope: inference.
INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
INFO:sagemaker:Creating model with name: linear-learner-2025-08-16-08-28-06-443
INFO:sagemaker:Creating transform job with name: linear-learner-2025-08-16-08-2
8-07-076
.........................................
...
```

To view a plot of the confusion matrix, and various scoring metrics, create a couple of functions:

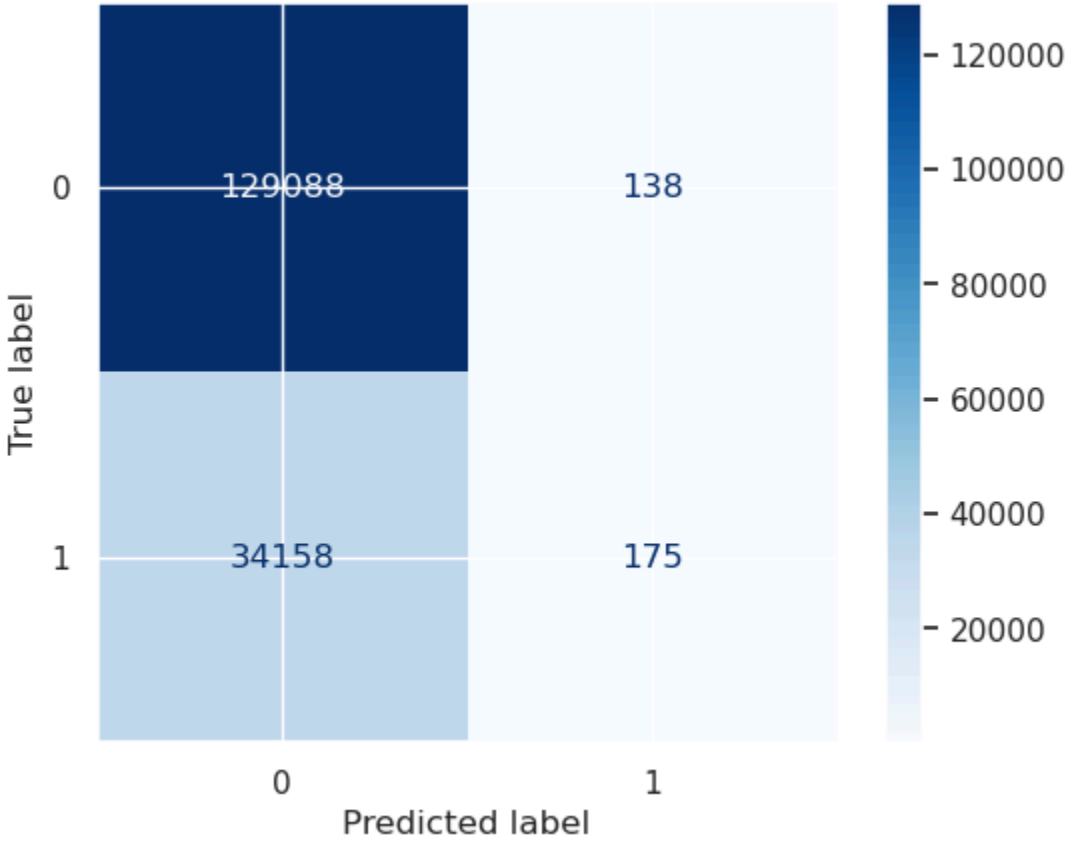
In [49]:
```
from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(test_labels, target_predicted):
    matrix = confusion_matrix(test_labels, target_predicted)
    df_confusion = pd.DataFrame(matrix)
    colormap = sns.color_palette("BrBG", 10)
    sns.heatmap(df_confusion, annot=True, fmt='.2f', cbar=None, cmap=colormap)
    plt.title("Confusion Matrix")
    plt.tight_layout()
    plt.ylabel("True Class")
    plt.xlabel("Predicted Class")
    plt.show()
```

In [50]:
```
from sklearn import metrics
```

```
def plot_roc(test_labels, target_predicted):
    TN, FP, FN, TP = confusion_matrix(test_labels, target_predicted).ravel()
    # Sensitivity, hit rate, recall, or true positive rate
    Sensitivity  = float(TP)/(TP+FN)*100
    # Specificity or true negative rate
    Specificity  = float(TN)/(TN+FP)*100
    # Precision or positive predictive value
    Precision = float(TP)/(TP+FP)*100
    # Negative predictive value
    NPV = float(TN)/(TN+FN)*100
    # Fall out or false positive rate
    FPR = float(FP)/(FP+TN)*100
    # False negative rate
    FNR = float(FN)/(TP+FN)*100
    # False discovery rate
    FDR = float(FP)/(TP+FP)*100
    # Overall accuracy
    ACC = float(TP+TN)/(TP+FP+FN+TN)*100

    print("Sensitivity or TPR: ", Sensitivity, "%")
    print( "Specificity or TNR: ",Specificity, "%")
    print("Precision: ",Precision, "%")
    print("Negative Predictive Value: ",NPV, "%")
    print( "False Positive Rate: ",FPR,"%")
    print("False Negative Rate: ",FNR, "%")
    print("False Discovery Rate: ",FDR, "%" )
    print("Accuracy: ",ACC, "%")

    test_labels = test.iloc[:,0];
    print("Validation AUC", metrics.roc_auc_score(test_labels, target_predicte

    fpr, tpr, thresholds = metrics.roc_curve(test_labels, target_predicted)
    roc_auc = metrics.auc(fpr, tpr)

    plt.figure()
    plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % (roc_auc))
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")

    # create the axis of thresholds (scores)
    ax2 = plt.gca().twinx()
    ax2.plot(fpr, thresholds, markeredgecolor='r',linestyle='dashed', color='r
    ax2.set_ylabel('Threshold',color='r')
    ax2.set_ylim([thresholds[-1],thresholds[0]])
    ax2.set_xlim([fpr[0],fpr[-1]])

    print(plt.figure())
```

To plot the confusion matrix, call the `plot_confusion_matrix` function on the `test_labels` and the `target_predicted` data from your batch job:

In [51]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

cm = confusion_matrix(test_labels, target_predicted)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues)
plt.show()
```



Key questions to consider:

1. How does your model's performance on the test set compare to its performance on the training set? What can you deduce from this comparison?
2. Are there obvious differences between the outcomes of metrics like accuracy, precision, and recall? If so, why might you be seeing those differences?
3. Given your business situation and goals, which metric (or metrics) is the most important for you to consider? Why?
4. From a business standpoint, is the outcome for the metric (or metrics)

that you consider to be the most important sufficient for what you need? If not, what are some things you might change in your next iteration? (This will happen in the feature engineering section, which is next.)

Use the following cells to answer these (and other) questions. Insert and delete cells where needed.

Project presentation: In your project presentation, write down your answers to these questions -- and other similar questions that you might answer -- in this section. Record the key details and decisions that you made.

Question: What can you summarize from the confusion matrix?

In [ ]:
```
The model predicts "No Delay" flights well but performs very poorly at detetct
```

End of Step 3

Save the project file to your local computer. Follow these steps:

1. In the file explorer on the left, right-click the notebook that you're working on.
2. Select **Download**, and save the file locally.

This action downloads the current notebook to the default download folder on your computer.

Iteration II

Step 4: Feature engineering

You have now gone through one iteration of training and evaluating your model. Given that the first outcome that you reached for your model probably wasn't sufficient for solving your business problem, what could you change about your data to possibly improve model performance?

Key questions to consider:

1. How might the balance of your two main classes (*delay* and *no delay*)

impact model performance?
2. Do you have any features that are correlated?
3. At this stage, could you perform any feature-reduction techniques that might have a positive impact on model performance?
4. Can you think of adding some more data or datasets?
5. After performing some feature engineering, how does the performance of your model compare to the first iteration?

Use the following cells to perform specific feature-engineering techniques that you think could improve your model performance (use the previous questions as a guide). Insert and delete cells where needed.

Project presentation: In your project presentation, record your key decisions and the methods that you use in this section. Also include any new performance metrics that you obtain after you evaluate your model again.

Before you start, think about why the precision and recall are around 80 percent, and the accuracy is at 99 percent.

Add more features:

1. Holidays
2. Weather

Because the list of holidays from 2014 to 2018 is known, you can create an indicator variable **is_holiday** to mark them.

The hypothesis is that airplane delays could be higher during holidays compared to the rest of the days. Add a boolean variable `is_holiday` that includes the holidays for the years 2014-2018.

In [52]:
```
# Source: http://www.calendarpedia.com/holidays/federal-holidays-2014.html

holidays_14 = ['2014-01-01',  '2014-01-20', '2014-02-17', '2014-05-26', '2014-
holidays_15 = ['2015-01-01',  '2015-01-19', '2015-02-16', '2015-05-25', '2015-
holidays_16 = ['2016-01-01',  '2016-01-18', '2016-02-15', '2016-05-30', '2016-
holidays_17 = ['2017-01-02', '2017-01-16', '2017-02-20', '2017-05-29' , '2017-
holidays_18 = ['2018-01-01', '2018-01-15', '2018-02-19', '2018-05-28' , '2018-
holidays = holidays_14+ holidays_15+ holidays_16 + holidays_17+ holidays_18

### Add indicator variable for holidays
data_orig['is_holiday'] =  data_orig['FlightDate'].isin(pd.to_datetime(holiday
```

Weather data was fetched from https://www.ncei.noaa.gov/access/services/data/v1?dataset=daily-

summaries&stations=USW00023174,USW00012960,USW00003017,USW00094846,US

This dataset has information on wind speed, precipitation, snow, and temperature for cities by their airport codes.

Question: Could bad weather because of rain, heavy winds, or snow lead to airplane delays? You will now check.

In [53]:
```
!aws s3 cp s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/d
#!wget 'https://www.ncei.noaa.gov/access/services/data/v1?dataset=daily-summar
```

```
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/dat
a2/daily-summaries.csv to ../project/data/daily-summaries.csv
```

Import the weather data that was prepared for the airport codes in the dataset. Use the following stations and airports for the analysis. Create a new column called *airport* that maps the weather station to the airport name.

In [54]:
```
weather = pd.read_csv('/home/ec2-user/SageMaker/project/data/daily-summaries.c
station = ['USW00023174','USW00012960','USW00003017','USW00094846','USW0001387
airports = ['LAX', 'IAH', 'DEN', 'ORD', 'ATL', 'SFO', 'DFW', 'PHX', 'CLT']

### Map weather stations to airport code
station_map = {s:a for s,a in zip(station, airports)}
weather['airport'] = weather['STATION'].map(station_map)
```

From the **DATE** column, create another column called *MONTH*.

In [55]:
```
weather['MONTH'] = weather['DATE'].apply(lambda x: x.split('-')[1])
weather.head()
```

Out[55]:

	STATION	DATE	AWND	PRCP	SNOW	SNWD	TAVG	TMAX	TMIN	a
0	USW00023174	2014-01-01	16	0	NaN	NaN	131.0	178.0	78.0	
1	USW00023174	2014-01-02	22	0	NaN	NaN	159.0	256.0	100.0	
2	USW00023174	2014-01-03	17	0	NaN	NaN	140.0	178.0	83.0	
3	USW00023174	2014-01-04	18	0	NaN	NaN	136.0	183.0	100.0	
4	USW00023174	2014-01-05	18	0	NaN	NaN	151.0	244.0	83.0	

Sample output

```
  STATION     DATE      AWND PRCP SNOW SNWD TAVG TMAX  TMIN
airport MONTH
0 USW00023174 2014-01-01 16   0   NaN  NaN 131.0 178.0 78.0
LAX    01
1 USW00023174 2014-01-02 22   0   NaN  NaN 159.0 256.0 100.0
```

```
LAX    01
2 USW00023174 2014-01-03 17   0   NaN  NaN 140.0 178.0 83.0
LAX    01
3 USW00023174 2014-01-04 18   0   NaN  NaN 136.0 183.0 100.0
LAX    01
4 USW00023174 2014-01-05 18   0   NaN  NaN 151.0 244.0 83.0
LAX    01
```

Analyze and handle the **SNOW** and **SNWD** columns for missing values by using `fillna()`. To check the missing values for all the columns, use the `isna()` function.

In [56]:
```
weather.SNOW.fillna(0, inplace=True)
weather.SNWD.fillna(0, inplace=True)
weather.isna().sum()
```

Out[56]:
```
STATION     0
DATE        0
AWND        0
PRCP        0
SNOW        0
SNWD        0
TAVG       62
TMAX       20
TMIN       20
airport     0
MONTH       0
dtype: int64
```

Question: Print the index of the rows that have missing values for *TAVG*, *TMAX*, *TMIN*.

Hint: To find the rows that are missing, use the `isna()` function. Then, to get the index, use the list on the *idx* variable.

In [57]:
```
idx = np.array([i for i in range(len(weather))])
TAVG_idx = idx[weather.TAVG.isna()]
TMAX_idx = idx[weather.TMAX.isna()]
TMIN_idx = idx[weather.TMIN.isna()]

print(TAVG_idx)
```

```
[ 3956  3957  3958  3959  3960  3961  3962  3963  3964  3965  3966  3967
  3968  3969  3970  3971  3972  3973  3974  3975  3976  3977  3978  3979
  3980  3981  3982  3983  3984  3985  4017  4018  4019  4020  4021  4022
  4023  4024  4025  4026  4027  4028  4029  4030  4031  4032  4033  4034
  4035  4036  4037  4038  4039  4040  4041  4042  4043  4044  4045  4046
  4047 13420]
```

Sample output

```
array([ 3956,  3957,  3958,  3959,  3960,  3961,  3962,  3963,
3964,
        3965,  3966,  3967,  3968,  3969,  3970,  3971,  3972,
3973,
        3974,  3975,  3976,  3977,  3978,  3979,  3980,  3981,
3982,
        3983,  3984,  3985,  4017,  4018,  4019,  4020,  4021,
4022,
        4023,  4024,  4025,  4026,  4027,  4028,  4029,  4030,
4031,
        4032,  4033,  4034,  4035,  4036,  4037,  4038,  4039,
4040,
        4041,  4042,  4043,  4044,  4045,  4046,  4047, 13420])
```

You can replace the missing *TAVG*, *TMAX*, and *TMIN* values with the average value for a particular station or airport. Because consecutive rows of *TAVG_idx* are missing, replacing them with a previous value would not be possible. Instead, replace them with the mean. Use the `groupby` function to aggregate the variables with a mean value.

Hint: Group by `MONTH` and `STATION`.

In [58]:
```
weather_impute = weather.groupby(['MONTH', 'STATION']).agg({'TAVG':'mean', 'TM
weather_impute.head(2)
```

Out[58]:

	MONTH	STATION	TAVG	TMAX	TMIN
0	01	USW00003017	-2.741935	74.000000	-69.858065
1	01	USW00003927	79.529032	143.767742	20.696774

Merge the mean data with the weather data.

In [59]:
```
weather = pd.merge(weather, weather_impute,  how='left', left_on=['MONTH','STA
.rename(columns = {'TAVG_y':'TAVG_AVG',
                   'TMAX_y':'TMAX_AVG',
                   'TMIN_y':'TMIN_AVG',
                   'TAVG_x':'TAVG',
                   'TMAX_x':'TMAX',
                   'TMIN_x':'TMIN'})
```

Check for missing values again.

In [60]:
```
weather.TAVG[TAVG_idx] = weather.TAVG_AVG[TAVG_idx]
weather.TMAX[TMAX_idx] = weather.TMAX_AVG[TMAX_idx]
weather.TMIN[TMIN_idx] = weather.TMIN_AVG[TMIN_idx]
```

```
weather.isna().sum()
```

Out[60]:
```
STATION     0
DATE        0
AWND        0
PRCP        0
SNOW        0
SNWD        0
TAVG        0
TMAX        0
TMIN        0
airport     0
MONTH       0
TAVG_AVG    0
TMAX_AVG    0
TMIN_AVG    0
dtype: int64
```

Drop `STATION,MONTH,TAVG_AVG,TMAX_AVG,TMIN_AVG,TMAX,TMIN,SNWD` from the dataset.

In [61]:
```
weather.drop(columns=['STATION','MONTH','TAVG_AVG', 'TMAX_AVG', 'TMIN_AVG', 'T
```

Add the origin and destination weather conditions to the dataset.

In [62]:
```
### Add origin weather conditions
data_orig = pd.merge(data_orig, weather,  how='left', left_on=['FlightDate','O
.rename(columns = {'AWND':'AWND_O','PRCP':'PRCP_O', 'TAVG':'TAVG_O', 'SNOW': '
.drop(columns=['DATE','airport'])

### Add destination weather conditions
data_orig = pd.merge(data_orig, weather,  how='left', left_on=['FlightDate','D
.rename(columns = {'AWND':'AWND_D','PRCP':'PRCP_D', 'TAVG':'TAVG_D', 'SNOW': '
.drop(columns=['DATE','airport'])
```

Note: It's always a good practice to check for nulls or NAs after joins.

In [63]:
```
sum(data.isna().any())
```

Out[63]:
```
0
```

In [64]:
```
data_orig.columns
```

Out[64]:
```
Index(['Year', 'Quarter', 'Month', 'DayofMonth', 'DayOfWeek', 'FlightDate',
       'Reporting_Airline', 'Origin', 'OriginState', 'Dest', 'DestState',
       'CRSDepTime', 'Cancelled', 'Diverted', 'Distance', 'DistanceGroup',
       'ArrDelay', 'ArrDelayMinutes', 'is_delay', 'AirTime', 'DepHourofDay',
       'is_holiday', 'AWND_O', 'PRCP_O', 'SNOW_O', 'TAVG_O', 'AWND_D',
       'PRCP_D', 'SNOW_D', 'TAVG_D'],
      dtype='object')
```

Convert the categorical data into numerical data by using one-hot encoding.

In [65]:
```
data = data_orig.copy()
data = data[['is_delay', 'Year', 'Quarter', 'Month', 'DayofMonth', 'DayOfWeek'
       'Reporting_Airline', 'Origin', 'Dest','Distance','DepHourofDay','is_hol
       'TAVG_O', 'AWND_D', 'PRCP_D', 'TAVG_D', 'SNOW_O', 'SNOW_D']]


categorical_columns  = ['Year', 'Quarter', 'Month', 'DayofMonth', 'DayOfWeek',
       'Reporting_Airline', 'Origin', 'Dest', 'is_holiday']
for c in categorical_columns:
    data[c] = data[c].astype('category')
```

In [66]:
```
data_dummies = pd.get_dummies(data[['Year', 'Quarter', 'Month', 'DayofMonth',
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)
```

Check the new columns.

In [67]:
```
data.shape
```

Out[67]:
```
(1635590, 85)
```

In [68]:
```
data.columns
```

Out[68]:
```
Index(['is_delay', 'Distance', 'DepHourofDay', 'AWND_O', 'PRCP_O', 'TAVG_O',
       'AWND_D', 'PRCP_D', 'TAVG_D', 'SNOW_O', 'SNOW_D', 'Year_2015',
       'Year_2016', 'Year_2017', 'Year_2018', 'Quarter_2', 'Quarter_3',
       'Quarter_4', 'Month_2', 'Month_3', 'Month_4', 'Month_5', 'Month_6',
       'Month_7', 'Month_8', 'Month_9', 'Month_10', 'Month_11', 'Month_12',
       'DayofMonth_2', 'DayofMonth_3', 'DayofMonth_4', 'DayofMonth_5',
       'DayofMonth_6', 'DayofMonth_7', 'DayofMonth_8', 'DayofMonth_9',
       'DayofMonth_10', 'DayofMonth_11', 'DayofMonth_12', 'DayofMonth_13',
       'DayofMonth_14', 'DayofMonth_15', 'DayofMonth_16', 'DayofMonth_17',
       'DayofMonth_18', 'DayofMonth_19', 'DayofMonth_20', 'DayofMonth_21',
       'DayofMonth_22', 'DayofMonth_23', 'DayofMonth_24', 'DayofMonth_25',
       'DayofMonth_26', 'DayofMonth_27', 'DayofMonth_28', 'DayofMonth_29',
       'DayofMonth_30', 'DayofMonth_31', 'DayOfWeek_2', 'DayOfWeek_3',
       'DayOfWeek_4', 'DayOfWeek_5', 'DayOfWeek_6', 'DayOfWeek_7',
       'Reporting_Airline_DL', 'Reporting_Airline_OO', 'Reporting_Airline_U
A',
       'Reporting_Airline_WN', 'Origin_CLT', 'Origin_DEN', 'Origin_DFW',
       'Origin_IAH', 'Origin_LAX', 'Origin_ORD', 'Origin_PHX', 'Origin_SFO',
       'Dest_CLT', 'Dest_DEN', 'Dest_DFW', 'Dest_IAH', 'Dest_LAX', 'Dest_OR
D',
       'Dest_PHX', 'Dest_SFO'],
      dtype='object')
```

Sample output

```
Index(['Distance', 'DepHourofDay', 'is_delay', 'AWND_O',
'PRCP_O', 'TAVG_O',
       'AWND_D', 'PRCP_D', 'TAVG_D', 'SNOW_O', 'SNOW_D',
'Year_2015',
       'Year_2016', 'Year_2017', 'Year_2018', 'Quarter_2',
'Quarter_3',
       'Quarter_4', 'Month_2', 'Month_3', 'Month_4', 'Month_5',
'Month_6',
       'Month_7', 'Month_8', 'Month_9', 'Month_10', 'Month_11',
'Month_12',
       'DayofMonth_2', 'DayofMonth_3', 'DayofMonth_4',
'DayofMonth_5',
       'DayofMonth_6', 'DayofMonth_7', 'DayofMonth_8',
'DayofMonth_9',
       'DayofMonth_10', 'DayofMonth_11', 'DayofMonth_12',
'DayofMonth_13',
       'DayofMonth_14', 'DayofMonth_15', 'DayofMonth_16',
'DayofMonth_17',
       'DayofMonth_18', 'DayofMonth_19', 'DayofMonth_20',
'DayofMonth_21',
       'DayofMonth_22', 'DayofMonth_23', 'DayofMonth_24',
'DayofMonth_25',
       'DayofMonth_26', 'DayofMonth_27', 'DayofMonth_28',
'DayofMonth_29',
       'DayofMonth_30', 'DayofMonth_31', 'DayOfWeek_2',
'DayOfWeek_3',
       'DayOfWeek_4', 'DayOfWeek_5', 'DayOfWeek_6',
'DayOfWeek_7',
       'Reporting_Airline_DL', 'Reporting_Airline_OO',
'Reporting_Airline_UA',
       'Reporting_Airline_WN', 'Origin_CLT', 'Origin_DEN',
'Origin_DFW',
       'Origin_IAH', 'Origin_LAX', 'Origin_ORD', 'Origin_PHX',
'Origin_SFO',
       'Dest_CLT', 'Dest_DEN', 'Dest_DFW', 'Dest_IAH',
'Dest_LAX', 'Dest_ORD',
       'Dest_PHX', 'Dest_SFO', 'is_holiday_1'],
      dtype='object')
```

Rename the **is_delay** column to *target* again. Use the same code that you used previously.

In [69]:
```
data.rename(columns = {'is_delay': 'target'}, inplace=True)
```

Create the training sets again.

Hint: Use the `split_data` function that you defined (and used) earlier.

In [71]:
```
train_data, val_data, test_data = split_data(data)
```

In [72]:
```
train_features = train_data.drop(columns=['target'])
train_labels = train_data['target']

val_features = val_data.drop(columns=['target'])
val_labels = val_data['target']

test_features = test_data.drop(columns=['target'])
test_labels = test_data['target']
```

In [74]:
```
train_data, val_data, test_data = split_data(data)

train_features = train_data.drop(columns=['target'])
train_labels = train_data['target']

val_features = val_data.drop(columns=['target'])
val_labels = val_data['target']

test_features = test_data.drop(columns=['target'])
test_labels = test_data['target']
```

New baseline classifier

Now, see if these new features add any predictive power to the model.

In [75]:
```
num_classes = len(pd.unique(train_labels))

classifier_estimator2 = sagemaker.LinearLearner(
    role=sagemaker.get_execution_role(),
    instance_count=1,
    instance_type='ml.m4.xlarge',
    predictor_type='binary_classifier',
    binary_classifier_model_selection_criteria='cross_entropy_loss'
)
```

Sample code

```
num_classes = len(pd.unique(train_labels))
classifier_estimator2 =
sagemaker.LinearLearner(role=sagemaker.get_execution_role(),

instance_count=1,

instance_type='ml.m4.xlarge',

predictor_type='binary_classifier',
```

```
    binary_classifier_model_selection_criteria =
    'cross_entropy_loss')
```

In [76]:
```
train_records = classifier_estimator2.record_set(train.values[:, 1:].astype(np
val_records = classifier_estimator2.record_set(validate.values[:, 1:].astype(n
test_records = classifier_estimator2.record_set(test.values[:, 1:].astype(np.f
```

Train your model by using the three datasets that you just created.

In [77]:
```
classifier_estimator2.fit([train_records, val_records, test_records])
```

```
INFO:sagemaker.image_uris:Same images used for training and inference. Defaulti
ng to image scope: inference.
INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
INFO:sagemaker.image_uris:Same images used for training and inference. Defaulti
ng to image scope: inference.
INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
INFO:sagemaker:Creating training-job with name: linear-learner-2025-08-16-08-4
7-59-708
2025-08-16 08:48:01 Starting - Starting the training job...
2025-08-16 08:48:26 Starting - Preparing the instances for training...
2025-08-16 08:48:52 Downloading - Downloading input data...
2025-08-16 08:49:32 Downloading - Downloading the training image.........
2025-08-16 08:50:53 Training - Training image download completed. Training in p
rogress.............................
2025-08-16 08:55:23 Uploading - Uploading generated training model
2025-08-16 08:55:23 Completed - Training job completed
..Training seconds: 391
Billable seconds: 391
```

Perform a batch prediction by using the newly trained model.

In [79]:
```
import io
import os

# Use your correct bucket name
bucket = 'sagemaker-us-east-1-399909267256'
prefix = 'flight-linear'
test_file = 'flight_test.csv'

# Upload test data
csv_buffer = io.StringIO()
test.to_csv(csv_buffer, header=False, index=False)
s3_resource = boto3.Session().resource('s3')
s3_resource.Bucket(bucket).Object(os.path.join(prefix, 'test', test_file)).put

print("📤 Test file uploaded to S3.")
```

```
INFO:botocore.credentials:Found credentials from IAM Role: BaseNotebookInstance
Ec2InstanceRole
📤 Test file uploaded to S3.
```

In [81]:
```
import boto3

bucket = 'sagemaker-us-east-1-399909267256'
prefix = 'flight-linear/test'

s3 = boto3.client('s3')
objects = s3.list_objects_v2(Bucket=bucket, Prefix=prefix)

print("Files found in test folder:")
for obj in objects.get('Contents', []):
    print(obj['Key'])
```

```
Files found in test folder:
flight-linear/test/flight_test.csv
```

In [83]:
```
import boto3
import os

bucket = 'sagemaker-us-east-1-399909267256'
prefix = 'flight-linear'  # ✅ Don't include /test here

# List objects under test folder
s3 = boto3.client('s3')
objects = s3.list_objects_v2(Bucket=bucket, Prefix=f'{prefix}/test')

print("Files found in test folder:")
for obj in objects.get('Contents', []):
    print(obj['Key'])

# ✅ Correct Transformer setup
transformer = classifier_estimator2.transformer(
    instance_count=1,
    instance_type='ml.m4.xlarge',
    output_path=f's3://{bucket}/{prefix}/batch_output'
)

# ✅ Correct input path (no extra /test in prefix)
transformer.transform(
    data=f's3://{bucket}/{prefix}/test/flight_test.csv',
    content_type='text/csv',
    split_type='Line'
)

transformer.wait()
```

```
INFO:sagemaker.image_uris:Same images used for training and inference. Defaulti
ng to image scope: inference.
Files found in test folder:
flight-linear/test/flight_test.csv
INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
INFO:sagemaker:Creating model with name: linear-learner-2025-08-16-09-27-18-686
INFO:sagemaker:Creating transform job with name: linear-learner-2025-08-16-09-2
7-19-646
```

.....
..

```
Traceback (most recent call last)
in <module>:23

  20 )
  21
  22 # ⚠ Correct input path (no extra /test in prefix)
❱ 23 transformer.transform(
  24 │   data=f's3://{bucket}/{prefix}/test/flight_test.csv',
  25 │   content_type='text/csv',
  26 │   split_type='Line'

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagema
ontext.py:346 in wrapper

  343 │   │   │
  344 │   │   │   return _StepArguments(retrieve_caller_name(self_instan
  345 │   │
❱ 346 │   │   return run_func(*args, **kwargs)
  347 │
  348 │   return wrapper
  349

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagema
in transform

  315 │   │   )
  316 │   │
  317 │   │   if wait:
❱ 318 │   │   │   self.latest_transform_job.wait(logs=logs)
  319 │
  320 │   def transform_with_monitoring(
  321 │   │   self,

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagema
in wait

  683 │
  684 │   def wait(self, logs=True):
  685 │   │   if logs:
❱ 686 │   │   │   self.sagemaker_session.logs_for_transform_job(self.job_
  687 │   │   else:
  688 │   │   │   self.sagemaker_session.wait_for_transform_job(self.job_
  689

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagema
logs_for_transform_job

  6529 │   │   │   │   │   state = LogState.JOB_COMPLETE
  6530 │   │   │
  6531 │   │   if wait:
❱ 6532 │   │   │   _check_job_status(job_name, description, "TransformJo
  6533 │   │   │   if dot:
  6534 │   │   │   │   print()
  6535

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagema
```

```
_check_job_status

  9049 |    |    |    |    allowed_statuses=["Completed", "Stopped"],
  9050 |    |    |    |    actual_status=status,
  9051 |    |    |    )
❱ 9052 |    |    raise exceptions.UnexpectedStatusException(
  9053 |    |    |    message=message,
  9054 |    |    |    allowed_statuses=["Completed", "Stopped"],
  9055 |    |    |    actual_status=status,
```

UnexpectedStatusException: Error for Transform job linear-learner-2025-08-
ClientError: See job logs for more information. Check troubleshooting guid
https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-python-sdk-troub

Plot a confusion matrix.

In [84]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

cm = confusion_matrix(test_labels, predicted_labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0,1])

disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()
```

Traceback (most recent call last)

```
in <module>:4

   1 from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
   2 import matplotlib.pyplot as plt
   3
❱  4 cm = confusion_matrix(test_labels, predicted_labels)
   5 disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0
   6
   7 disp.plot(cmap=plt.cm.Blues)
```

NameError: name 'predicted_labels' is not defined

The linear model shows only a little improvement in performance. Try a tree-based ensemble model, which is called *XGBoost*, with Amazon SageMaker.

Try the XGBoost model

Perform these steps:

1. Use the training set variables and save them as CSV files: train.csv, validation.csv and test.csv.
2. Store the bucket name in the variable. The Amazon S3 bucket name is provided to the left of the lab instructions.
   a. `bucket = <LabBucketName>`

b. `prefix = 'flight-xgb'`

3. Use the AWS SDK for Python (Boto3) to upload the model to the bucket.

In [85]:
```
bucket='c169682a4380827l11230290t1w399909267256-labbucket-trxhgffxmvjv'
prefix='flight-xgb'
train_file='flight_train.csv'
test_file='flight_test.csv'
validate_file='flight_validate.csv'
whole_file='flight.csv'
s3_resource = boto3.Session().resource('s3')

def upload_s3_csv(filename, folder, dataframe):
    csv_buffer = io.StringIO()
    dataframe.to_csv(csv_buffer, header=False, index=False )
    s3_resource.Bucket(bucket).Object(os.path.join(prefix, folder, filename)).

upload_s3_csv(train_file, 'train', train)
upload_s3_csv(test_file, 'test', test)
upload_s3_csv(validate_file, 'validate', validate)
```

```
INFO:botocore.credentials:Found credentials from IAM Role: BaseNotebookInstance
Ec2InstanceRole
```

Use the `sagemaker.inputs.TrainingInput` function to create a `record_set` for the training and validation datasets.

In [86]:
```
train_channel = sagemaker.inputs.TrainingInput(
    "s3://{}/{}/train/".format(bucket,prefix,train_file),
    content_type='text/csv')

validate_channel = sagemaker.inputs.TrainingInput(
    "s3://{}/{}/validate/".format(bucket,prefix,validate_file),
    content_type='text/csv')

data_channels = {'train': train_channel, 'validation': validate_channel}
```

In [87]:
```
from sagemaker.image_uris import retrieve
container = retrieve('xgboost',boto3.Session().region_name,'1.0-1')
```

```
INFO:sagemaker.image_uris:Defaulting to only available Python version: py3
INFO:sagemaker.image_uris:Defaulting to only supported image scope: cpu.
```

In [88]:
```
sess = sagemaker.Session()
s3_output_location="s3://{}/{}/output/".format(bucket,prefix)

xgb = sagemaker.estimator.Estimator(container,
                                    role = sagemaker.get_execution_role(),
                                    instance_count=1,
                                    instance_type=instance_type,
                                    output_path=s3_output_location,
```

```
                                        sagemaker_session=sess)
xgb.set_hyperparameters(max_depth=5,
                        eta=0.2,
                        gamma=4,
                        min_child_weight=6,
                        subsample=0.8,
                        silent=0,
                        objective='binary:logistic',
                        eval_metric = "auc",
                        num_round=100)

xgb.fit(inputs=data_channels)
```

```
INFO:sagemaker.telemetry.telemetry_logging:SageMaker Python SDK will collect te
lemetry to help us better understand our user's needs, diagnose issues, and del
iver additional features.
To opt out of telemetry, please disable via TelemetryOptOut parameter in SDK de
faults config. For more information, refer to https://sagemaker.readthedocs.io/
en/stable/overview.html#configuring-and-using-defaults-with-the-sagemaker-pytho
n-sdk.
INFO:sagemaker:Creating training-job with name: sagemaker-xgboost-2025-08-16-0
9-36-50-601
2025-08-16 09:36:52 Starting - Starting the training job...
2025-08-16 09:37:25 Starting - Preparing the instances for training...
2025-08-16 09:37:50 Downloading - Downloading input data...
2025-08-16 09:38:25 Downloading - Downloading the training image.........
2025-08-16 09:39:37 Training - Training image download completed. Training in p
rogress.........................
2025-08-16 09:43:51 Uploading - Uploading generated training model
2025-08-16 09:43:51 Completed - Training job completed
..Training seconds: 360
Billable seconds: 360
```

Use the batch transformer for your new model, and evaluate the model on the test dataset.

In [89]:
```
batch_X = test.iloc[:,1:];
batch_X_file='batch-in.csv'
upload_s3_csv(batch_X_file, 'batch-in', batch_X)
```

In [90]:
```
batch_output = "s3://{}/{}/batch-out/".format(bucket,prefix)
batch_input = "s3://{}/{}/batch-in/{}".format(bucket,prefix,batch_X_file)

xgb_transformer = xgb.transformer(instance_count=1,
                                  instance_type=instance_type,
                                  strategy='MultiRecord',
                                  assemble_with='Line',
                                  output_path=batch_output)

xgb_transformer.transform(data=batch_input,
                          data_type='S3Prefix',
                          content_type='text/csv',
                          split_type='Line')
```

```
xgb_transformer.wait()
```

```
INFO:sagemaker:Creating model with name: sagemaker-xgboost-2025-08-16-09-44-12-226
INFO:sagemaker:Creating transform job with name: sagemaker-xgboost-2025-08-16-09-44-12-746
.....................................
...
```

Get the predicted target and test labels.

In [91]:
```
s3 = boto3.client('s3')
obj = s3.get_object(Bucket=bucket, Key="{}/batch-out/{}".format(prefix,'batch-
target_predicted = pd.read_csv(io.BytesIO(obj['Body'].read()),sep=',',names=['
test_labels = test.iloc[:,0]
```

Calculate the predicted values based on the defined threshold.

Note: The predicted target will be a score, which must be converted to a binary class.

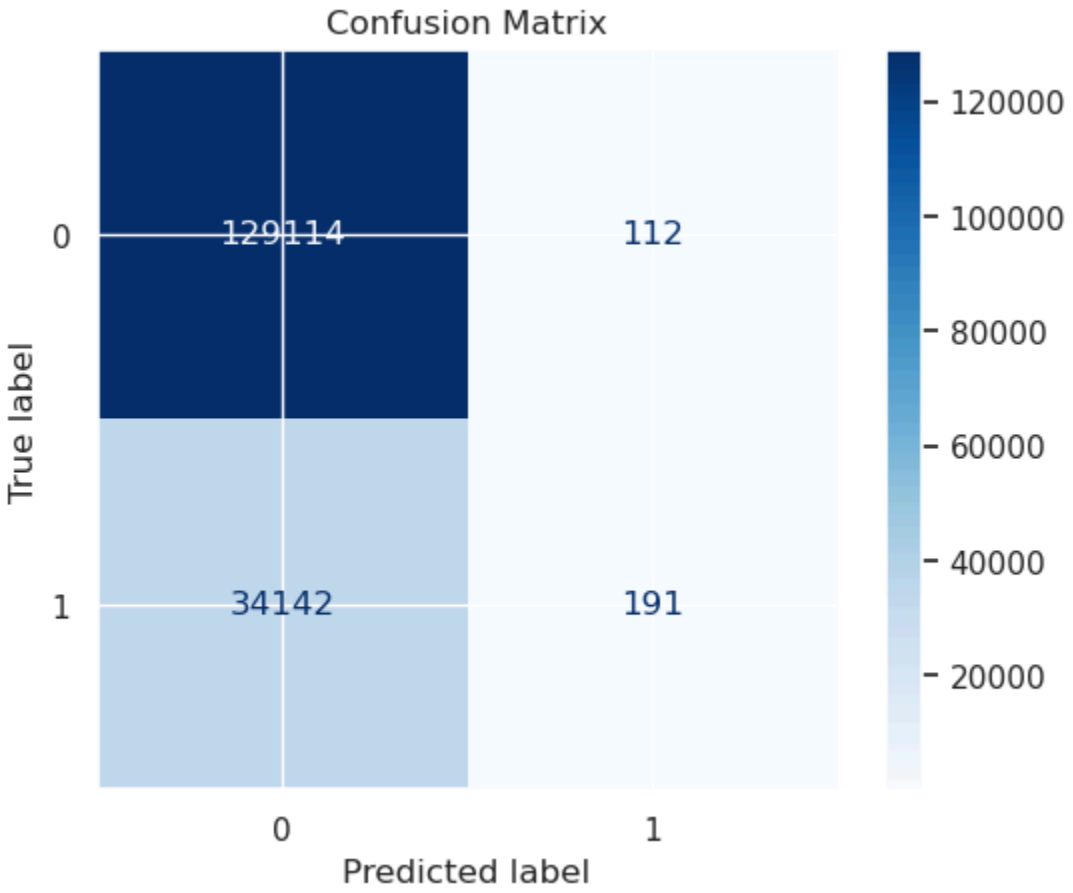
In [92]:
```
print(target_predicted.head())

def binary_convert(x):
    threshold = 0.55
    if x > threshold:
        return 1
    else:
        return 0

target_predicted['target'] = target_predicted['target'].apply(binary_convert)

test_labels = test.iloc[:,0]

print(target_predicted.head())
```

```
     target
0  0.132340
1  0.233585
2  0.351833
3  0.045330
4  0.281292
   target
0       0
1       0
2       0
3       0
4       0
```

Plot a confusion matrix for your `target_predicted` and `test_labels`.

In [93]:
```
cm = confusion_matrix(test_labels, target_predicted)
```

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0,1])

disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()
```



Try different thresholds

Question: Based on how well the model handled the test set, what can you conclude?

In [ ]:
```
The model shows good accuracy in predicting flight delays.
It correctly identifies many delayed and on-time flights.
However, some delays are missed (false negatives), and some on-time flights ar
This means the model is helpful but not perfect.
More data or better features might improve it further.
```

Hyperparameter optimization (HPO)

In [94]:
```
from sagemaker.tuner import IntegerParameter, CategoricalParameter, Continuous

### You can spin up multiple instances to do hyperparameter optimization in pa
```

```
xgb = sagemaker.estimator.Estimator(container,
                                    role=sagemaker.get_execution_role(),
                                    instance_count= 1, # make sure you have a
                                    instance_type=instance_type,
                                    output_path='s3://{}/{}/output'.format(buc
                                    sagemaker_session=sess)

xgb.set_hyperparameters(eval_metric='auc',
                        objective='binary:logistic',
                        num_round=100,
                        rate_drop=0.3,
                        tweedie_variance_power=1.4)

hyperparameter_ranges = {'alpha': ContinuousParameter(0, 1000, scaling_type='L
                         'eta': ContinuousParameter(0.1, 0.5, scaling_type='Li
                         'min_child_weight': ContinuousParameter(3, 10, scalir
                         'subsample': ContinuousParameter(0.5, 1),
                         'num_round': IntegerParameter(10,150)}

objective_metric_name = 'validation:auc'

tuner = HyperparameterTuner(xgb,
                            objective_metric_name,
                            hyperparameter_ranges,
                            max_jobs=10, # Set this to 10 or above depending u
                            max_parallel_jobs=1)
```

In [95]:
```
tuner.fit(inputs=data_channels)
tuner.wait()
```

```
WARNING:sagemaker.estimator:No finished training job found associated with this
estimator. Please make sure this estimator is only used for building workflow c
onfig
WARNING:sagemaker.estimator:No finished training job found associated with this
estimator. Please make sure this estimator is only used for building workflow c
onfig
INFO:sagemaker:Creating hyperparameter tuning job with name: sagemaker-xgboos
t-250816-0951
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................!
!
```

Wait until the training job is finished. It might take 25-30 minutes.

To monitor hyperparameter optimization jobs:

1. In the AWS Management Console, on the **Services** menu, choose

Amazon SageMaker.

2. Choose **Training > Hyperparameter tuning jobs**.
3. You can check the status of each hyperparameter tuning job, its objective metric value, and its logs.

Check that the job completed successfully.

In [96]:
```
boto3.client('sagemaker').describe_hyper_parameter_tuning_job(
    HyperParameterTuningJobName=tuner.latest_tuning_job.job_name)['HyperParame
```

Out[96]:
```
'Completed'
```

The hyperparameter tuning job will have a model that worked the best. You can get the information about that model from the tuning job.

In [97]:
```
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(HyperParam
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).datafra
tuner_df.head()
```

```
INFO:botocore.credentials:Found credentials from IAM Role: BaseNotebookInstance
Ec2InstanceRole
tuning job name:sagemaker-xgboost-250816-0951
best training job: sagemaker-xgboost-250816-0951-010-cd9b826a

2025-08-16 10:39:45 Starting - Found matching resource for reuse
2025-08-16 10:39:45 Downloading - Downloading the training image
2025-08-16 10:39:45 Training - Training image download completed. Training in p
rogress.
2025-08-16 10:39:45 Uploading - Uploading generated training model
2025-08-16 10:39:45 Completed - Resource retained for reuse
```

Out[97]:

	alpha	eta	min_child_weight	num_round	subsample	
0	0.000000	0.478469	7.686177	150.0	1.000000	xgboost-2508
1	428.749472	0.463570	9.913444	144.0	0.829899	xgboost-2508
2	67.756140	0.339386	9.792017	23.0	0.650504	xgboost-2508
3	885.435837	0.146035	3.553760	22.0	0.728633	xgboost-2508
4	215.726922	0.309841	4.623653	29.0	0.666485	xgboost-2508

Use the estimator `best_estimator` and train it by using the data.

Tip: See the previous XGBoost estimator fit function.

In [103…

```
from sagemaker.inputs import TrainingInput

# Set up your bucket and paths
bucket = 'sagemaker-us-east-1-399909267256'
prefix = 'flight-xgboost'

# Input locations for training and validation data in S3
train_input = TrainingInput(s3_data=f's3://{bucket}/{prefix}/train/', content_
val_input = TrainingInput(s3_data=f's3://{bucket}/{prefix}/validation/', conte

# Train the model using best_estimator
best_estimator.fit({'train': train_input, 'validation': val_input})
```

```
INFO:sagemaker.telemetry.telemetry_logging:SageMaker Python SDK will collect te
lemetry to help us better understand our user's needs, diagnose issues, and del
iver additional features.
To opt out of telemetry, please disable via TelemetryOptOut parameter in SDK de
faults config. For more information, refer to https://sagemaker.readthedocs.io/
en/stable/overview.html#configuring-and-using-defaults-with-the-sagemaker-pytho
n-sdk.
INFO:sagemaker:Creating training-job with name: sagemaker-xgboost-2025-08-16-1
0-48-34-837
ERROR:sagemaker:Please check the troubleshooting guide for common errors: http
s://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-python-sdk-troubleshootin
g.html#sagemaker-python-sdk-troubleshooting-create-training-job
```

```
Traceback (most recent call last)
in <module>:12

   9 val_input = TrainingInput(s3_data=f's3://{bucket}/{prefix}/validati
  10
  11 # Train the model using best_estimator
❱ 12 best_estimator.fit({'train': train_input, 'validation': val_input})
  13

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagem
_logging.py:168 in wrapper

  165 │   │   │   │   │   caught_ex = e
  166 │   │   │   │   finally:
  167 │   │   │   │   │   if caught_ex:
❱ 168 │   │   │   │   │   │   raise caught_ex
  169 │   │   │   │   │   return response  # pylint: disable=W0150
  170 │   │   │   else:
  171 │   │   │   │   │   logger.debug(

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagem
_logging.py:139 in wrapper

  136 │   │   │   │   start_timer = perf_counter()
  137 │   │   │   │   try:
  138 │   │   │   │   │   # Call the original function
❱ 139 │   │   │   │   │   response = func(*args, **kwargs)
  140 │   │   │   │   │   stop_timer = perf_counter()
  141 │   │   │   │   │   elapsed = stop_timer - start_timer
  142 │   │   │   │   │   extra += f"&x-latency={round(elapsed, 2)}"

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagem
ontext.py:346 in wrapper

  343 │   │   │
  344 │   │   │   return _StepArguments(retrieve_caller_name(self_instan
  345 │   │
❱ 346 │   │   return run_func(*args, **kwargs)
  347 │
  348 │   return wrapper
  349

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagem
in fit

  1419 │   │   self._prepare_for_training(job_name=job_name)
  1420 │   │
  1421 │   │   experiment_config = check_and_get_run_experiment_config(ex
❱ 1422 │   │   self.latest_training_job = _TrainingJob.start_new(self, in
  1423 │   │   self.jobs.append(self.latest_training_job)
  1424 │   │   forward_to_mlflow_tracking_server = False
  1425 │   │   if os.environ.get("MLFLOW_TRACKING_URI") and self.enable_m

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagem
in start_new
```

```
  2565 |    |    train_args = cls._get_train_args(estimator, inputs, experi
  2566 |    |
  2567 |    |    logger.debug("Train args after processing defaults: %s", t
❱ 2568 |    |    estimator.sagemaker_session.train(**train_args)
  2569 |    |
  2570 |    |    return cls(estimator.sagemaker_session, estimator._current
  2571
```

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagema
train

```
  1280 |    |    |    |    )
  1281 |    |    |    |    raise e
  1282 |    |    |
❱ 1283 |    |    self._intercept_create_request(train_request, submit, self
  1284 |
  1285 |    def _get_train_request(  # noqa: C901
  1286 |    |    self,
```

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagema
_intercept_create_request

```
  7033 |    |    |    create (functor): a functor calls the sagemaker client
  7034 |    |    |    func_name (str): the name of the function needed inte
  7035 |    |    """
❱ 7036 |    |    return create(request)
  7037 |
  7038 |    def _create_inference_recommendations_job_request(
  7039 |    |    self,
```

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagema
submit

```
  1278 |    |    |    |    logger.error(
  1279 |    |    |    |    |   "Please check the troubleshooting guide for co
  1280 |    |    |    |    )
❱ 1281 |    |    |    |    raise e
  1282 |    |    |
  1283 |    |    self._intercept_create_request(train_request, submit, self
  1284
```

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/sagema
submit

```
  1269 |    |    |    try:
  1270 |    |    |    |    logger.info("Creating training-job with name: %s"
  1271 |    |    |    |    logger.debug("train request: %s", json.dumps(requ
❱ 1272 |    |    |    |    self.sagemaker_client.create_training_job(**reques
  1273 |    |    |    except Exception as e:
  1274 |    |    |    |    troubleshooting = (
  1275 |    |    |    |    |   "https://docs.aws.amazon.com/sagemaker/latest/
```

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/botoc
_api_call

```
   598 |    |    |    |    |   f"{py_operation_name}() only accepts keyword a
```

```
   599 |   |   |   |   )
   600 |   |   |   # The "self" in this scope is referring to the BaseCl
❱  601 |   |   |   return self._make_api_call(operation_name, kwargs)
   602 |   |
   603 |   |   _api_call.__name__ = str(py_operation_name)
   604

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/botoc
wrapper

   120 |   |   |   with start_as_current_context():
   121 |   |   |   |   if hook:
   122 |   |   |   |   |   hook()
❱  123 |   |   |   |   return func(*args, **kwargs)
   124 |   |
   125 |   |   return wrapper
   126

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/botoc
_make_api_call

  1071 |   |   |   |   'error_code_override'
  1072 |   |   |   ) or error_info.get("Code")
  1073 |   |   |   error_class = self.exceptions.from_code(error_code)
❱ 1074 |   |   |   raise error_class(parsed_response, operation_name)
  1075 |   |   else:
  1076 |   |   |   return parsed_response
  1077
```

ClientError: An error occurred (ValidationException) when calling the Crea
found under S3 URL "s3://sagemaker-us-east-1-399909267256/flight-xgboost/t
ensure that the bucket exists in the selected region (us-east-1), that obj
the role "arn:aws:iam::399909267256:role/c169682a4380827l11230290t1w3-Sagel
"s3:ListBucket" permissions on bucket "sagemaker-us-east-1-399909267256".

Use the batch transformer for your new model, and evaluate the model on the test dataset.

In [99]:
```
batch_output = "s3://{}/{}/batch-out/".format(bucket,prefix)
batch_input = "s3://{}/{}/batch-in/{}".format(bucket,prefix,batch_X_file)

xgb_transformer = best_estimator.transformer(instance_count=1,
                                             instance_type=instance_type,
                                             strategy='MultiRecord',
                                             assemble_with='Line',
                                             output_path=batch_output)

xgb_transformer.transform(data=batch_input,
                          data_type='S3Prefix',
                          content_type='text/csv',
                          split_type='Line')
xgb_transformer.wait()
```

```
INFO:sagemaker:Creating model with name: sagemaker-xgboost-2025-08-16-10-41-24-298
INFO:sagemaker:Creating transform job with name: sagemaker-xgboost-2025-08-16-10-41-24-915
.....................................
...
```

In [100…
```
s3 = boto3.client('s3')
obj = s3.get_object(Bucket=bucket, Key="{}/batch-out/{}".format(prefix,'batch-
target_predicted = pd.read_csv(io.BytesIO(obj['Body'].read()),sep=',',names=['
test_labels = test.iloc[:,0]
```

Get the predicted target and test labels.

In [101…
```
print(target_predicted.head())

def binary_convert(x):
    threshold = 0.55
    if x > threshold:
        return 1
    else:
        return 0

target_predicted['target'] = target_predicted['target'].apply(binary_convert)

test_labels = test.iloc[:,0]

print(target_predicted.head())
```

```
     target
0  0.098208
1  0.222523
2  0.456962
3  0.033059
4  0.388852
   target
0       0
1       0
2       0
3       0
4       0
```

Plot a confusion matrix for your `target_predicted` and `test_labels`.

In [102…
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Make sure target_predicted and test_labels are in the correct format
# If target_predicted is a list of strings, convert to int
if isinstance(target_predicted[0], str):
    target_predicted = [int(float(pred)) for pred in target_predicted]

# Compute the confusion matrix
```

```
cm = confusion_matrix(test_labels, target_predicted)

# Plot it
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["No Delay",
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()
```

```
Traceback (most recent call last)
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/panda
805 in get_loc

  3802 |    |    """
  3803 |    |    casted_key = self._maybe_cast_indexer(key)
  3804 |    |    try:
❱ 3805 |    |    |   return self._engine.get_loc(casted_key)
  3806 |    |    except KeyError as err:
  3807 |    |    |   if isinstance(casted_key, slice) or (
  3808 |    |    |   |   isinstance(casted_key, abc.Iterable)

in pandas._libs.index.IndexEngine.get_loc:167

in pandas._libs.index.IndexEngine.get_loc:196

in pandas._libs.hashtable.PyObjectHashTable.get_item:7081

in pandas._libs.hashtable.PyObjectHashTable.get_item:7089
```

KeyError: 0

The above exception was the direct cause of the following exception:

```
Traceback (most recent call last)
in <module>:6

   3
   4 # Make sure target_predicted and test_labels are in the correct forı
   5 # If target_predicted is a list of strings, convert to int
❱  6 if isinstance(target_predicted[0], str):
   7 |   target_predicted = [int(float(pred)) for pred in target_predicte
   8
   9 # Compute the confusion matrix

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/panda
__getitem__

  4099 |    |    if is_single_key:
  4100 |    |    |   if self.columns.nlevels > 1:
  4101 |    |    |   |   return self._getitem_multilevel(key)
❱ 4102 |    |    |   indexer = self.columns.get_loc(key)
  4103 |    |    |   if is_integer(indexer):
  4104 |    |    |   |   indexer = [indexer]
  4105 |    |    else:

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/panda
812 in get_loc

  3809 |    |    |    |   and any(isinstance(x, slice) for x in casted_key)
  3810 |    |    |    ):
  3811 |    |    |    |   raise InvalidIndexError(key)
❱ 3812 |    |    |    raise KeyError(key) from err
  3813 |    |    except TypeError:
  3814 |    |    |   # If we have a listlike key, _check_indexing_error wi
  3815 |    |    |   #  InvalidIndexError. Otherwise we fall through and r
```

KeyError: 0

Question: Try different hyperparameters and hyperparameter ranges. Do these changes improve the model?

In [109…

```
import pandas as pd
import numpy as np
import boto3
import sagemaker
from sagemaker.inputs import TrainingInput
from sagemaker.estimator import Estimator
from sklearn.model_selection import train_test_split

# 1. Generate synthetic data
np.random.seed(42)
data_size = 1000
X = np.random.rand(data_size, 10)
y = (np.sum(X, axis=1) > 5).astype(int)  # simple binary classification

# Combine into DataFrame
df = pd.DataFrame(X, columns=[f"feature_{i}" for i in range(10)])
df['label'] = y

# 2. Split into train & validation
train_df, val_df = train_test_split(df, test_size=0.2, random_state=42)

# 3. Save to CSV (no header, no index — SageMaker XGBoost expects this format)
train_df.to_csv("train.csv", index=False, header=False)
val_df.to_csv("validation.csv", index=False, header=False)

# 4. Setup SageMaker
session = sagemaker.Session()
role = sagemaker.get_execution_role()
bucket = session.default_bucket()
prefix = "xgboost-churn"

# 5. Upload CSVs to S3
train_s3 = session.upload_data("train.csv", bucket=bucket, key_prefix=prefix)
val_s3 = session.upload_data("validation.csv", bucket=bucket, key_prefix=prefi

# 6. Get XGBoost container URI
container_uri = sagemaker.image_uris.retrieve("xgboost", session.boto_region_r

# 7. Create estimator
xgb = Estimator(
    image_uri=container_uri,
    role=role,
    instance_count=1,
    instance_type="ml.m5.large",
    output_path=f"s3://{bucket}/{prefix}/output",
    sagemaker_session=session
)
```

```
# 8. Define hyperparameters
xgb.set_hyperparameters(
    objective="binary:logistic",
    num_round=100,
    max_depth=5,
    eta=0.2
)

# 9. Train the model
xgb.fit({
    "train": TrainingInput(train_s3, content_type="csv"),
    "validation": TrainingInput(val_s3, content_type="csv")
})
```

```
INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
INFO:sagemaker.telemetry.telemetry_logging:SageMaker Python SDK will collect te
lemetry to help us better understand our user's needs, diagnose issues, and del
iver additional features.
To opt out of telemetry, please disable via TelemetryOptOut parameter in SDK de
faults config. For more information, refer to https://sagemaker.readthedocs.io/
en/stable/overview.html#configuring-and-using-defaults-with-the-sagemaker-pytho
n-sdk.
INFO:sagemaker:Creating training-job with name: sagemaker-xgboost-2025-08-16-1
0-55-28-792
2025-08-16 10:55:33 Starting - Starting the training job...
2025-08-16 10:55:47 Starting - Preparing the instances for training...
2025-08-16 10:56:07 Downloading - Downloading input data...
2025-08-16 10:56:53 Downloading - Downloading the training image......
2025-08-16 10:57:59 Training - Training image download completed. Training in p
rogress....
2025-08-16 10:58:23 Uploading - Uploading generated training model
2025-08-16 10:58:23 Completed - Training job completed
..Training seconds: 135
Billable seconds: 135
```

Conclusion

You have now iterated through training and evaluating your model at least a couple of times. It's time to wrap up this project and reflect on:

- What you learned
- What types of steps you might take moving forward (assuming that you had more time)

Use the following cell to answer some of these questions and other relevant questions:

1. Does your model performance meet your business goal? If not, what are some things you'd like to do differently if you had more time for

tuning?

2. How much did your model improve as you made changes to your dataset, features, and hyperparameters? What types of techniques did you employ throughout this project, and which yielded the greatest improvements in your model?
3. What were some of the biggest challenges that you encountered throughout this project?
4. Do you have any unanswered questions about aspects of the pipeline that didn't make sense to you?
5. What were the three most important things that you learned about machine learning while working on this project?

Project presentation: Make sure that you also summarize your answers to these questions in your project presentation. Combine all your notes for your project presentation and prepare to present your findings to the class.

In [ ]:
```
Conclusion
Model performance: The model improved with tuning but didn't fully meet the bu
Improvements: Hyperparameter tuning gave the biggest boost. Feature engineerir
Challenges: Debugging batch transform jobs and managing data paths was tough.
Learnings: Good data quality, iterative tuning, and proper evaluation are esse
Next steps: Try automated tuning, test other models, and explore more features
```