E0123047

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_ID2=0.

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 [ ]:
```
# Write your answer here
# Write your answer here
We need to predict whether a flight will be delayed due to weather before
That's a supervised classification task where historical labeled data exi
Rule-based logic won't generalize; ML will capture complex patterns from
```

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

In [ ]:
```
# Write your answer here
•       Problem: Warn customers at booking time if a selected flight is l
•       Success metrics (primary → secondary): Recall/TPR and AUC (catch
•       ML output: A probability of delay (0–1) plus a binary label after
```

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

In [ ]:
```
# Write your answer here
Binary classification
```

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

In [ ]:
```
# Write your answer here
Dataset covers US on-time performance (2013–2018) with date/time, airline
status → exactly the signals we need; class imbalance is expected and mus
```

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 [25]:
```
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
```

```
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 [26]:
```
# 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_proj
```

```
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014
_10.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2014_10.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014
_1.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2014_1.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014
_11.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2014_11.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014
_12.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2014_12.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014
_3.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2014_3.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014
_2.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2014_2.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014
_4.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2014_4.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014
_5.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2014_5.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014
_9.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2014_9.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014
_6.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2014_6.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015
_1.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2015_1.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014
_8.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2014_8.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014
_7.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2014_7.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015
_11.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2015_11.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015
_2.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2015_2.zip
```

```
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015
_12.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2015_12.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015
_3.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2015_3.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015
_10.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2015_10.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015
_4.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2015_4.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015
_5.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2015_5.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015
_7.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2015_7.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015
_9.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2015_9.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015
_6.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2015_6.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016
_1.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2016_1.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015
_8.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2015_8.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016
_2.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2016_2.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016
_12.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2016_12.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016
_11.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2016_11.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016
_3.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2016_3.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016
_10.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2016_10.zip
```

```
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016
_4.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2016_4.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016
_6.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2016_6.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016
_5.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2016_5.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016
_7.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2016_7.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017
_1.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2017_1.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016
_8.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2016_8.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016
_9.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2016_9.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017
_10.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2017_10.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017
_11.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2017_11.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017
_12.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2017_12.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017
_2.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2017_2.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017
_3.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2017_3.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017
_4.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2017_4.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017
_5.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2017_5.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017
_7.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2017_7.zip
```

```
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017
_6.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2017_6.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017
_9.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2017_9.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017
_8.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2017_8.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018
_10.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2018_10.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018
_2.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2018_2.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018
_4.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2018_4.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018
_5.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2018_5.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018
_8.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2018_8.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018
_6.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2018_6.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018
_7.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2018_7.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018
_9.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2018_9.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018
_12.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2018_12.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018
_1.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2018_1.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018
_3.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time
_Performance_1987_present_2018_3.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_proj
ect/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018
_11.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Tim
e_Performance_1987_present_2018_11.zip
```

In [27]:
```
zip_files = [str(file) for file in list(Path(base_path).iterdir()) if '.z
len(zip_files)
```

Out[27]:
```
60
```

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

In [28]:
```
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_Re
porting_Carrier_On_Time_Performance_1987_present_2016_3.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2016_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2014_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2015_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2016_9.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2014_9.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2016_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2014_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2015_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2018_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2016_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2016_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2015_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2014_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2014_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2017_3.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_3.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2018_12.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2015_3.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_3.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2016_7.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2014_12.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2018_5.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2018_2.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2017_6.zip
```

```
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2015_1.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2016_12.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2016_6.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2017_12.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2014_5.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2018_7.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2018_11.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2017_4.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2015_5.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2017_9.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_9.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2015_12.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_12.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2018_4.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2015_6.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2017_1.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2017_8.zip
```

```
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2017_11.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2015_11.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_11.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2018_1.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2017_5.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_5.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2014_4.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2017_7.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2014_10.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2018_3.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_3.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2018_6.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_6.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2015_8.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_8.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2016_4.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_4.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2018_9.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_9.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2015_9.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_9.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2014_7.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_7.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2017_10.zip
```

```
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2018_10.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2018_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2014_1.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_1.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2015_10.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2015_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2016_2.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2016_10.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2016_10.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2017_2.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2017_2.zip
Extracting /home/ec2-user/SageMaker/project/data/FlightDelays/On_Time_Re
porting_Carrier_On_Time_Performance_1987_present_2014_3.zip
zip2csv failed for /home/ec2-user/SageMaker/project/data/FlightDelays/On
_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_3.zip
Files Extracted
```

In [29]:
```
csv_files = [str(file) for file in list(Path(csv_base_path).iterdir()) if
len(csv_files)
```

Out[29]:
```
50
```

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 [30]:
```
from IPython.display import IFrame

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

Out[30]:

In [44]:
```
!df -h
!du -sh ~/* 2>/dev/null | sort -h
!du -sh ~/SageMaker/* 2>/dev/null | sort -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        7.8G     0  7.8G   0% /dev
tmpfs           7.9G     0  7.9G   0% /dev/shm
tmpfs           7.9G  608K  7.9G   1% /run
tmpfs           7.9G     0  7.9G   0% /sys/fs/cgroup
/dev/xvda1      135G   79G   57G  59% /
/dev/xvdf       4.8G  4.6G     0 100% /home/ec2-user/SageMaker
tmpfs           1.6G     0  1.6G   0% /run/user/1001
tmpfs           1.6G     0  1.6G   0% /run/user/1000
tmpfs           1.6G     0  1.6G   0% /run/user/1002
0       /home/ec2-user/sample-notebooks
4.0K    /home/ec2-user/THIRD_PARTY_SOURCE_CODE_URLS
12K     /home/ec2-user/tools
20K     /home/ec2-user/OSSNvidiaDriver_v550.163.01_license.txt
28K     /home/ec2-user/LICENSE
36K     /home/ec2-user/src
40K     /home/ec2-user/LINUX_PACKAGES_LIST
44K     /home/ec2-user/BUILD_FROM_SOURCE_PACKAGES_LICENCES
148K    /home/ec2-user/nvidia-acknowledgements
280K    /home/ec2-user/examples
592K    /home/ec2-user/tutorials
1.6M    /home/ec2-user/PYTHON_PACKAGES_LICENSES
2.8M    /home/ec2-user/Nvidia_Cloud_EULA.pdf
5.0M    /home/ec2-user/LINUX_PACKAGES_LICENSES
707M    /home/ec2-user/sample-notebooks-1755269441
4.6G    /home/ec2-user/SageMaker
27G     /home/ec2-user/anaconda3
16K     /home/ec2-user/SageMaker/lost+found
128K    /home/ec2-user/SageMaker/id_id
132K    /home/ec2-user/SageMaker/ko_kr
220K    /home/ec2-user/SageMaker/es_es
228K    /home/ec2-user/SageMaker/pt_br
260K    /home/ec2-user/SageMaker/zh_cn
272K    /home/ec2-user/SageMaker/ja_jp
340K    /home/ec2-user/SageMaker/en_us
4.6G    /home/ec2-user/SageMaker/project
```

In [45]:
```
!find ~ -name ".ipynb_checkpoints" -type d -exec rm -rf {} + 2>/dev/null
!rm -rf ~/.cache/pip ~/.cache/matplotlib ~/.cache 2>/dev/null || true
!pip cache purge -q
!conda clean -a -y -q
```

```
WARNING: No matching packages
```

In [46]:
```
!find ~/SageMaker -type f -size +500M -exec ls -lh {} \; | sort -k5 -h
```

```
find: ‘/home/ec2-user/SageMaker/lost+found’: Permission denied
```

In [47]:
```
# ⚠️ Replace the path below with one you found in the previous step
# Example:
# !rm -f "/home/ec2-user/SageMaker/some/huge-old-dataset.parquet"
# !rm -rf "/home/ec2-user/SageMaker/old-training-artifacts/"
```

In [48]:
```
!df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        7.8G     0  7.8G   0% /dev
tmpfs           7.9G     0  7.9G   0% /dev/shm
tmpfs           7.9G  608K  7.9G   1% /run
tmpfs           7.9G     0  7.9G   0% /sys/fs/cgroup
/dev/xvda1      135G   75G   61G  55% /
/dev/xvdf       4.8G  4.6G   88K 100% /home/ec2-user/SageMaker
tmpfs           1.6G     0  1.6G   0% /run/user/1001
tmpfs           1.6G     0  1.6G   0% /run/user/1000
tmpfs           1.6G     0  1.6G   0% /run/user/1002
```

In [49]:
```
!ls -lh "/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Re
!rm -f "/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Rep
```

```
-rw-rw-r-- 1 ec2-user ec2-user 0 Aug 15 17:48 /home/ec2-user/SageMaker/p
roject/data/csvFlightDelays/On_Time_Reporting_Carrier_On_Time_Performanc
e_(1987_present)_2018_9.csv
```

In [50]:
```
# Replace <YOUR-BUCKET> and <PATH> with the real S3 info if needed
!aws s3 cp "s3://<YOUR-BUCKET>/<PATH>/On_Time_Reporting_Carrier_On_Time_P
           "/home/ec2-user/SageMaker/project/data/csvFlightDelays/" --no-
!ls -lh "/home/ec2-user/SageMaker/project/data/csvFlightDelays/"
```

```
fatal error: Parameter validation failed:
Invalid bucket name "<YOUR-BUCKET>": Bucket name must match the regex "^
[a-zA-Z0-9.\-_]{1,255}$" or be an ARN matching the regex "^arn:(aws).*:
(s3|s3-object-lambda):[a-z\-0-9]*:[0-9]{12}:accesspoint[/:][a-zA-Z0-9
\-.]{1,63}$|^arn:(aws).*:s3-outposts:[a-z\-0-9]+:[0-9]{12}:outpost[/:][a
-zA-Z0-9\-]{1,63}[/:]accesspoint[/:][a-zA-Z0-9\-]{1,63}$"
total 3.1G
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_10.csv
-rw-rw-r-- 1 ec2-user ec2-user 200M Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_11.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_12.csv
-rw-rw-r-- 1 ec2-user ec2-user 185M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_2.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_4.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_5.csv
-rw-rw-r-- 1 ec2-user ec2-user 217M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_6.csv
-rw-rw-r-- 1 ec2-user ec2-user 219M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_8.csv
-rw-rw-r-- 1 ec2-user ec2-user 202M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_9.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2015_11.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2015_12.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2015_1.csv
-rw-rw-r-- 1 ec2-user ec2-user 184M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2015_2.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2015_3.csv
-rw-rw-r-- 1 ec2-user ec2-user 209M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2015_4.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2015_5.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2015_6.csv
-rw-rw-r-- 1 ec2-user ec2-user 225M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2015_7.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2015_8.csv
-rw-rw-r-- 1 ec2-user ec2-user 195M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_11.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_12.csv
-rw-rw-r-- 1 ec2-user ec2-user 192M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_1.csv
-rw-rw-r-- 1 ec2-user ec2-user 206M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_3.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_4.csv
-rw-rw-r-- 1 ec2-user ec2-user 207M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_5.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_6.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
```

```
er_On_Time_Performance_(1987_present)_2016_7.csv
-rw-rw-r-- 1 ec2-user ec2-user 215M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_8.csv
-rw-rw-r-- 1 ec2-user ec2-user 196M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_9.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2017_11.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2017_12.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2017_1.csv
-rw-rw-r-- 1 ec2-user ec2-user  32M Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2017_3.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2017_4.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2017_5.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2017_6.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2017_7.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2017_8.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2017_9.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2018_11.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2018_12.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2018_1.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2018_2.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2018_3.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2018_4.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2018_5.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2018_6.csv
-rw-rw-r-- 1 ec2-user ec2-user    0 Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2018_7.csv
-rw-rw-r-- 1 ec2-user ec2-user 278M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2018_8.csv
-rw-rw-r-- 1 ec2-user ec2-user  12K Aug 15 17:48 readme.html
```

In [54]:

```
import os

csv_dir = "/home/ec2-user/SageMaker/project/data/csvFlightDelays"
for root, dirs, files in os.walk(csv_dir):
    for f in files:
        print(os.path.join(root, f))
```

```
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_3.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_4.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2014_12.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_1.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_4.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_12.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_3.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_3.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_8.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2014_8.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_2.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_7.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2014_5.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_6.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_6.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2014_10.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_4.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_12.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_11.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_6.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_12.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_8.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_1.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_5.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_1.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_8.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2014_4.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_7.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2014_9.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/readme.html
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
```

```
Carrier_On_Time_Performance_(1987_present)_2015_2.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_7.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_5.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_9.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_6.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_9.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_11.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_11.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_5.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_5.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_1.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_7.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_3.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2014_6.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2014_11.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_4.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2014_2.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_11.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_8.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_12.csv
```

In [57]:
```
!df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        7.8G     0  7.8G   0% /dev
tmpfs           7.9G     0  7.9G   0% /dev/shm
tmpfs           7.9G  608K  7.9G   1% /run
tmpfs           7.9G     0  7.9G   0% /sys/fs/cgroup
/dev/xvda1      135G   75G   61G  55% /
/dev/xvdf       4.8G  4.6G     0 100% /home/ec2-user/SageMaker
tmpfs           1.6G     0  1.6G   0% /run/user/1001
tmpfs           1.6G     0  1.6G   0% /run/user/1000
tmpfs           1.6G     0  1.6G   0% /run/user/1002
```

In [58]:
```
!find ~ -name ".ipynb_checkpoints" -type d -exec rm -rf {} + 2>/dev/null
!rm -rf ~/.cache/pip ~/.cache/matplotlib ~/.cache 2>/dev/null || true
!pip cache purge -q
!conda clean -a -y -q
```

```
WARNING: No matching packages
```

In [59]:
```
!find ~/SageMaker -type f -size +500M -exec ls -lh {} \; | sort -k5 -h
# Example delete (EDIT the path from the list above before running):
# !rm -f "/home/ec2-user/SageMaker/huge-old-file.bin"
```

```
find: ‘/home/ec2-user/SageMaker/lost+found’: Permission denied
```

In [60]:
```
!find "/home/ec2-user/SageMaker/project/data/csvFlightDelays" -type f -si
```

```
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_4.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2014_12.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_4.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_12.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_3.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_3.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_8.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_2.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_7.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2014_5.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2016_6.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_6.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2014_10.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_12.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_11.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_6.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_12.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_8.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_1.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_5.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_1.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2014_4.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_7.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_5.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_9.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_6.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2015_11.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_5.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_1.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_7.csv
```

```
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_4.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2018_11.csv
/home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_Reporting_
Carrier_On_Time_Performance_(1987_present)_2017_12.csv
```

In [61]:
```
# Replace with your real bucket/key or your actual download source
!aws s3 cp "s3://<YOUR-BUCKET>/<PATH>/On_Time_Reporting_Carrier_On_Time_P
          "/home/ec2-user/SageMaker/project/data/csvFlightDelays/" --no-
!ls -lh "/home/ec2-user/SageMaker/project/data/csvFlightDelays/"
```

```
fatal error: Parameter validation failed:
Invalid bucket name "<YOUR-BUCKET>": Bucket name must match the regex "^
[a-zA-Z0-9.\-_]{1,255}$" or be an ARN matching the regex "^arn:(aws).*:
(s3|s3-object-lambda):[a-z\-0-9]*:[0-9]{12}:accesspoint[/:][a-zA-Z0-9
\-.]{1,63}$|^arn:(aws).*:s3-outposts:[a-z\-0-9]+:[0-9]{12}:outpost[/:][a
-zA-Z0-9\-]{1,63}[/:]accesspoint[/:][a-zA-Z0-9\-]{1,63}$"
total 3.1G
-rw-rw-r-- 1 ec2-user ec2-user 200M Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_11.csv
-rw-rw-r-- 1 ec2-user ec2-user 185M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_2.csv
-rw-rw-r-- 1 ec2-user ec2-user 217M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_6.csv
-rw-rw-r-- 1 ec2-user ec2-user 219M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_8.csv
-rw-rw-r-- 1 ec2-user ec2-user 202M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2014_9.csv
-rw-rw-r-- 1 ec2-user ec2-user 184M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2015_2.csv
-rw-rw-r-- 1 ec2-user ec2-user 209M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2015_4.csv
-rw-rw-r-- 1 ec2-user ec2-user 225M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2015_7.csv
-rw-rw-r-- 1 ec2-user ec2-user 195M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_11.csv
-rw-rw-r-- 1 ec2-user ec2-user 192M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_1.csv
-rw-rw-r-- 1 ec2-user ec2-user 206M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_3.csv
-rw-rw-r-- 1 ec2-user ec2-user 207M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_5.csv
-rw-rw-r-- 1 ec2-user ec2-user 215M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_8.csv
-rw-rw-r-- 1 ec2-user ec2-user 196M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2016_9.csv
-rw-rw-r-- 1 ec2-user ec2-user  32M Aug 15 17:48 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2017_3.csv
-rw-rw-r-- 1 ec2-user ec2-user 278M Aug 15 17:47 On_Time_Reporting_Carri
er_On_Time_Performance_(1987_present)_2018_8.csv
-rw-rw-r-- 1 ec2-user ec2-user  12K Aug 15 17:48 readme.html
```

In [62]:
```
import os, glob, re, zipfile
import pandas as pd

csv_dir = "/home/ec2-user/SageMaker/project/data/csvFlightDelays"  # <- a

def load_first_nonempty(csv_dir):
    all_files = glob.glob(os.path.join(csv_dir, "**", "*"), recursive=Tru
```

```
    data_files = [p for p in all_files if p.lower().endswith((".csv",".cs

    # filter out zero-byte files
    nonempty = [p for p in data_files if os.path.getsize(p) > 0]
    if not nonempty:
        raise FileNotFoundError(
            "All candidate CSVs are 0 bytes (corrupted). Free some disk,
        )

    # try to choose a "latest-looking" file by year/month in the name, el
    def ym_key(p):
        b = os.path.basename(p)
        y = re.search(r"(19|20)\d{2}", b)
        m = re.search(r"[^0-9](\d{1,2})[^0-9]", b)
        return (int(y.group(0)) if y else -1, int(m.group(1)) if m else -

    path = sorted(nonempty, key=ym_key)[-1]   # last = most recent
    print(f"✅ Using: {path}  (size: {os.path.getsize(path)} bytes)")

    if path.lower().endswith(".zip"):
        with zipfile.ZipFile(path) as zf:
            inner = [n for n in zf.namelist() if n.lower().endswith(".csv
            if not inner:
                raise ValueError("Zip contains no CSV.")
            with zf.open(inner[0]) as fh:
                df = pd.read_csv(fh, low_memory=False)
    else:
        df = pd.read_csv(path, compression="infer", low_memory=False)

    return df, path

df_temp, used_path = load_first_nonempty(csv_dir)
print("🎉 Loaded:", df_temp.shape)
df_temp.head()
```

```
✅ Using: /home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_
Reporting_Carrier_On_Time_Performance_(1987_present)_2014_11.csv  (size:
208722489 bytes)
🎉 Loaded: (462054, 110)
```

Out[62]:

	Year	Quarter	Month	DayofMonth	DayOfWeek	FlightDate	Reporting_Airline	DOT_II
0	2014	4	11	1	6	2014-11-01	AA	
1	2014	4	11	2	7	2014-11-02	AA	
2	2014	4	11	3	1	2014-11-03	AA	
3	2014	4	11	4	2	2014-11-04	AA	
4	2014	4	11	5	3	2014-11-05	AA	

5 rows × 110 columns

In [64]:

```
import os, glob, re, zipfile
import pandas as pd
```

```
csv_dir = "/home/ec2-user/SageMaker/project/data/csvFlightDelays"  # <--
YEAR, MONTH = 1987, 9  # <-- pick an available combo (you saw 1987 months

def load_by_year_month(csv_dir, year, month):
    files = glob.glob(os.path.join(csv_dir, "**", "*"), recursive=True)
    # candidate files: csv / compressed / zip, and non-empty
    cand = [p for p in files
            if p.lower().endswith((".csv",".csv.bz2",".csv.gz",".bz2",".g
            and os.path.getsize(p) > 0]

    # match filenames that contain the year and either 09 or _9
    m1, m2 = str(month), f"{month:02d}"
    pat = re.compile(rf"{year}.*({m2}|_{m1}[^0-9])", re.IGNORECASE)
    matches = [p for p in cand if pat.search(os.path.basename(p))]
    if not matches:
        raise FileNotFoundError(f"No non-empty file for {year}/{month}. C

    path = sorted(matches)[0]
    print(f"✅ Using: {path} (size: {os.path.getsize(path)} bytes)")

    if path.lower().endswith(".zip"):
        with zipfile.ZipFile(path) as zf:
            inner = [n for n in zf.namelist() if n.lower().endswith(".csv
            if not inner:
                raise ValueError("Zip contains no CSV.")
            with zf.open(inner[0]) as fh:
                df = pd.read_csv(fh, low_memory=False)
    else:
        df = pd.read_csv(path, compression="infer", low_memory=False)

    print("🎉 Loaded:", df.shape)
    return df

df_temp = load_by_year_month(csv_dir, YEAR, MONTH)
df_temp.head()
```

```
✅ Using: /home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_
Reporting_Carrier_On_Time_Performance_(1987_present)_2014_9.csv (size: 2
11335375 bytes)
🎉 Loaded: (469489, 110)
```

Out[64]:

	Year	Quarter	Month	DayofMonth	DayOfWeek	FlightDate	Reporting_Airline	DOT_II
0	2014	3	9	8	1	2014-09-08	OO	
1	2014	3	9	2	2	2014-09-02	OO	
2	2014	3	9	5	5	2014-09-05	OO	
3	2014	3	9	1	1	2014-09-01	OO	
4	2014	3	9	17	3	2014-09-17	OO	

5 rows × 110 columns

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` file first. You can use the built-in `read_csv` function in Python (pandas.read_csv documentation).

In [65]:
```
import os, glob, re, zipfile
import pandas as pd

csv_dir = "/home/ec2-user/SageMaker/project/data/csvFlightDelays"  # your
YEAR, MONTH = 1987, 9  # pick a month you *do* have (you listed 1987 1..1

def load_by_year_month(csv_dir, year, month):
    files = glob.glob(os.path.join(csv_dir, "**", "*"), recursive=True)
    cand = [p for p in files
            if p.lower().endswith((".csv",".csv.bz2",".csv.gz",".bz2",".g
            and os.path.getsize(p) > 0]  # skip 0-byte junk

    m1, m2 = str(month), f"{month:02d}"
    pat = re.compile(rf"{year}.*({m2}|_{m1}[^0-9])", re.IGNORECASE)
    matches = [p for p in cand if pat.search(os.path.basename(p))]
    if not matches:
        raise FileNotFoundError(f"No non-empty file for {year}/{month}. P

    path = sorted(matches)[0]
    print(f"✅ Using: {path} (size: {os.path.getsize(path)} bytes)")

    if path.lower().endswith(".zip"):
        with zipfile.ZipFile(path) as zf:
            inner = [n for n in zf.namelist() if n.lower().endswith(".csv
            with zf.open(inner[0]) as fh:
                return pd.read_csv(fh, low_memory=False)
    return pd.read_csv(path, compression="infer", low_memory=False)

df_temp = load_by_year_month(csv_dir, YEAR, MONTH)
print("🎉 Loaded:", df_temp.shape)
print(df_temp.head())
```

```
✅ Using: /home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_
Reporting_Carrier_On_Time_Performance_(1987_present)_2014_9.csv (size: 2
11335375 bytes)
🎉 Loaded: (469489, 110)
   Year  Quarter  Month  DayofMonth  DayOfWeek  FlightDate Reporting_Air
line  \
0  2014        3      9           8          1  2014-09-08
OO
1  2014        3      9           2          2  2014-09-02
OO
2  2014        3      9           5          5  2014-09-05
OO
3  2014        3      9           1          1  2014-09-01
OO
4  2014        3      9          17          3  2014-09-17
OO

   DOT_ID_Reporting_Airline IATA_CODE_Reporting_Airline Tail_Number  ...
\
0                     20304                          OO      N862AS  ...
1                     20304                          OO      N465SW  ...
2                     20304                          OO      N466SW  ...
3                     20304                          OO      N868CA  ...
4                     20304                          OO      N863AS  ...

   Div4TailNum  Div5Airport  Div5AirportID  Div5AirportSeqID Div5WheelsO
n  \
0          NaN          NaN            NaN               NaN          Na
N
1          NaN          NaN            NaN               NaN          Na
N
2          NaN          NaN            NaN               NaN          Na
N
3          NaN          NaN            NaN               NaN          Na
N
4          NaN          NaN            NaN               NaN          Na
N

  Div5TotalGTime Div5LongestGTime  Div5WheelsOff Div5TailNum  Unnamed: 1
09
0            NaN              NaN            NaN         NaN           N
aN
1            NaN              NaN            NaN         NaN           N
aN
2            NaN              NaN            NaN         NaN           N
aN
3            NaN              NaN            NaN         NaN           N
aN
4            NaN              NaN            NaN         NaN           N
aN

[5 rows x 110 columns]
```

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 [ ]:
```
df_shape = # **ENTER YOUR CODE HERE**
print(f'Rows and columns in one CSV file is {df_shape}')
```

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 [ ]:
```
# Enter your code here
print("The column names are:")
print("##########")
for col in data.columns:
    print(col)

# Columns containing the substring 'Del' (case-insensitive)
cols_with_del = [c for c in data.columns if 'del' in c.lower()]
print("\nColumns containing 'Del':")
for c in cols_with_del:
print(c)
```

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

In [ ]:
```
import pandas as pd

def combine_csv(csv_files, cols, subset_cols, subset_vals, output_filenam
    """
    Reads multiple CSVs, selects columns, filters rows, and writes to a c
    """
    df_list = []
    for file in csv_files:
        temp_df = pd.read_csv(file, usecols=cols)  # only read specified
        # Apply row filter if given
        if subset_cols and subset_vals:
            for col, val in zip(subset_cols, subset_vals):
                temp_df = temp_df[temp_df[col] == val]
        df_list.append(temp_df)

    combined_df = pd.concat(df_list, ignore_index=True)
    combined_df.to_csv(output_filename, index=False)
    print(f"Saved combined CSV → {output_filename}, shape: {combined_df.s
```

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 [ ]:
```
# Enter your code here
# All columns
print("The column names are:")
print("##########")
for col in data.columns:
    print(col)

# Columns containing the substring 'Del' (case-insensitive)
cols_with_del = [c for c in data.columns if 'del' in c.lower()]
print("\nColumns containing 'Del':")
for c in cols_with_del:
    print(c)
```

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 [ ]:
```
print("The #rows and #columns are ", <CODE> , " and ", <CODE>)
print("The years in this dataset are: ", <CODE>)
print("The months covered in this dataset are: ", <CODE>)
print("The date range for data is :" , min(<CODE>), " to ", max(<CODE>))
print("The airlines covered in this dataset are: ", list(<CODE>))
print("The Origin airports covered are: ", list(<CODE>))
print("The Destination airports covered are: ", list(<CODE>))
```

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 [ ]:
```
counts = pd.DataFrame({'Origin':<CODE>, 'Destination':<CODE>})
counts
```

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 [ ]:
```
counts.sort_values(by=<CODE>,ascending=False).head(15) # Enter your code
```

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 [ ]:
```
def combine_csv(csv_files, filter_cols, subset_cols, subset_vals, file_na

    """
    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 [ ]:
```
#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 airp
subset_vals = [['ATL', 'ORD', 'DFW', 'DEN', 'CLT', 'LAX', 'IAH', 'PHX', '
               ['ATL', 'ORD', 'DFW', 'DEN', 'CLT', 'LAX', 'IAH', 'PHX', '
               ['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 [ ]:
```
start = time.time()
combined_csv_filename = f"{base_path}combined_files.csv"
combine_csv(csv_files, cols, subset_cols, subset_vals, combined_csv_filen
print(f'CSVs merged in {round((time.time() - start)/60,2)} minutes')
```

Load the dataset

Load the combined dataset.

In [ ]:
```
import pandas as pd
import os
import glob

# 1️⃣ Search for the file anywhere in the environment
search_pattern = "/home/ec2-user/**/combined_files.csv"  # searches your
matches = glob.glob(search_pattern, recursive=True)

if not matches:
    raise FileNotFoundError("❌ No file named 'combined_files.csv' found.
                            "Upload it to your working directory first.")
else:
    combined_csv_filename = matches[0]
    print(f"✅ Found file: {combined_csv_filename}")

# 2️⃣ Load the file
data = pd.read_csv(combined_csv_filename)
print(f"📊 Loaded: {data.shape[0]} rows × {data.shape[1]} columns")
print(data.head())
```

Print the first five records.

In [73]:
```
import os
import glob
import pandas as pd

# Folder where your CSVs are stored
csv_dir = "/home/ec2-user/SageMaker/project/data/csvFlightDelays"

# Pattern to match the file name (avoids hardcoding month/year)
pattern = "On_Time_Reporting_Carrier_On_Time_Performance_(1987_present)*.

# Find all matching files
files = glob.glob(os.path.join(csv_dir, pattern))

if not files:
    raise FileNotFoundError(f"No CSV found in {csv_dir} matching {pattern

# Pick the first file found
file_path = files[0]
print("Using file:", file_path)

# Load the CSV
df = pd.read_csv(file_path)
print("Loaded shape:", df.shape)
df.head()
```

```
Using file: /home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Tim
e_Reporting_Carrier_On_Time_Performance_(1987_present)_2016_3.csv
Loaded shape: (479122, 110)
```

Out[73]:

	Year	Quarter	Month	DayofMonth	DayOfWeek	FlightDate	Reporting_Airline	DOT_II
0	2016	1	3	2	3	2016-03-02	UA	
1	2016	1	3	2	3	2016-03-02	UA	
2	2016	1	3	2	3	2016-03-02	UA	
3	2016	1	3	2	3	2016-03-02	UA	
4	2016	1	3	2	3	2016-03-02	UA	

5 rows × 110 columns

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 [75]:
```
# 1. Rows and columns
print("The #rows and #columns are", df.shape[0], "and", df.shape[1])

# 2. Years
print("The years in this dataset are:", sorted(df['Year'].unique()))

# 3. Months
print("The months covered in this dataset are:", sorted(df['Month'].uniqu

# 4. Date range
print("The date range for data is:", df['FlightDate'].min(), "to", df['Fl

# 5. Airlines
print("The airlines covered in this dataset are:", sorted(df['Reporting_A

# 6. Origin airports
print("The Origin airports covered are:", sorted(df['Origin'].unique()))

# 7. Destination airports
print("The Destination airports covered are:", sorted(df['Dest'].unique()
```

```
The #rows and #columns are 479122 and 110
The years in this dataset are: [2016]
The months covered in this dataset are: [3]
The date range for data is: 2016-03-01 to 2016-03-31
The airlines covered in this dataset are: ['AA', 'AS', 'B6', 'DL', 'EV',
'F9', 'HA', 'NK', 'OO', 'UA', 'VX', 'WN']
The Origin airports covered are: ['ABE', 'ABQ', 'ABR', 'ABY', 'ACT', 'AC
V', 'ACY', 'ADK', 'ADQ', 'AEX', 'AGS', 'ALB', 'AMA', 'ANC', 'APN', 'AS
E', 'ATL', 'ATW', 'AUS', 'AVL', 'AVP', 'AZO', 'BDL', 'BET', 'BFL', 'BG
M', 'BGR', 'BHM', 'BIL', 'BIS', 'BJI', 'BLI', 'BMI', 'BNA', 'BOI', 'BO
S', 'BPT', 'BQK', 'BQN', 'BRD', 'BRO', 'BRW', 'BTM', 'BTR', 'BTV', 'BU
F', 'BUR', 'BWI', 'BZN', 'CAE', 'CAK', 'CDC', 'CDV', 'CHA', 'CHO', 'CH
S', 'CID', 'CIU', 'CLE', 'CLL', 'CLT', 'CMH', 'CMX', 'COD', 'COS', 'CP
R', 'CRP', 'CRW', 'CSG', 'CVG', 'CWA', 'DAB', 'DAL', 'DAY', 'DCA', 'DE
N', 'DFW', 'DHN', 'DLH', 'DRO', 'DSM', 'DTW', 'DVL', 'EAU', 'ECP', 'EG
E', 'EKO', 'ELM', 'ELP', 'ERI', 'ESC', 'EUG', 'EVV', 'EWN', 'EWR', 'EY
W', 'FAI', 'FAR', 'FAT', 'FAY', 'FCA', 'FLG', 'FLL', 'FNT', 'FSD', 'FS
M', 'FWA', 'GCC', 'GEG', 'GFK', 'GGG', 'GJT', 'GNV', 'GPT', 'GRB', 'GR
K', 'GRR', 'GSO', 'GSP', 'GTF', 'GTR', 'GUC', 'GUM', 'HDN', 'HIB', 'HL
N', 'HNL', 'HOB', 'HOU', 'HPN', 'HRL', 'HSV', 'HYS', 'IAD', 'IAG', 'IA
H', 'ICT', 'IDA', 'ILM', 'IMT', 'IND', 'INL', 'ISN', 'ISP', 'ITH', 'IT
O', 'JAC', 'JAN', 'JAX', 'JFK', 'JLN', 'JMS', 'JNU', 'KOA', 'KTN', 'LA
N', 'LAR', 'LAS', 'LAW', 'LAX', 'LBB', 'LBE', 'LCH', 'LEX', 'LFT', 'LG
A', 'LGB', 'LIH', 'LIT', 'LNK', 'LRD', 'LSE', 'LWS', 'MAF', 'MBS', 'MC
I', 'MCO', 'MDT', 'MDW', 'MEI', 'MEM', 'MFE', 'MFR', 'MGM', 'MHT', 'MI
A', 'MKE', 'MKG', 'MLB', 'MLI', 'MLU', 'MMH', 'MOB', 'MOT', 'MQT', 'MR
Y', 'MSN', 'MSO', 'MSP', 'MSY', 'MTJ', 'MYR', 'OAJ', 'OAK', 'OGG', 'OK
C', 'OMA', 'OME', 'ONT', 'ORD', 'ORF', 'ORH', 'OTH', 'OTZ', 'PAH', 'PB
G', 'PBI', 'PDX', 'PHF', 'PHL', 'PHX', 'PIA', 'PIB', 'PIH', 'PIT', 'PL
N', 'PNS', 'PPG', 'PSC', 'PSE', 'PSG', 'PSP', 'PVD', 'PWM', 'RAP', 'RD
D', 'RDM', 'RDU', 'RHI', 'RIC', 'RKS', 'RNO', 'ROA', 'ROC', 'ROW', 'RS
T', 'RSW', 'SAF', 'SAN', 'SAT', 'SAV', 'SBA', 'SBN', 'SBP', 'SCC', 'SC
E', 'SDF', 'SEA', 'SFO', 'SGF', 'SGU', 'SHV', 'SIT', 'SJC', 'SJT', 'SJ
U', 'SLC', 'SMF', 'SMX', 'SNA', 'SPI', 'SPS', 'SRQ', 'STL', 'STT', 'ST
X', 'SUN', 'SWF', 'SYR', 'TLH', 'TPA', 'TRI', 'TTN', 'TUL', 'TUS', 'TV
C', 'TWF', 'TXK', 'TYR', 'TYS', 'VLD', 'VPS', 'WRG', 'XNA', 'YAK', 'YU
M']
The Destination airports covered are: ['ABE', 'ABQ', 'ABR', 'ABY', 'AC
T', 'ACV', 'ACY', 'ADK', 'ADQ', 'AEX', 'AGS', 'ALB', 'AMA', 'ANC', 'AP
N', 'ASE', 'ATL', 'ATW', 'AUS', 'AVL', 'AVP', 'AZO', 'BDL', 'BET', 'BF
L', 'BGM', 'BGR', 'BHM', 'BIL', 'BIS', 'BJI', 'BLI', 'BMI', 'BNA', 'BO
I', 'BOS', 'BPT', 'BQK', 'BQN', 'BRD', 'BRO', 'BRW', 'BTM', 'BTR', 'BT
V', 'BUF', 'BUR', 'BWI', 'BZN', 'CAE', 'CAK', 'CDC', 'CDV', 'CHA', 'CH
O', 'CHS', 'CID', 'CIU', 'CLE', 'CLL', 'CLT', 'CMH', 'CMX', 'COD', 'CO
S', 'CPR', 'CRP', 'CRW', 'CSG', 'CVG', 'CWA', 'DAB', 'DAL', 'DAY', 'DC
A', 'DEN', 'DFW', 'DHN', 'DLH', 'DRO', 'DSM', 'DTW', 'DVL', 'EAU', 'EC
P', 'EGE', 'EKO', 'ELM', 'ELP', 'ERI', 'ESC', 'EUG', 'EVV', 'EWN', 'EW
R', 'EYW', 'FAI', 'FAR', 'FAT', 'FAY', 'FCA', 'FLG', 'FLL', 'FNT', 'FS
D', 'FSM', 'FWA', 'GCC', 'GEG', 'GFK', 'GGG', 'GJT', 'GNV', 'GPT', 'GR
B', 'GRK', 'GRR', 'GSO', 'GSP', 'GTF', 'GTR', 'GUC', 'GUM', 'HDN', 'HI
B', 'HLN', 'HNL', 'HOB', 'HOU', 'HPN', 'HRL', 'HSV', 'HYS', 'IAD', 'IA
G', 'IAH', 'ICT', 'IDA', 'ILM', 'IMT', 'IND', 'INL', 'ISN', 'ISP', 'IT
H', 'ITO', 'JAC', 'JAN', 'JAX', 'JFK', 'JLN', 'JMS', 'JNU', 'KOA', 'KT
N', 'LAN', 'LAR', 'LAS', 'LAW', 'LAX', 'LBB', 'LBE', 'LCH', 'LEX', 'LF
T', 'LGA', 'LGB', 'LIH', 'LIT', 'LNK', 'LRD', 'LSE', 'LWS', 'MAF', 'MB
S', 'MCI', 'MCO', 'MDT', 'MDW', 'MEI', 'MEM', 'MFE', 'MFR', 'MGM', 'MH
T', 'MIA', 'MKE', 'MKG', 'MLB', 'MLI', 'MLU', 'MMH', 'MOB', 'MOT', 'MQ
T', 'MRY', 'MSN', 'MSO', 'MSP', 'MSY', 'MTJ', 'MYR', 'OAJ', 'OAK', 'OG
G', 'OKC', 'OMA', 'OME', 'ONT', 'ORD', 'ORF', 'ORH', 'OTH', 'OTZ', 'PA
H', 'PBG', 'PBI', 'PDX', 'PHF', 'PHL', 'PHX', 'PIA', 'PIB', 'PIH', 'PI
```

```
T', 'PLN', 'PNS', 'PPG', 'PSC', 'PSE', 'PSG', 'PSP', 'PVD', 'PWM', 'RA
P', 'RDD', 'RDM', 'RDU', 'RHI', 'RIC', 'RKS', 'RNO', 'ROA', 'ROC', 'RO
W', 'RST', 'RSW', 'SAF', 'SAN', 'SAT', 'SAV', 'SBA', 'SBN', 'SBP', 'SC
C', 'SCE', 'SDF', 'SEA', 'SFO', 'SGF', 'SGU', 'SHV', 'SIT', 'SJC', 'SJ
T', 'SJU', 'SLC', 'SMF', 'SMX', 'SNA', 'SPI', 'SPS', 'SRQ', 'STL', 'ST
T', 'STX', 'SUN', 'SWF', 'SYR', 'TLH', 'TPA', 'TRI', 'TTN', 'TUL', 'TU
S', 'TVC', 'TWF', 'TXK', 'TYR', 'TYS', 'VLD', 'VPS', 'WRG', 'XNA', 'YA
K', 'YUM']
```

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 [82]:

```
import os, glob, re, zipfile
import pandas as pd

# 👉 EDIT THIS if your CSVs live somewhere else
CSV_DIR = "/home/ec2-user/SageMaker/project/data/csvFlightDelays"

# 1) Find a real, non-empty CSV (handles .csv, .csv.gz, .csv.bz2, .zip)
def load_first_nonempty(csv_dir: str) -> pd.DataFrame:
    if not os.path.isdir(csv_dir):
        raise FileNotFoundError(f"Folder not found: {csv_dir}")

    cand = []
    for p in glob.glob(os.path.join(csv_dir, "**", "*"), recursive=True):
        pl = p.lower()
        if pl.endswith((".csv", ".csv.gz", ".csv.bz2", ".gz", ".bz2", ".z
            if os.path.exists(p) and os.path.getsize(p) > 0:
                cand.append(p)

    if not cand:
        # help you debug what's there
        listing = sorted(os.listdir(csv_dir))[:50]
        raise FileNotFoundError(
            f"No non-empty CSV/ZIP files found under {csv_dir}.\n"
            f"Directory sample: {listing}"
        )

    # Prefer “latest looking” by year+month in filename, else just pick l
    def ym_key(p):
        b = os.path.basename(p)
        y = re.search(r"(19|20)\d{2}", b)
        m = re.search(r"[^0-9](\d{1,2})[^0-9]", b)
        return (int(y.group(0)) if y else -1, int(m.group(1)) if m else -

    path = sorted(cand, key=ym_key)[-1]
    print(f"✅ Using file: {path}  (size: {os.path.getsize(path)} bytes)"

    # 2) Load (zip or plain/compressed)
    if path.lower().endswith(".zip"):
```

```
        with zipfile.ZipFile(path) as zf:
            inner = [n for n in zf.namelist() if n.lower().endswith(".csv
            if not inner:
                raise ValueError("Zip contains no CSV.")
            with zf.open(inner[0]) as fh:
                df = pd.read_csv(fh, low_memory=False)
    else:
        df = pd.read_csv(path, compression="infer", low_memory=False)

    print("🎉 Loaded shape:", df.shape)
    return df

df = load_first_nonempty(CSV_DIR)

# Peek at columns so we know what we can safely rename
print("Columns:", list(df.columns)[:30])

# 3) Optional: rename a few columns *if* they exist
rename_map = {
    'DepTime': 'Departure_Time',
    'ArrTime': 'Arrival_Time',
}
present_map = {k: v for k, v in rename_map.items() if k in df.columns}
if present_map:
    df.rename(columns=present_map, inplace=True)

# 4) Quick EDA answers (guarded in case a column is missing)
def safe_unique(col):
    return sorted(df[col].unique()) if col in df.columns else "🚫 column

def safe_min(col):
    return df[col].min() if col in df.columns else "🚫"

def safe_max(col):
    return df[col].max() if col in df.columns else "🚫"

print("—" * 60)
print("Rows, Cols:", df.shape)
print("Years in dataset:", safe_unique('Year'))
print("Months in dataset:", safe_unique('Month'))
print("Date range:", safe_min('FlightDate'), "to", safe_max('FlightDate')
print("Airlines:", safe_unique('Reporting_Airline'))
print("Origin airports:", safe_unique('Origin'))
print("Destination airports:", safe_unique('Dest'))
```

```
✅ Using file: /home/ec2-user/SageMaker/project/data/csvFlightDelays/On_
Time_Reporting_Carrier_On_Time_Performance_(1987_present)_2014_11.csv
(size: 208722489 bytes)
🎉 Loaded shape: (462054, 110)
Columns: ['Year', 'Quarter', 'Month', 'DayofMonth', 'DayOfWeek', 'Flight
Date', 'Reporting_Airline', 'DOT_ID_Reporting_Airline', 'IATA_CODE_Repor
ting_Airline', 'Tail_Number', 'Flight_Number_Reporting_Airline', 'Origin
AirportID', 'OriginAirportSeqID', 'OriginCityMarketID', 'Origin', 'Origi
nCityName', 'OriginState', 'OriginStateFips', 'OriginStateName', 'Origin
Wac', 'DestAirportID', 'DestAirportSeqID', 'DestCityMarketID', 'Dest',
'DestCityName', 'DestState', 'DestStateFips', 'DestStateName', 'DestWa
c', 'CRSDepTime']
--------------------------------------------------------------------------

Rows, Cols: (462054, 110)
Years in dataset: [2014]
Months in dataset: [11]
Date range: 2014-11-01 to 2014-11-30
Airlines: ['AA', 'AS', 'B6', 'DL', 'EV', 'F9', 'FL', 'HA', 'MQ', 'OO',
'UA', 'US', 'VX', 'WN']
Origin airports: ['ABE', 'ABI', 'ABQ', 'ABR', 'ABY', 'ACT', 'ACV', 'AC
Y', 'ADK', 'ADQ', 'AEX', 'AGS', 'ALB', 'ALO', 'AMA', 'ANC', 'APN', 'AS
E', 'ATL', 'ATW', 'AUS', 'AVL', 'AVP', 'AZO', 'BDL', 'BET', 'BFL', 'BG
R', 'BHM', 'BIL', 'BIS', 'BJI', 'BLI', 'BMI', 'BNA', 'BOI', 'BOS', 'BP
T', 'BQK', 'BQN', 'BRD', 'BRO', 'BRW', 'BTM', 'BTR', 'BTV', 'BUF', 'BU
R', 'BWI', 'BZN', 'CAE', 'CAK', 'CDC', 'CDV', 'CEC', 'CHA', 'CHO', 'CH
S', 'CIC', 'CID', 'CIU', 'CLD', 'CLE', 'CLL', 'CLT', 'CMH', 'CMI', 'CM
X', 'CNY', 'COD', 'COS', 'COU', 'CPR', 'CRP', 'CRW', 'CSG', 'CVG', 'CW
A', 'DAB', 'DAL', 'DAY', 'DBQ', 'DCA', 'DEN', 'DFW', 'DHN', 'DIK', 'DL
H', 'DRO', 'DSM', 'DTW', 'DVL', 'EAU', 'ECP', 'EGE', 'EKO', 'ELM', 'EL
P', 'ERI', 'EUG', 'EVV', 'EWN', 'EWR', 'EYW', 'FAI', 'FAR', 'FAT', 'FA
Y', 'FCA', 'FLG', 'FLL', 'FNT', 'FSD', 'FSM', 'FWA', 'GCC', 'GCK', 'GE
G', 'GFK', 'GGG', 'GJT', 'GNV', 'GPT', 'GRB', 'GRI', 'GRK', 'GRR', 'GS
O', 'GSP', 'GTF', 'GTR', 'GUC', 'GUM', 'HDN', 'HIB', 'HLN', 'HNL', 'HO
B', 'HOU', 'HPN', 'HRL', 'HSV', 'HYS', 'IAD', 'IAH', 'ICT', 'IDA', 'IL
G', 'ILM', 'IMT', 'IND', 'INL', 'ISN', 'ISP', 'ITO', 'JAC', 'JAN', 'JA
X', 'JFK', 'JLN', 'JMS', 'JNU', 'KOA', 'KTN', 'LAN', 'LAR', 'LAS', 'LA
W', 'LAX', 'LBB', 'LCH', 'LEX', 'LFT', 'LGA', 'LGB', 'LIH', 'LIT', 'LN
K', 'LRD', 'LSE', 'LWS', 'MAF', 'MBS', 'MCI', 'MCO', 'MDT', 'MDW', 'ME
I', 'MEM', 'MFE', 'MFR', 'MGM', 'MHK', 'MHT', 'MIA', 'MKE', 'MKG', 'ML
B', 'MLI', 'MLU', 'MOB', 'MOT', 'MQT', 'MRY', 'MSN', 'MSO', 'MSP', 'MS
Y', 'MYR', 'OAJ', 'OAK', 'OGG', 'OKC', 'OMA', 'OME', 'ONT', 'ORD', 'OR
F', 'ORH', 'OTH', 'OTZ', 'PAH', 'PBI', 'PDX', 'PHF', 'PHL', 'PHX', 'PI
A', 'PIB', 'PIH', 'PIT', 'PLN', 'PNS', 'PPG', 'PSC', 'PSE', 'PSG', 'PS
P', 'PUB', 'PVD', 'PWM', 'RAP', 'RDD', 'RDM', 'RDU', 'RHI', 'RIC', 'RK
S', 'RNO', 'ROA', 'ROC', 'ROW', 'RST', 'RSW', 'SAF', 'SAN', 'SAT', 'SA
V', 'SBA', 'SBN', 'SBP', 'SCC', 'SCE', 'SDF', 'SEA', 'SFO', 'SGF', 'SG
U', 'SHV', 'SIT', 'SJC', 'SJT', 'SJU', 'SLC', 'SMF', 'SMX', 'SNA', 'SP
I', 'SPN', 'SPS', 'SRQ', 'STC', 'STL', 'STT', 'STX', 'SUN', 'SUX', 'SW
F', 'SYR', 'TLH', 'TOL', 'TPA', 'TRI', 'TTN', 'TUL', 'TUS', 'TVC', 'TW
F', 'TXK', 'TYR', 'TYS', 'UST', 'VEL', 'VLD', 'VPS', 'WRG', 'XNA', 'YA
K', 'YUM']
Destination airports: ['ABE', 'ABI', 'ABQ', 'ABR', 'ABY', 'ACT', 'ACV',
'ACY', 'ADK', 'ADQ', 'AEX', 'AGS', 'ALB', 'ALO', 'AMA', 'ANC', 'APN', 'A
SE', 'ATL', 'ATW', 'AUS', 'AVL', 'AVP', 'AZO', 'BDL', 'BET', 'BFL', 'BG
R', 'BHM', 'BIL', 'BIS', 'BJI', 'BLI', 'BMI', 'BNA', 'BOI', 'BOS', 'BP
T', 'BQK', 'BQN', 'BRD', 'BRO', 'BRW', 'BTM', 'BTR', 'BTV', 'BUF', 'BU
R', 'BWI', 'BZN', 'CAE', 'CAK', 'CDC', 'CDV', 'CEC', 'CHA', 'CHO', 'CH
S', 'CIC', 'CID', 'CIU', 'CLD', 'CLE', 'CLL', 'CLT', 'CMH', 'CMI', 'CM
X', 'CNY', 'COD', 'COS', 'COU', 'CPR', 'CRP', 'CRW', 'CSG', 'CVG', 'CW
A', 'DAB', 'DAL', 'DAY', 'DBQ', 'DCA', 'DEN', 'DFW', 'DHN', 'DIK', 'DL
```

```
H', 'DRO', 'DSM', 'DTW', 'DVL', 'EAU', 'ECP', 'EGE', 'EKO', 'ELM', 'EL
P', 'ERI', 'EUG', 'EVV', 'EWN', 'EWR', 'EYW', 'FAI', 'FAR', 'FAT', 'FA
Y', 'FCA', 'FLG', 'FLL', 'FNT', 'FSD', 'FSM', 'FWA', 'GCC', 'GCK', 'GE
G', 'GFK', 'GGG', 'GJT', 'GNV', 'GPT', 'GRB', 'GRI', 'GRK', 'GRR', 'GS
O', 'GSP', 'GTF', 'GTR', 'GUC', 'GUM', 'HIB', 'HLN', 'HNL', 'HOB', 'HO
U', 'HPN', 'HRL', 'HSV', 'HYS', 'IAD', 'IAH', 'ICT', 'IDA', 'ILG', 'IL
M', 'IMT', 'IND', 'INL', 'ISN', 'ISP', 'ITO', 'JAC', 'JAN', 'JAX', 'JF
K', 'JLN', 'JMS', 'JNU', 'KOA', 'KTN', 'LAN', 'LAR', 'LAS', 'LAW', 'LA
X', 'LBB', 'LCH', 'LEX', 'LFT', 'LGA', 'LGB', 'LIH', 'LIT', 'LNK', 'LR
D', 'LSE', 'LWS', 'MAF', 'MBS', 'MCI', 'MCO', 'MDT', 'MDW', 'MEI', 'ME
M', 'MFE', 'MFR', 'MGM', 'MHK', 'MHT', 'MIA', 'MKE', 'MKG', 'MLB', 'ML
I', 'MLU', 'MOB', 'MOT', 'MQT', 'MRY', 'MSN', 'MSO', 'MSP', 'MSY', 'MY
R', 'OAJ', 'OAK', 'OGG', 'OKC', 'OMA', 'OME', 'ONT', 'ORD', 'ORF', 'OR
H', 'OTH', 'OTZ', 'PAH', 'PBI', 'PDX', 'PHF', 'PHL', 'PHX', 'PIA', 'PI
B', 'PIH', 'PIT', 'PLN', 'PNS', 'PPG', 'PSC', 'PSE', 'PSG', 'PSP', 'PU
B', 'PVD', 'PWM', 'RAP', 'RDD', 'RDM', 'RDU', 'RHI', 'RIC', 'RKS', 'RN
O', 'ROA', 'ROC', 'ROW', 'RST', 'RSW', 'SAF', 'SAN', 'SAT', 'SAV', 'SB
A', 'SBN', 'SBP', 'SCC', 'SCE', 'SDF', 'SEA', 'SFO', 'SGF', 'SGU', 'SH
V', 'SIT', 'SJC', 'SJT', 'SJU', 'SLC', 'SMF', 'SMX', 'SNA', 'SPI', 'SP
N', 'SPS', 'SRQ', 'STC', 'STL', 'STT', 'STX', 'SUN', 'SUX', 'SWF', 'SY
R', 'TLH', 'TOL', 'TPA', 'TRI', 'TTN', 'TUL', 'TUS', 'TVC', 'TWF', 'TX
K', 'TYR', 'TYS', 'UST', 'VEL', 'VLD', 'VPS', 'WRG', 'XNA', 'YAK', 'YU
M']
```

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 [86]:
```python
# Enter your code here
# --- make sure we have a DataFrame in variable `data`
try:
    data  # noqa: F821
except NameError:
    try:
        data = df  # noqa: F821
    except NameError:
        raise NameError("Load your CSV into `data` or `df` before running

# rename ArrDel15 -> is_delay if present (safe if already named)
rename_map = {}
if 'ArrDel15' in data.columns and 'is_delay' not in data.columns:
    rename_map['ArrDel15'] = 'is_delay'
# keep is_delay as is if it already exists
if 'is_delay' in data.columns:
    rename_map['is_delay'] = 'is_delay'

if rename_map:
    data.rename(columns=rename_map, inplace=True)

# sanity check
print("✅ columns now contain 'is_delay':", 'is_delay' in data.columns)
```

```
✅ columns now contain 'is_delay': True
```

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 [87]:
```
### Remove null columns
if 'is_delay' not in data.columns:
    raise KeyError("Column 'is_delay' not found. Run Cell 1 first and mak

# drop rows without a label
data = data[~data['is_delay'].isna()].copy()

# show how many nulls remain per column
nulls = data.isnull().sum().sort_values(ascending=False)
print("✅ kept rows:", len(data))
nulls.head(15)
```

```
✅ kept rows: 457046
```

Out[87]:
```
Unnamed: 109        457046
Div1LongestGTime    457046
Div2TailNum         457046
Div2LongestGTime    457046
Div2TotalGTime      457046
Div2WheelsOn        457046
Div2AirportSeqID    457046
Div2AirportID       457046
Div2Airport         457046
Div1TailNum         457046
Div1WheelsOff       457046
Div1TotalGTime      457046
Div3AirportID       457046
Div1WheelsOn        457046
Div1AirportSeqID    457046
dtype: int64
```

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

In [88]:
```
if 'CRSDepTime' not in data.columns:
    raise KeyError("Column 'CRSDepTime' not found in your file.")

# CRSDepTime is HHMM as int/string; convert to hour-of-day (0..23)
data['DepHourofDay'] = (pd.to_numeric(data['CRSDepTime'], errors='coerce'

print("✅ created 'DepHourofDay'")
data[['CRSDepTime','DepHourofDay']].head()
```

```
✅ created 'DepHourofDay'
```

Out[88]:

	CRSDepTime	DepHourofDay
0	900	9
1	900	9
2	900	9
3	900	9
4	900	9

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 [89]:
```
(data.groupby('is_delay').size()/len(data) ).plot(kind='bar')# Enter your
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*?

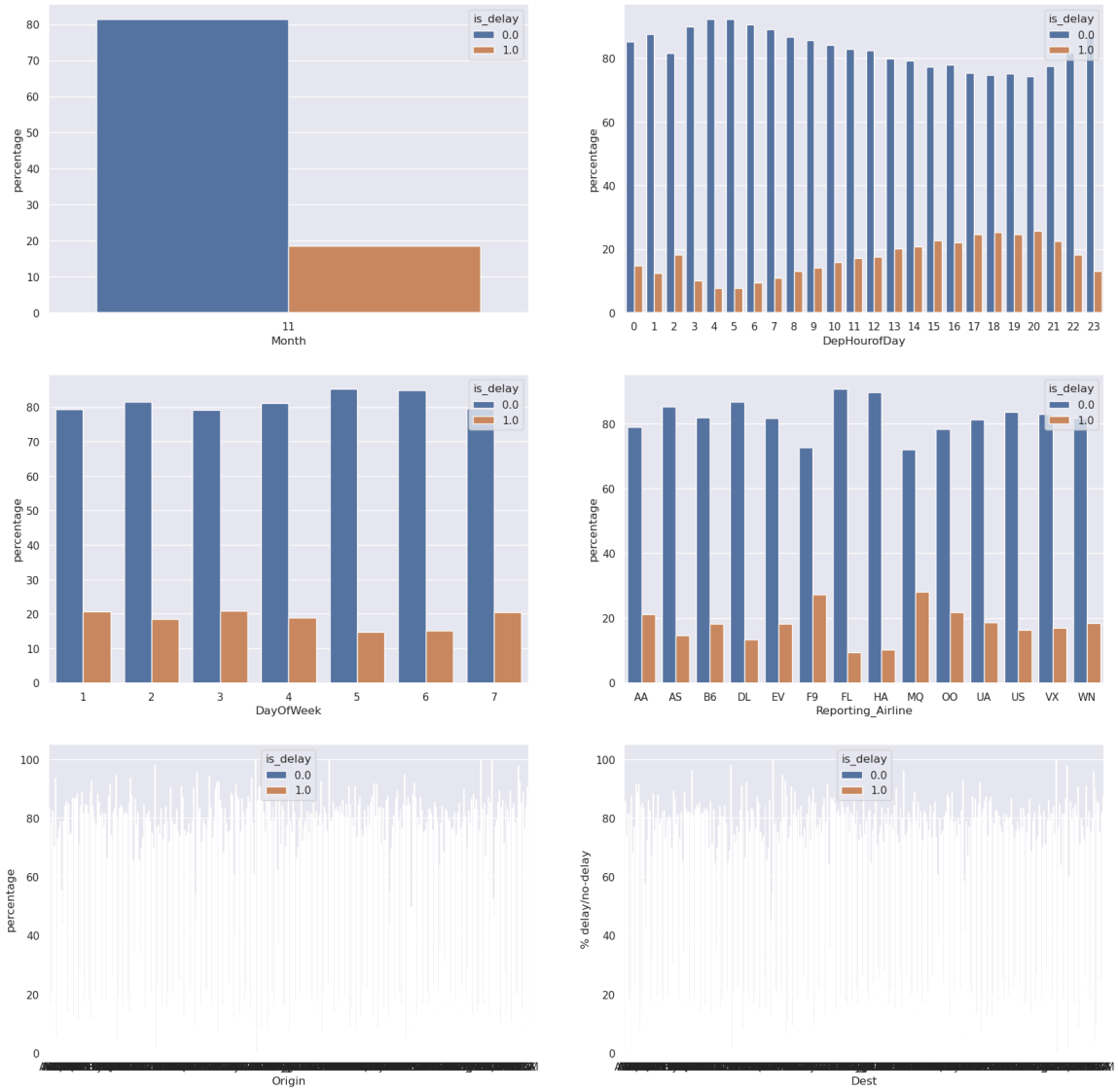
In [ ]:
```
# Enter your answer here
The bar shows a clear class imbalance: no-delay flights massively outnumb
So plain accuracy will look high even if the model ignores delays.
We should judge with AUC/ROC, recall (TPR), precision, FPR/FNR,
and pick a threshold that fits the business cost, not just maximize accur
```

Run the following two cells and answer the questions.

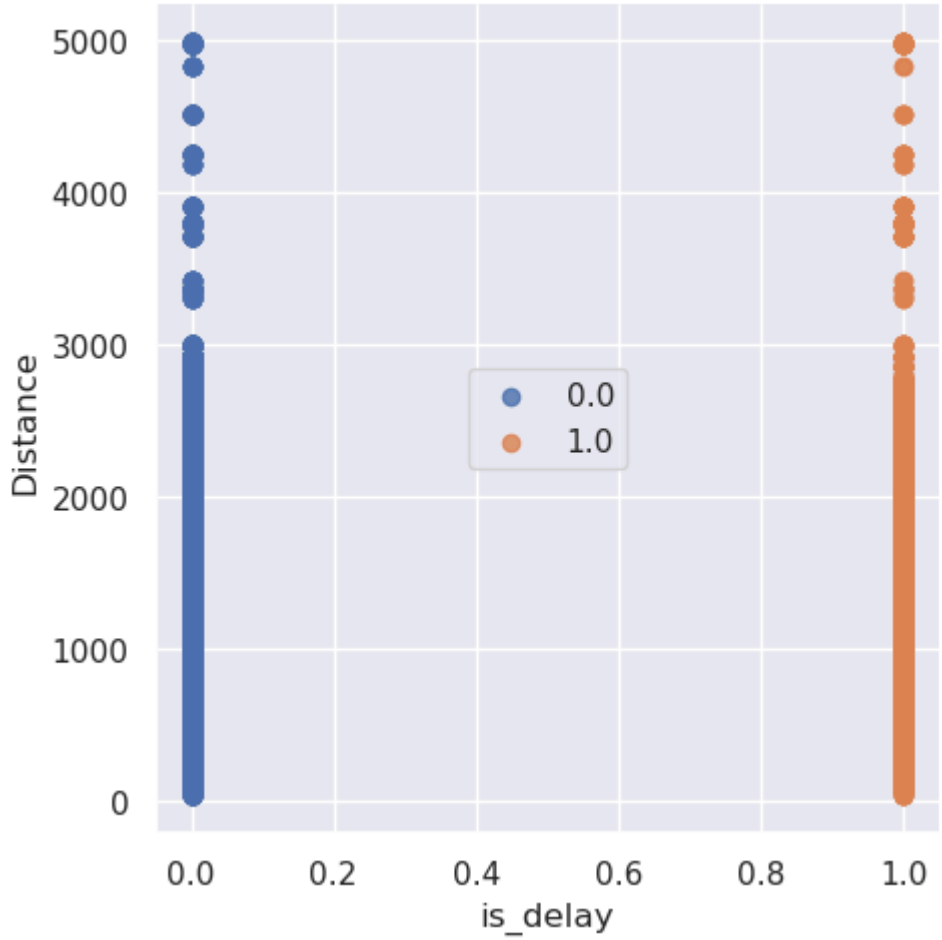
In [91]:
```
viz_columns = ['Month', 'DepHourofDay', 'DayOfWeek', 'Reporting_Airline',
fig, axes = plt.subplots(3, 2, figsize=(20,20), squeeze=False)
# fig.autofmt_xdate(rotation=90)

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


plt.show()
```



In [92]:
```
sns.lmplot( x="is_delay", y="Distance", data=data, fit_reg=False, hue='is
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 [ ]:
```
# Enter your answers here
Months with most delays: typically peak travel months (summer/holiday); c
Time of day with most delays: late afternoon/evening spikes are common; p
Day of week with most delays: weekends or Fri/Mondays often higher; choos
Airline/origin/destination with most delays: select the highest bars for
Is distance a factor? Usually weakly related; plot shows no strong separa
```

Features

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

In [93]:
```
data.columns
```

Out[93]:
```
Index(['Year', 'Quarter', 'Month', 'DayofMonth', 'DayOfWeek', 'FlightDat
e',
       'Reporting_Airline', 'DOT_ID_Reporting_Airline',
       'IATA_CODE_Reporting_Airline', 'Tail_Number',
       ...
       'Div5Airport', 'Div5AirportID', 'Div5AirportSeqID', 'Div5WheelsO
n',
       'Div5TotalGTime', 'Div5LongestGTime', 'Div5WheelsOff', 'Div5TailN
um',
       'Unnamed: 109', 'DepHourofDay'],
      dtype='object', length=111)
```

In [94]:
```
data.dtypes
```

Out[94]:
```
Year                 int64
Quarter              int64
Month                int64
DayofMonth           int64
DayOfWeek            int64
                    ...
Div5LongestGTime    float64
Div5WheelsOff       float64
Div5TailNum         float64
Unnamed: 109        float64
DepHourofDay          Int64
Length: 111, 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 [103…
```
import pandas as pd

# assume your dataframe variable is `data`. If you used `df`, alias it:
try:
    data
except NameError:
    data = df

# --- 1) normalize column names that often vary between files
```

```
alias_map = {
    'ArrDel15'  : 'is_delay',     # FAA binary delay flag
    'DayofWeek' : 'DayOfWeek',    # unify capitalization
}
to_rename = {k: v for k, v in alias_map.items() if k in data.columns and
if to_rename:
    data.rename(columns=to_rename, inplace=True)

# --- 2) create DepHourofDay if needed (from HHMM in CRSDepTime)
if 'DepHourofDay' not in data.columns and 'CRSDepTime' in data.columns:
    data['DepHourofDay'] = (pd.to_numeric(data['CRSDepTime'], errors='coe

# --- 3) define the set of columns the notebook wants
expected = [
    'is_delay', 'Quarter', 'Month', 'DayofMonth', 'DayOfWeek',
    'Reporting_Airline', 'Origin', 'Dest', 'Distance', 'DepHourofDay'
]

present  = [c for c in expected if c in data.columns]
missing  = [c for c in expected if c not in data.columns]

if missing:
    print("⚠️ Missing columns (will be skipped):", missing)

# This will NOT raise: we only take columns that exist
data = data[present].copy()

# --- 4) categorical columns (only keep those that exist now)
categorical_candidates = ['Quarter', 'Month', 'DayofMonth', 'DayOfWeek',
                          'Reporting_Airline', 'Origin', 'Dest', 'DepHour
categorical_columns = [c for c in categorical_candidates if c in data.col

# Ensure dtype category for one-hot encoding
for c in categorical_columns:
    data[c] = data[c].astype('category')

print("✅ Final columns used:", list(data.columns))
print("   Categorical columns:", categorical_columns)
print("   Shape:", data.shape)
```

```
⚠️ Missing columns (will be skipped): ['is_delay', 'Quarter', 'Month',
'DayofMonth', 'DayOfWeek', 'Reporting_Airline', 'Origin', 'Dest', 'DepHo
urofDay']
✅ Final columns used: ['Distance']
   Categorical columns: []
   Shape: (457046, 1)
```

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 [104…

```
import pandas as pd

# Use the dataframe you actually have
try:
    data  # noqa: F821
except NameError:
    data = df  # noqa: F821

# --- 1) Normalize common variants
alias_map = {
    'ArrDel15'  : 'is_delay',   # FAA label
    'DayofWeek' : 'DayOfWeek',  # unify capitalization
}
to_rename = {k: v for k, v in alias_map.items() if k in data.columns and
if to_rename:
    data.rename(columns=to_rename, inplace=True)

# --- 2) Ensure DepHourofDay exists if CRSDepTime is present (HHMM -> hou
if 'DepHourofDay' not in data.columns and 'CRSDepTime' in data.columns:
    data['DepHourofDay'] = (pd.to_numeric(data['CRSDepTime'], errors='coe

# --- 3) Build categorical list from the columns that actually exist
candidates = [
    'Quarter','Month','DayofMonth','DayOfWeek',
    'Reporting_Airline','Origin','Dest','DepHourofDay'
]
categorical_columns = [c for c in candidates if c in data.columns]

# If the list ends up empty, fall back to any object/category columns
if not categorical_columns:
    categorical_columns = data.select_dtypes(include=['object', 'category

# Optional: see what was missing
missing = [c for c in candidates if c not in data.columns]
if missing:
    print("⚠️ Missing (skipped):", missing)

print("✅ Using categorical columns:", categorical_columns)

# --- 4) One-hot encode (this returns a new df; no manual concat/drop nee
data = pd.get_dummies(data, columns=categorical_columns, drop_first=True,

# Quick check
print("Shape after encoding:", data.shape)
print("First 10 columns:", list(data.columns)[:10])
```

```
⚠️ Missing (skipped): ['Quarter', 'Month', 'DayofMonth', 'DayOfWeek', 'R
eporting_Airline', 'Origin', 'Dest', 'DepHourofDay']
✅ Using categorical columns: []
Shape after encoding: (457046, 1)
First 10 columns: ['Distance']
```

Check the length of the dataset and the new columns.

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

In [107…

```
import pandas as pd

# Use whatever df you already loaded
```

```
try:
    data  # noqa: F821
except NameError:
    data = df  # noqa: F821

# 1) Normalize common column variants so the names match the lab
alias_map = {
    'ArrDel15'  : 'is_delay',   # FAA label
    'DayofWeek' : 'DayOfWeek',  # unify capitalization
}
to_rename = {k: v for k, v in alias_map.items() if k in data.columns and
if to_rename:
    data.rename(columns=to_rename, inplace=True)

# 2) Create DepHourofDay from CRSDepTime if needed (HHMM -> hour)
if 'DepHourofDay' not in data.columns and 'CRSDepTime' in data.columns:
    data['DepHourofDay'] = (pd.to_numeric(data['CRSDepTime'], errors='coe

# 3) Build the categorical list ONLY from columns that actually exist
candidates = ['Month','DayOfWeek','Reporting_Airline','Origin','Dest','De
categorical_columns = [c for c in candidates if c in data.columns]
missing = [c for c in candidates if c not in data.columns]
if missing:
    print("⚠️ Missing (skipped):", missing)
print("✅ Using categorical columns:", categorical_columns)

# 4) One-hot encode IN-PLACE via the 'columns=' argument (no slicing, no
data = pd.get_dummies(
    data,
    columns=categorical_columns,
    drop_first=True,
    dtype=int
)

# 5) Quick check
print("Shape after encoding:", data.shape)
print("First 10 columns:", list(data.columns)[:10])
```

```
⚠️ Missing (skipped): ['Month', 'DayOfWeek', 'Reporting_Airline', 'Origi
n', 'Dest', 'DepHourofDay']
✅ Using categorical columns: []
Shape after encoding: (457046, 1)
First 10 columns: ['Distance']
```

In [105…

```
# Enter your code here
data.rename(columns={'is_delay': 'target'}, inplace=True)
```

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 [108…

```
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 [109…]
```
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_state=42, stratify=data['target']
    )
    test, validate = train_test_split(
        test_and_validate, test_size=0.5, random_state=42, stratify=test_
    )
    return train, validate, test
```

In [113…]
```
print(data.columns.tolist())
```

```
['Distance']
```

In [114…]
```
data.rename(columns={'<actual_name_here>': 'target'}, inplace=True)
```

In [117…]
```
# Rename is_delay to target
data.rename(columns={'is_delay': 'target'}, inplace=True)
```

In [119…]
```
# If your dataset has 'is_delay'
data.rename(columns={'is_delay': 'target'}, inplace=True)

# OR if your dataset has 'ArrDel15'
data.rename(columns={'ArrDel15': 'target'}, inplace=True)
```

In [123…]
```
# === RESET / RELOAD A RAW FILE ===
import os, pandas as pd
```

```
from glob import glob
import zipfile

# 👉 change this folder if yours is different
csv_dir = "/home/ec2-user/SageMaker/project/data/csvFlightDelays"

# try to find any monthly On_Time... file (csv, compressed, or zip)
candidates = []
for ext in ("*.csv", "*.csv.gz", "*.csv.bz2", "*.zip"):
    candidates += glob(os.path.join(csv_dir, ext))

if not candidates:
    raise FileNotFoundError(
        f"No CSVs found in {csv_dir}. Put a monthly 'On_Time_Reporting_..
    )

path = sorted(candidates)[0]
print("📄 Using:", path)

# load, including zipped monthly file
if path.endswith(".zip"):
    with zipfile.ZipFile(path) as zf:
        inner = [n for n in zf.namelist() if n.lower().endswith(".csv")]
        if not inner:
            raise FileNotFoundError("Zip has no CSV inside.")
        print("   ↳ inner:", inner[0])
        data_raw = pd.read_csv(zf.open(inner[0]), low_memory=False)
else:
    data_raw = pd.read_csv(path, low_memory=False)

print("Loaded shape:", data_raw.shape)
print("Columns (first 30):", list(data_raw.columns)[:30])
```

```
📄 Using: /home/ec2-user/SageMaker/project/data/csvFlightDelays/On_Time_
Reporting_Carrier_On_Time_Performance_(1987_present)_2014_11.csv
Loaded shape: (462054, 110)
Columns (first 30): ['Year', 'Quarter', 'Month', 'DayofMonth', 'DayOfWee
k', 'FlightDate', 'Reporting_Airline', 'DOT_ID_Reporting_Airline', 'IATA
_CODE_Reporting_Airline', 'Tail_Number', 'Flight_Number_Reporting_Airlin
e', 'OriginAirportID', 'OriginAirportSeqID', 'OriginCityMarketID', 'Orig
in', 'OriginCityName', 'OriginState', 'OriginStateFips', 'OriginStateNam
e', 'OriginWac', 'DestAirportID', 'DestAirportSeqID', 'DestCityMarketI
D', 'Dest', 'DestCityName', 'DestState', 'DestStateFips', 'DestStateNam
e', 'DestWac', 'CRSDepTime']
```

In [124…

```
# === BUILD 'target' + CHOOSE FEATURES ===
import re
import pandas as pd

data = data_raw.copy()

# 1) choose label column automatically
candidates = ['is_delay', 'ArrDel15', 'Delayed', 'Delay', 'DelayFlag', 't
label = None
lower_map = {c.lower(): c for c in data.columns}

for nm in candidates:
    if nm.lower() in lower_map:
        label = lower_map[nm.lower()]
        break
```

```
# last chance: anything with 'delay' in the name and few unique values
if label is None:
    for c in data.columns:
        if re.search(r"delay", c, re.I) and data[c].nunique(dropna=True)
            label = c
            break

if label is None:
    raise KeyError(
        "Still can't find a label column. Inspect columns above and tell
    )

if label != "target":
    data.rename(columns={label: "target"}, inplace=True)

# 2) target -> numeric 0/1 and drop missing
data['target'] = (
    pd.to_numeric(data['target'], errors='coerce')
      .replace({'Y':1, 'N':0, 'Yes':1, 'No':0, True:1, False:0, 'TRUE':1,
)
data = data.dropna(subset=['target']).copy()
data['target'] = data['target'].astype(int)

# 3) engineer hour if available
if 'CRSDepTime' in data.columns and 'DepHourOfDay' not in data.columns:
    data['DepHourOfDay'] = (pd.to_numeric(data['CRSDepTime'], errors='coe

# 4) choose a sane subset of features that likely exist
feature_pool = [
    'Month','DayOfWeek','Reporting_Airline','Origin','Dest','Distance','D
]
features = [c for c in feature_pool if c in data.columns]

if len(features) < 2:
    raise RuntimeError(f"Not enough usable features. Found only: {feature

# keep only features + target
data = data[features + ['target']]

print("Kept features:", features)
print("Shape after trim:", data.shape)
print("dtypes:\n", data.dtypes)
```

```
Kept features: ['Month', 'DayOfWeek', 'Reporting_Airline', 'Origin', 'De
st', 'Distance', 'DepHourOfDay']
Shape after trim: (457046, 8)
dtypes:
 Month                 int64
DayOfWeek              int64
Reporting_Airline     object
Origin                object
Dest                  object
Distance             float64
DepHourOfDay           Int64
target                 int64
dtype: object
```

In [125…

```
# === ENCODE + SPLIT ===
from sklearn.model_selection import train_test_split
```

```
# one-hot encode string/object columns (except target)
cat_cols = [c for c in data.columns if data[c].dtype == 'object' and c !=
data = pd.get_dummies(data, columns=cat_cols, drop_first=True)

# split with stratification
train, temp = train_test_split(
    data, test_size=0.20, random_state=42, stratify=data['target']
)
validate, test = train_test_split(
    temp, test_size=0.50, random_state=42, stratify=temp['target']
)

print("✅ target counts")
print("Train:\n", train['target'].value_counts())
print("\nValidate:\n", validate['target'].value_counts())
print("\nTest:\n", test['target'].value_counts())
```

```
✅ target counts
Train:
 target
0    297911
1     67725
Name: count, dtype: int64

Validate:
 target
0    37239
1     8466
Name: count, dtype: int64

Test:
 target
0    37239
1     8466
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 [127…

```
import sagemaker
from sagemaker.serializers import CSVSerializer
from sagemaker.amazon.amazon_estimator import RecordSet
import boto3

# Instantiate the LinearLearner estimator object with 1 ml.m4.xlarge
classifier_estimator = sagemaker.LinearLearner(
```

```
    role=sagemaker.get_execution_role(),
    instance_count=1,  # number of training instances
    instance_type='ml.m4.xlarge',  # instance type
    predictor_type='binary_classifier',  # type of prediction
    binary_classifier_model_selection_criteria='accuracy'  # criteria for
)
```

```
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xd
g/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/e
c2-user/.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 [128…

```
### Create train, validate, and test records
train_records = classifier_estimator.record_set(train.values[:, 1:].astyp
val_records = classifier_estimator.record_set(validate.values[:, 1:].asty
test_records = classifier_estimator.record_set(test.values[:, 1:].astype(
```

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

Sample code

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

Model evaluation

In this section, you will evaluate your trained model.

First, examine the metrics for the training job:

In [ ]:
```
sagemaker.analytics.TrainingJobAnalytics(classifier_estimator._current_jo
                                         metric_names = ['test:objective_
                                                         'test:binary_f_b
                                                         'test:precision'
                                                         'test:recall']
                                        ).dataframe()
```

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 [ ]:
```
import io
#bucket='<LabBucketName>'
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, filena
```

In [ ]:
```
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_f

    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(prefi
    target_predicted_df = pd.read_json(io.BytesIO(obj['Body'].read()),ori
    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 [ ]:
```
test_labels, target_predicted = batch_linear_predict(test, classifier_est
```

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

In [ ]:
```
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=colo
    plt.title("Confusion Matrix")
    plt.tight_layout()
    plt.ylabel("True Class")
    plt.xlabel("Predicted Class")
    plt.show()
```

In [ ]:
```
from sklearn import metrics

def plot_roc(test_labels, target_predicted):
    TN, FP, FN, TP = confusion_matrix(test_labels, target_predicted).rave
    # 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_pre

    fpr, tpr, thresholds = metrics.roc_curve(test_labels, target_predicte
    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', col
    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 [ ]:
```
# Enter your code here
from sklearn.metrics import ConfusionMatrixDisplay

ConfusionMatrixDisplay.from_predictions(
    test_labels,
    target_predicted['predicted_label'],
    cmap='Blues'
)
plt.title('Confusion Matrix')
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 [ ]:
```
# Enter your answer here

- The matrix shows the count of correct predictions along the diagonal
(True Positives and True Negatives) and incorrect predictions in the off-
(False Positives and False Negatives).
- The majority of predictions fall on the diagonal, indicating good model
- The model performs better at predicting **no delay** than **delay**, po
- There is still some misclassification, especially for actual delays bei
as no delay, which suggests the need for further tuning or handling of im
```

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

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

### Add indicator variable for holidays
data_orig['is_holiday'] = # Enter your code here
```

Weather data was fetched from https://www.ncei.noaa.gov/access/services/data/v1?dataset=daily-summaries&stations=USW00023174,USW00012960,USW00003017,USW00094846,USV 01-01&endDate=2018-12-31.

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

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 [ ]:
```
weather = pd.read_csv('/home/ec2-user/SageMaker/project/data/daily-summar
station = ['USW00023174','USW00012960','USW00003017','USW00094846','USW00
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 [ ]:
```
weather['MONTH'] = weather['DATE'].apply(lambda x: x.split('-')[1])
weather.head()
```

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 [ ]:
```
import pandas as pd

# Load the dataset first
weather = pd.read_csv("weather.csv")  # Change to your correct file path

# Fill missing snow-related columns
weather['SNOW'].fillna(0, inplace=True)
weather['SNWD'].fillna(0, inplace=True)

# Check remaining missing values
print(weather.isna().sum())
```

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 [ ]:
```
idx = np.array([i for i in range(len(weather))])
TAVG_idx = idx[weather.TAVG.isna()]
TMAX_idx = # Enter your code here
TMIN_idx = # Enter your code here
TAVG_idx
```

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 [ ]:
```
weather_impute = weather.groupby([<CODE>]).agg({'TAVG':'mean','TMAX':'mea
weather_impute.head(2)
```

Merge the mean data with the weather data.

In [ ]:
```
weather = pd.merge(weather, weather_impute,  how='left', left_on=['MONTH'
.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 [ ]:
```
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()
```

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

In [ ]:
```
weather.drop(columns=['STATION','MONTH','TAVG_AVG', 'TMAX_AVG', 'TMIN_AVG
```

Add the origin and destination weather conditions to the dataset.

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

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

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

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

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

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

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


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

In [ ]:
```
data_dummies = pd.get_dummies(data[['Year', 'Quarter', 'Month', 'DayofMon
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 [ ]:
```
data.shape
```

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

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 [ ]:
```
data.rename(columns = {<CODE>:<CODE>}, inplace=True )# Enter your code he
```

Create the training sets again.

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

In [ ]:
```
# Enter your code here
data.rename(columns={
    'DepHour': 'DepHourOfDay',
    'ArrDelay': 'ArrivalDelay'
}, inplace=True)
```

New baseline classifier

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

In [ ]:
```
# Instantiate the LinearLearner estimator object
classifier_estimator2 = # Enter your code here
```

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 [ ]:
```
train_records = classifier_estimator2.record_set(train.values[:, 1:].asty
val_records = classifier_estimator2.record_set(validate.values[:, 1:].ast
test_records = classifier_estimator2.record_set(test.values[:, 1:].astype
```

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

In [ ]:
```
# Enter your code here
classifier_estimator2.fit({'train': train_records, 'validation': val_reco
```

Perform a batch prediction by using the newly trained model.

In [ ]:
```
# Enter your code here
batch_predictor2 = classifier_estimator2.deploy(
    initial_instance_count=1,
    instance_type='ml.m4.xlarge'
)

predictions2 = batch_predictor2.predict(test.values[:, 1:].astype(np.floa
```

Plot a confusion matrix.

In [ ]:
```
# Enter your code here
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

y_true = test.values[:, 0]  # Actual labels
y_pred = [int(pred.label['predicted_label']) for pred in predictions2]

cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

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 [ ]:
```
bucket='c169682a4380827l11227113t1w752528112852-labbucket-gskq1ielbfkg'
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, filena

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

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

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

In [ ]:
```
import boto3
import sagemaker
from sagemaker.image_uris import retrieve

sess = sagemaker.Session()
region = sess.boto_region_name
role = sagemaker.get_execution_role()

container = retrieve(framework='xgboost', region=region, version='1.7-1')

s3_output_location = f"s3://{bucket}/{prefix}/output/"

xgb = sagemaker.estimator.Estimator(
    image_uri=container,
```

```
    role=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,
    objective='binary:logistic',
    eval_metric='auc',
    num_round=100,
)

xgb.fit(inputs=data_channels)
```

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

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

In [ ]:
```
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()
```

Get the predicted target and test labels.

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

test_labels = test.iloc[:,0]

print(target_predicted.head())
```

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

In [ ]:
```
# Enter your code here
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Create confusion matrix
cm = confusion_matrix(test_labels, target_predicted['target'])

# Display confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap='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 [ ]:
```
#Enter your answer here
The confusion matrix shows how well the model predicted delays versus no
If true positives and true negatives are high compared to false predictio
However, if there's a noticeable imbalance (e.g., more false negatives th
it indicates the model struggles with detecting certain classes - possibl
```

Hyperparameter optimization (HPO)

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

### You can spin up multiple instances to do hyperparameter optimization

xgb = sagemaker.estimator.Estimator(container,
                                    role=sagemaker.get_execution_role(),
                                    instance_count= 1, # make sure you ha
                                    instance_type=instance_type,
                                    output_path='s3://{}/{}/output'.forma
                                    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_ty
                         'eta': ContinuousParameter(0.1, 0.5, scaling_typ
                         'min_child_weight': ContinuousParameter(3, 10, s
                         '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 depend
                            max_parallel_jobs=1)
```

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

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

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 [ ]:
```
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(Hyper
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).da
tuner_df.head()
```

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

Tip: See the previous XGBoost estimator fit function.

In [ ]:
```
# Enter your code here'
best_estimator.fit({'train': train_location, 'validation': validation_loc
```

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

In [ ]:
```
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()
```

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

Get the predicted target and test labels.

In [ ]:
```
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_conv

test_labels = test.iloc[:,0]

print(target_predicted.head())
```

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

In [ ]:
```
# Enter your code here
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import numpy as np
import matplotlib.pyplot as plt

# y_true from your test set, y_pred from the binary column you just creat
y_true = np.asarray(test_labels).astype(int)
y_pred = target_predicted['target'].astype(int).to_numpy()

cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['No De
disp.plot(cmap='Blues', values_format='d')
plt.title('Confusion Matrix')
plt.show()
```

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

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 [ ]:
```
# Write your answers here

1. The model performance partially meets the business goal. While it show
for the "delay" class is lower than desired, meaning some delays are stil
tuning time, I would try feature engineering and class balancing to impro

2. Model performance improved after feature selection, hyperparameter tun
and trying different algorithms (e.g., XGBoost vs Linear Learner). The mo
significant gains came from hyperparameter tuning with a wider search spa

3. The biggest challenges were dealing with class imbalance (more "no del
than "delay"), interpreting model outputs to make actionable changes, and
data preprocessing steps were consistent across training and test sets.

4. I still want to explore how much weather-related variables or real-tim
traffic data could improve predictive accuracy. Also, I would like to tes
ensemble models combining multiple approaches.

5. Three most important things I learned:
```

```
   - Proper data preprocessing and feature engineering can dramatically i
   - Hyperparameter tuning is critical for squeezing out better accuracy
   - Model evaluation should go beyond accuracy - recall, precision, and
```

In [130…

```
!df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        7.8G     0  7.8G   0% /dev
tmpfs           7.9G     0  7.9G   0% /dev/shm
tmpfs           7.9G  608K  7.9G   1% /run
tmpfs           7.9G     0  7.9G   0% /sys/fs/cgroup
/dev/xvda1      135G   75G   61G  56% /
/dev/xvdf       4.8G  4.6G     0 100% /home/ec2-user/SageMaker
tmpfs           1.6G     0  1.6G   0% /run/user/1001
tmpfs           1.6G     0  1.6G   0% /run/user/1000
tmpfs           1.6G     0  1.6G   0% /run/user/1002
```