Jeffery Dirden
Professor Johnson
ITAI 1371
November 1, 2024

Challenge Lab

Introduction:

The problem statement involves enhancing the customer experience for a travel booking website by predicting
weather-related flight delays. When customers book flights to or from the busiest domestic airports in the US,
the website aims to notify them if their flight is likely to be delayed due to weather conditions.

Objective:
The objective of the machine learning pipeline is to develop a predictive model that classifies upcoming flights
as either "Delayed" or "On-Time" based on weather data and flight information. By using historical flight data
(including on-time performance and weather conditions), the model will be trained to identify patterns that typically
lead to delays. This prediction can then be provided to customers at booking time, helping them make informed travel decisions
and ultimately improving user satisfaction.

Setup

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

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

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

import pandas as pd
import numpy as np
import subprocess

import matplotlib.pyplot as plt
import seaborn as sns

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

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

```
Matplotlib is building the font cache; this may take a moment.
```

Step 2: Data preprocessing and visualization

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

Specific questions to consider

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

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

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

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

In [2]:
```
# download the files

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

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

```
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_11.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_11.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_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_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_12.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_12.zip
download: s3://aws-tc-largeobjects/CUR-TF-200-ACMLFO-1/flight_delay_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_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_project/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_project/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_project/data/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2014_5.zip to ../project/data/FlightDelays/On_Time_Reporting_Carrier_On_Time_Performance
```

So of course I started by getting all the necessary data imported and uploaded gave everything sometime to get initialized.

Loaded a CSV file before getting ready to combine them. I used On_Time_Reporting_Carrier_On_Time_Performance_(1987_present)_2018_9.csv

Load sample CSV file

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

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

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

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

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

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

Question: Print the first 10 rows of the dataset.

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

In [9]:
```
# Enter your code here
df_temp.head(10)
```

Out[9]:

	Year	Quarter	Month	DayofMonth	DayOfWeek	FlightDate	Reporting_Airline	DOT_ID_Reporting_Airline	IATA_CODE_Reporting_Airline	Tail_Number	...	Div4TailNum	Div5Airport	Div5AirportID	Div5AirportSeqID	Div5WheelsOn	Div5TotalGTime	Div5LongestGTime	Div5Whe
0	2018	3	9	3	1	2018-09-03	9E	20363	9E	N908XJ	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	2018	3	9	9	7	2018-09-09	9E	20363	9E	N315PQ	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	2018	3	9	10	1	2018-09-10	9E	20363	9E	N582CA	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	2018	3	9	13	4	2018-09-13	9E	20363	9E	N292PQ	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	2018	3	9	14	5	2018-09-14	9E	20363	9E	N600LR	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
5	2018	3	9	16	7	2018-09-16	9E	20363	9E	N316PQ	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
6	2018	3	9	17	1	2018-09-17	9E	20363	9E	N916XJ	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
7	2018	3	9	20	4	2018-09-20	9E	20363	9E	N371CA	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
8	2018	3	9	21	5	2018-09-21	9E	20363	9E	N601LR	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
9	2018	3	9	23	7	2018-09-23	9E	20363	9E	N906XJ	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

10 rows × 110 columns

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

```
The #rows and #columns are  585749  and  110
The years in this dataset are:  [2018]
The months covered in this dataset are:  [9]
The date range for data is : 2018-09-01  to  2018-09-30
The airlines covered in this dataset are:  ['9E', 'B6', 'WN', 'YV', 'YX', 'EV', 'AA', 'AS', 'DL', 'HA', 'UA', 'F9', 'G4', 'MQ', 'NK', 'OH', 'OO']
The Orgin airports covered are:  ['DFW', 'LGA', 'MSN', 'MSP', 'ATL', 'BDL', 'VLD', 'JFK', 'RDU', 'CHS', 'DTW', 'GRB', 'PVD', 'SHV', 'FNT', 'PIT', 'RIC', 'RST', 'RSW', 'CVG', 'LIT', 'ORD', 'JAX', 'TRI', 'BOS', 'CWA', 'DCA', 'CHO',
'AVP', 'IND', 'GRR', 'BTR', 'MEM', 'TUL', 'CLE', 'STL', 'BTV', 'OMA', 'MGM', 'TVC', 'SAV', 'GSP', 'EWR', 'OAJ', 'BNA', 'MCI', 'TLH', 'ROC', 'LEX', 'PWM', 'BUF', 'AGS', 'CLT', 'GSO', 'BWI', 'SAT', 'PHL', 'TYS', 'ACK', 'DSM', 'GN
V', 'AVL', 'BGR', 'MHT', 'ILM', 'MOT', 'IAH', 'SBN', 'SYR', 'ORF', 'MKE', 'XNA', 'MSY', 'PBI', 'ABE', 'HPN', 'EVV', 'ALB', 'LNK', 'AUS', 'PHF', 'CHA', 'GTR', 'BMI', 'BQK', 'CID', 'CAK', 'ATW', 'ABY', 'CAE', 'SRQ', 'MLI', 'BHM',
'IAD', 'CSG', 'CMH', 'MCO', 'MBS', 'FLL', 'SDF', 'TPA', 'MVY', 'LAS', 'LGB', 'SFO', 'SAN', 'LAX', 'RNO', 'PDX', 'ANC', 'ABQ', 'SLC', 'DEN', 'PHX', 'OAK', 'SMF', 'SJU', 'SEA', 'HOU', 'STX', 'BUR', 'SWF', 'SJC', 'DAB', 'BQN', 'PS
E', 'ORH', 'HYA', 'STT', 'ONT', 'HRL', 'ICT', 'ISP', 'LBB', 'MAF', 'MDW', 'OKC', 'PNS', 'SNA', 'TUS', 'AMA', 'BOI', 'CRP', 'DAL', 'ECP', 'ELP', 'GEG', 'LFT', 'MFE', 'MDT', 'JAN', 'COS', 'MOB', 'VPS', 'MTJ', 'DRO', 'GPT', 'BFL',
'MRY', 'SBA', 'PSP', 'FSD', 'BRO', 'RAP', 'COU', 'STS', 'PIA', 'FAT', 'SBP', 'FSM', 'HSV', 'BIS', 'DAY', 'BZN', 'MIA', 'EYW', 'MYR', 'HHH', 'GJT', 'FAR', 'SGF', 'HOB', 'CLL', 'LRD', 'AEX', 'ERI', 'MLU', 'LCH', 'ROA', 'LAW', 'MH
K', 'GRK', 'SAF', 'GRI', 'JLN', 'ROW', 'FWA', 'CRW', 'LAN', 'OGG', 'HNL', 'KOA', 'EGE', 'LIH', 'MLB', 'JAC', 'FAI', 'RDM', 'ADQ', 'BET', 'BRW', 'SCC', 'KTN', 'YAK', 'CDV', 'JNU', 'SIT', 'PSG', 'WRG', 'OME', 'OTZ', 'ADK', 'FCA',
'FAY', 'PSC', 'BIL', 'MSO', 'ITO', 'PPG', 'MFR', 'EUG', 'GUM', 'SPN', 'DLH', 'TTN', 'BKG', 'SFB', 'PIE', 'PGD', 'AZA', 'SMX', 'RFD', 'SCK', 'OWB', 'HTS', 'BLV', 'IAG', 'USA', 'GFK', 'BLI', 'ELM', 'PBG', 'LCK', 'GTF', 'OGD', 'ID
A', 'PVU', 'TOL', 'PSM', 'CKB', 'HGR', 'SPI', 'STC', 'ACT', 'TYR', 'ABI', 'AZO', 'CMI', 'BPT', 'GCK', 'MQT', 'ALO', 'TXK', 'SPS', 'SWO', 'DBQ', 'SUX', 'SJT', 'GGG', 'LSE', 'LBE', 'ACY', 'LYH', 'PGV', 'HVN', 'EWN', 'DHN', 'PIH',
'IMT', 'WYS', 'CPR', 'SCE', 'HLN', 'SUN', 'ISN', 'CMX', 'EAU', 'LWB', 'SHD', 'LBF', 'HYS', 'SLN', 'EAR', 'VEL', 'CNY', 'GCC', 'RKS', 'PUB', 'LBL', 'MKG', 'PAH', 'CGI', 'UIN', 'BFF', 'DVL', 'JMS', 'LAR', 'SGU', 'PRC', 'ASE', 'RD
D', 'ACV', 'OTH', 'COD', 'LWS', 'ABR', 'APN', 'ESC', 'PLN', 'BJI', 'BRD', 'BTM', 'CDC', 'CIU', 'EKO', 'TWF', 'HIB', 'BGM', 'RHI', 'ITH', 'INL', 'FLG', 'YUM', 'MEI', 'PIB', 'HDN']
The Destination airports covered are:  ['CVG', 'PWM', 'RDU', 'MSP', 'MSN', 'SHV', 'CLT', 'PIT', 'RIC', 'IAH', 'ATL', 'JFK', 'DCA', 'DTW', 'LGA', 'TYS', 'PVD', 'FNT', 'LIT', 'BUF', 'ORD', 'TRI', 'IND', 'BGR', 'AVP', 'BWI', 'LEX',
'BDL', 'GRR', 'CWA', 'TUL', 'MEM', 'AGS', 'EWR', 'MGM', 'PHL', 'SYR', 'OMA', 'STL', 'TVC', 'ORF', 'CLE', 'ABY', 'BOS', 'OAJ', 'TLH', 'BTR', 'SAT', 'JAX', 'BNA', 'CHO', 'VLD', 'ROC', 'DFW', 'GNV', 'ACK', 'PBI', 'CHS', 'GRB', 'MO
T', 'MKE', 'DSM', 'ILM', 'GSO', 'MCI', 'SBN', 'BTV', 'MVY', 'XNA', 'RST', 'EVV', 'HPN', 'RSW', 'MDT', 'ROA', 'GSP', 'MCO', 'CSG', 'SAV', 'PHF', 'ALB', 'CHA', 'ABE', 'BMI', 'MSY', 'IAD', 'GTR', 'CID', 'CAK', 'ATW', 'AUS', 'BQK',
'MLI', 'CAE', 'CMH', 'AVL', 'MBS', 'FLL', 'SDF', 'TPA', 'LNK', 'SRQ', 'MHT', 'BHM', 'LAS', 'SFO', 'SAN', 'RNO', 'LGB', 'ANC', 'PDX', 'SJU', 'ABQ', 'SLC', 'DEN', 'LAX', 'PHX', 'OAK', 'SMF', 'SEA', 'STX', 'BUR', 'DAB', 'SJC', 'SW
F', 'HOU', 'BQN', 'PSE', 'ORH', 'HYA', 'STT', 'ONT', 'DAL', 'ECP', 'ELP', 'HRL', 'MAF', 'MDW', 'OKC', 'PNS', 'SNA', 'AMA', 'BOI', 'GEG', 'ICT', 'LBB', 'TUS', 'ISP', 'CRP', 'MFE', 'LFT', 'VPS', 'JAN', 'COS', 'MOB', 'DRO', 'GPT',
'BFL', 'COU', 'SBP', 'MTJ', 'SBA', 'PSP', 'FSD', 'FSM', 'BRO', 'PIA', 'STS', 'FAT', 'RAP', 'MRY', 'HSV', 'BIS', 'DAY', 'BZN', 'MIA', 'EYW', 'MYR', 'HHH', 'GJT', 'FAR', 'MLU', 'LRD', 'CLL', 'LCH', 'FWA', 'GRK', 'SGF', 'HOB', 'LA
W', 'MHK', 'SAF', 'JLN', 'ROW', 'GRI', 'AEX', 'CRW', 'LAN', 'ERI', 'HNL', 'KOA', 'OGG', 'EGE', 'LIH', 'JAC', 'MLB', 'RDM', 'BET', 'ADQ', 'BRW', 'SCC', 'FAI', 'JNU', 'CDV', 'YAK', 'SIT', 'KTN', 'WRG', 'PSG', 'OME', 'OTZ', 'ADK',
'FCA', 'BIL', 'PSC', 'FAY', 'MSO', 'ITO', 'PPG', 'MFR', 'DLH', 'EUG', 'GUM', 'SPN', 'TTN', 'BKG', 'AZA', 'SFB', 'LCK', 'BLI', 'SCK', 'PIE', 'RFD', 'PVU', 'PBG', 'BLV', 'PGD', 'SPI', 'USA', 'TOL', 'IDA', 'ELM', 'HTS', 'HGR', 'SM
X', 'OGD', 'GFK', 'STC', 'GTF', 'IAG', 'CKB', 'OWB', 'PSM', 'ABI', 'TYR', 'ALO', 'SUX', 'AZO', 'ACT', 'CMI', 'BPT', 'TXK', 'SWO', 'SPS', 'DBQ', 'SJT', 'GGG', 'LSE', 'MQT', 'GCK', 'LBE', 'ACY', 'LYH', 'PGV', 'HVN', 'EWN', 'DHN',
'PIH', 'WYS', 'SCE', 'IMT', 'HLN', 'ASE', 'SUN', 'ISN', 'EAR', 'SGU', 'VEL', 'SHD', 'LWB', 'MKG', 'SLN', 'HYS', 'BFF', 'PUB', 'LBL', 'CMX', 'EAU', 'PAH', 'UIN', 'RKS', 'CGI', 'CNY', 'JMS', 'DVL', 'LAR', 'GCC', 'LBF', 'PRC', 'RD
D', 'ACV', 'OTH', 'COD', 'LWS', 'ABR', 'APN', 'PLN', 'BJI', 'CPR', 'BRD', 'BTM', 'CDC', 'CIU', 'ESC', 'EKO', 'ITH', 'HIB', 'BGM', 'TWF', 'RHI', 'INL', 'FLG', 'YUM', 'MEI', 'PIB', 'HDN']
```

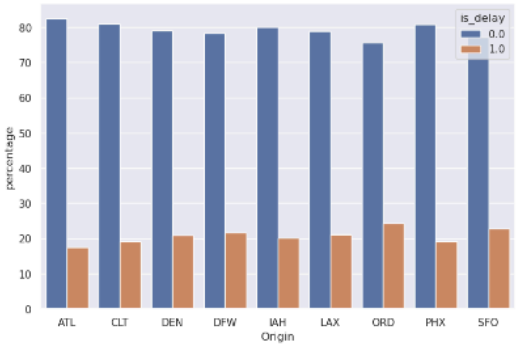
Question: What is the count of all the origin and destination airports?

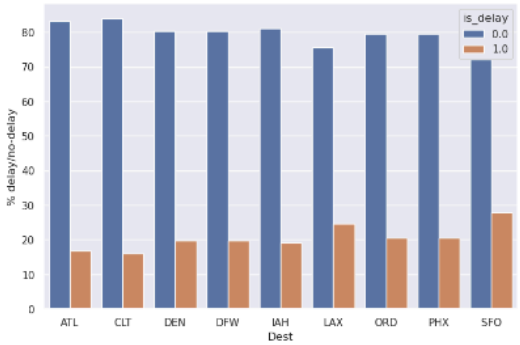
Hint: To find the values for each airport by using the **Origin** and **Dest** columns, you can use the `values_count` function in pandas (pandas.Series.value_counts documentation).

This part here gave me some issues I was inserting wrong classes, and variables and kept getting error messages. I had to learn the correct verbiage from the dataset to use in order for it to run correctly.

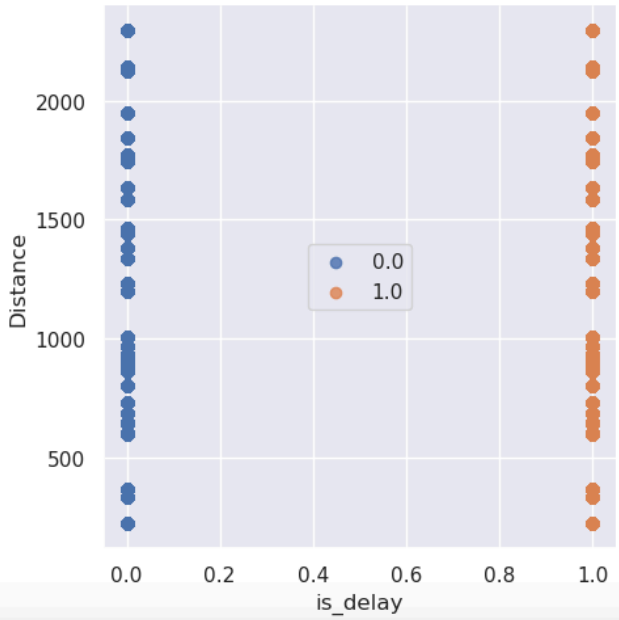
In [43]:
```
(data.groupby('is_delay').size()/len(data) ).plot(kind='bar')# Enter your code here
plt.ylabel('Frequency')
plt.title('Distribution of classes')
plt.show()
```

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





In [45]:
```
sns.lmplot( x="is_delay", y="Distance", data=data, fit_reg=False, hue='is_delay', legend=False)
plt.legend(loc='center')
plt.xlabel('is_delay')
plt.ylabel('Distance')
plt.show()
```



In [49]:
```
data_dummies = pd.get_dummies(data[['Quarter', 'Month', 'DayofMonth', 'DayOfWeek', 'Reporting_Airline', 'O
data_dummies = data_dummies.replace({True: 1, False: 0})
data = pd.concat([data, data_dummies], axis = 1)
data.drop(categorical_columns,axis=1, inplace=True)
```

Check the length of the dataset and the new columns.

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

In [50]:
```
# Enter your code here
data.head()
```

Out[50]:

	is_delay	Distance	Quarter_2	Quarter_3	Quarter_4	Month_2	Month_3	Month_4	Month_5	Month_6	...	DepHourofDay_14
0	0.0	337.0	0	1	0	0	0	0	0	0	...	0
1	0.0	1635.0	0	1	0	0	0	0	0	0	...	0
2	0.0	888.0	0	1	0	0	0	0	0	0	...	0
3	0.0	1379.0	0	1	0	0	0	0	0	0	...	0
4	0.0	862.0	0	1	0	0	0	0	0	0	...	0

5 rows × 94 columns

In [51]:
```
# Enter your code here
data.shape
```

Out[51]:
```
(1635590, 94)
```

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 [54]:
```
data.rename(columns = {'is_delay':'target'}, inplace=True )# Enter your code here
```

End of Step 2

Heres when things started to pick up starting with train-test split and actually giving my model some datasets to train was fun.

Train-test split

In [55]:
```
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['targe
    test, validate = train_test_split(test_and_validate, test_size=0.5, random_state=42, stratify=test_and
    return train, validate, test
```

In [56]:
```
train, validate, test = split_data(data)
print(train['target'].value_counts())
print(test['target'].value_counts())
print(validate['target'].value_counts())
```

```
0.0    1033806
1.0     274666
Name: target, dtype: int64
0.0    129226
1.0     34333
Name: target, dtype: int64
0.0    129226
1.0     34333
Name: target, 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 [60]:
```
### Fit the classifier
# Enter your code here
classifier_estimator.fit([train_records,val_records,test_records])
```

```
INFO:sagemaker.image_uris:Same images used for training and inference. Defaulting to image scope: inferen
ce.
INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
INFO:sagemaker:Creating training-job with name: linear-learner-2024-11-04-00-35-08-187
2024-11-04 00:35:09 Starting - Starting the training job...
2024-11-04 00:35:36 Starting - Preparing the instances for training.....
2024-11-04 00:36:22 Downloading - Downloading input data...
2024-11-04 00:37:08 Downloading - Downloading the training image.........
2024-11-04 00:38:19 Training - Training image download completed. Training in progress.Docker entrypoint
called with argument(s): train
Running default environment configuration script
[11/04/2024 00:38:35 INFO 140618102482752] Reading default configuration from /opt/amazon/lib/python3.8/s
ite-packages/algorithm/resources/default-input.json: {'mini_batch_size': '1000', 'epochs': '15', 'feature
_dim': 'auto', 'use_bias': 'true', 'binary_classifier_model_selection_criteria': 'accuracy', 'f_beta':
'1.0', 'target_recall': '0.8', 'target_precision': '0.8', 'num_models': 'auto', 'num_calibration_sample
s': '10000000', 'init_method': 'uniform', 'init_scale': '0.07', 'init_sigma': '0.01', 'init_bias': '0.0',
'optimizer': 'auto', 'loss': 'auto', 'margin': '1.0', 'quantile': '0.5', 'loss_insensitivity': '0.01', 'h
uber_delta': '1.0', 'num_classes': '1', 'accuracy_top_k': '3', 'wd': 'auto', 'l1': 'auto', 'momentum': 'a
uto', 'learning_rate': 'auto', 'beta_1': 'auto', 'beta_2': 'auto', 'bias_lr_mult': 'auto', 'bias_wd_mul
t': 'auto', 'use_lr_scheduler': 'true', 'lr_scheduler_step': 'auto', 'lr_scheduler_factor': 'auto', 'lr_s
cheduler_minimum_lr': 'auto', 'positive_example_weight_mult': '1.0', 'balance_multiclass_weights': 'fals
e', 'normalize_data': 'true', 'normalize_label': 'auto', 'unbias_data': 'auto', 'unbias_label': 'auto',
'num_point_for_scaler': '10000', '_kvstore': 'auto', '_num_gpus': 'auto', '_num_kv_servers': 'auto', '_lo
g_level': 'info', '_tuning_objective_metric': '', 'early_stopping_patience': '3', 'early_stopping_toleran
ce': '0.001', '_enable_profiler': 'false'}
[11/04/2024 00:38:35 INFO 140618102482752] Merging with provided configuration from /opt/ml/input/config/
hyperparameters.json: {'binary_classifier_model_selection_criteria': 'cross_entropy_loss', 'feature_dim':
'93', 'mini_batch_size': '1000', 'predictor_type': 'binary_classifier'}
[11/04/2024 00:38:35 INFO 140618102482752] Final configuration: {'mini_batch_size': '1000', 'epochs': '1
```

Having to find the LabBucketName was very interesting I really had no idea what it was about or what to do I did some extensive research to understand how to access the S3 panel to access it and how to apply it, I guess its all geographic based.

```
import io
# Having to find the LabBucketName was very interesting
bucket='sagemaker-us-east-1-905418072867'
prefix='flight-linear'
train_file='flight_train.csv'
test_file='flight_test.csv'
validate_file='flight_validate.csv'
whole_file='flight.csv'
s3_resource = boto3.Session().resource('s3')

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

```
INFO:botocore.credentials:Found credentials from IAM Role: BaseNotebookInstanceEc2InstanceRole
```

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

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

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

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

    classifier_transformer.wait()

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

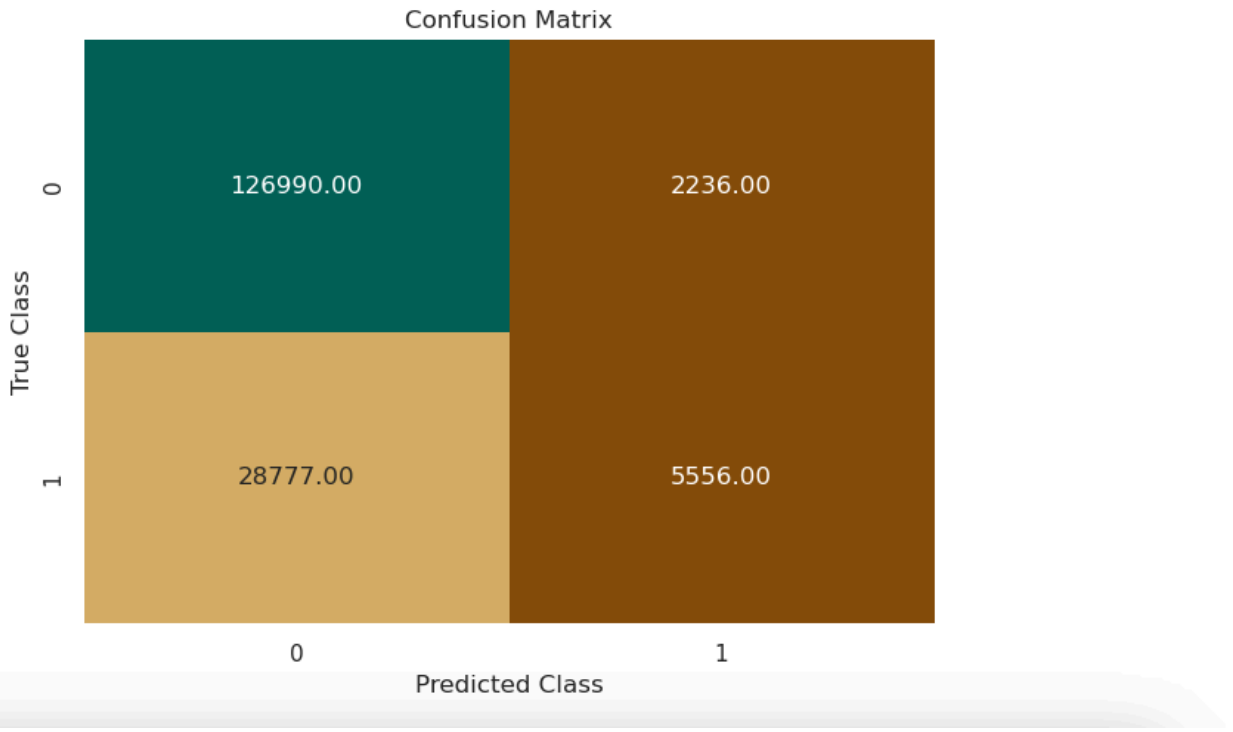
So from here I basically just ran some predictions, and created a confusion matrix as instructed to analyze some data of the test_labels & targer_predicted.

```
test_labels, target_predicted = batch_linear_predict(test, classifier_estimator)
```

```
INFO:sagemaker.image_uris:Same images used for training and inference. Defaulting to image scope: inferen
ce.
INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
INFO:sagemaker:Creating model with name: linear-learner-2024-11-04-00-44-22-307
INFO:sagemaker:Creating transform job with name: linear-learner-2024-11-04-00-44-22-994
.....................................Docker entrypoint called with argument(s): serve
Running default environment configuration script
[11/04/2024 00:51:06 INFO 139940816336704] Memory profiler is not enabled by the environment variable ENA
BLE_PROFILER.
/opt/amazon/lib/python3.8/site-packages/mxnet/model.py:97: SyntaxWarning: "is" with a literal. Did you me
an "=="?
  if num_device is 1 and 'dist' not in kvstore:
/opt/amazon/lib/python3.8/site-packages/scipy/optimize/_shgo.py:495: SyntaxWarning: "is" with a literal.
Did you mean "=="?
  if cons['type'] is 'ineq':
/opt/amazon/lib/python3.8/site-packages/scipy/optimize/_shgo.py:743: SyntaxWarning: "is not" with a liter
al. Did you mean "!="?
  if len(self.X_min) is not 0:
[11/04/2024 00:51:10 WARNING 139940816336704] Loggers have already been setup.
[11/04/2024 00:51:10 INFO 139940816336704] loaded entry point class algorithm.serve.server_config:config_
api
[11/04/2024 00:51:10 INFO 139940816336704] loading entry points
[11/04/2024 00:51:10 INFO 139940816336704] loaded request iterator application/json
[11/04/2024 00:51:10 INFO 139940816336704] loaded request iterator application/jsonlines
[11/04/2024 00:51:10 INFO 139940816336704] loaded request iterator application/x-recordio-protobuf
[11/04/2024 00:51:10 INFO 139940816336704] loaded request iterator text/csv
[11/04/2024 00:51:10 INFO 139940816336704] loaded response encoder application/json
[11/04/2024 00:51:10 INFO 139940816336704] loaded response encoder application/jsonlines
[11/04/2024 00:51:10 WARNING 139940816336704] Loggers have already been setup.
[11/04/2024 00:51:10 INFO 139940816336704] loaded entry point class algorithm.serve.server_config:config_
api
[11/04/2024 00:51:10 INFO 139940816336704] loading entry points
[11/04/2024 00:51:10 INFO 139940816336704] loaded request iterator application/json
[11/04/2024 00:51:10 INFO 139940816336704] loaded request iterator application/jsonlines
[11/04/2024 00:51:10 INFO 139940816336704] loaded request iterator application/x-recordio-protobuf
[11/04/2024 00:51:10 INFO 139940816336704] loaded request iterator text/csv
[11/04/2024 00:51:10 INFO 139940816336704] loaded response encoder application/json
```

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

```
# Enter your code here
plot_confusion_matrix(test_labels, target_predicted)
```



Hyperparameter optimization (HPO)

```
from sagemaker.tuner import IntegerParameter, CategoricalParameter, ContinuousParameter, HyperparameterTun

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

xgb = sagemaker.estimator.Estimator(container,
                                    role=sagemaker.get_execution_role(),
                                    instance_count= 1, # make sure you have a limit set for these instance
                                    instance_type=instance_type,
                                    output_path='s3://{}/{}/output'.format(bucket, prefix),
                                    sagemaker_session=sess)

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

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

objective_metric_name = 'validation:auc'

tuner = HyperparameterTuner(xgb,
                            objective_metric_name,
                            hyperparameter_ranges,
                            max_jobs=10, # Set this to 10 or above depending upon budget and available tim
                            max_parallel_jobs=1)
```

```
tuner.fit(inputs=data_channels)
tuner.wait()
```

```
WARNING:sagemaker.estimator:No finished training job found associated with this estimator. Please make su
re this estimator is only used for building workflow config
WARNING:sagemaker.estimator:No finished training job found associated with this estimator. Please make su
re this estimator is only used for building workflow config
INFO:sagemaker:Creating hyperparameter tuning job with name: sagemaker-xgboost-241104-0136
.....................................................................................................
!
```

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

```
# Enter your code here
best_estimator.fit(inputs=data_channels)
```

```
INFO:sagemaker:Creating training-job with name: sagemaker-xgboost-2024-11-04-02-39-36-851
2024-11-04 02:39:38 Starting - Starting the training job...
2024-11-04 02:39:51 Starting - Preparing the instances for training...
2024-11-04 02:40:18 Downloading - Downloading input data...
2024-11-04 02:40:54 Downloading - Downloading the training image......
2024-11-04 02:41:44 Training - Training image download completed. Training in progress.[2024-11-04 02:41:
53.419 ip-10-2-95-24.ec2.internal:7 INFO utils.py:27] RULE_JOB_STOP_SIGNAL_FILENAME: None
INFO:sagemaker-containers:Imported framework sagemaker_xgboost_container.training
INFO:sagemaker-containers:Failed to parse hyperparameter _tuning_objective_metric value validation:auc to
Json.
Returning the value itself
INFO:sagemaker-containers:Failed to parse hyperparameter eval_metric value auc to Json.
Returning the value itself
INFO:sagemaker-containers:Failed to parse hyperparameter objective value binary:logistic to Json.
Returning the value itself
INFO:sagemaker-containers:No GPUs detected (normal if no gpus installed)
INFO:sagemaker_xgboost_container.training:Running XGBoost Sagemaker in algorithm mode
INFO:root:Determined delimiter of CSV input is ','
INFO:root:Determined delimiter of CSV input is ','
INFO:root:Determined delimiter of CSV input is ','
INFO:root:Determined delimiter of CSV input is ','
INFO:root:Single node training.
INFO:root:Setting up HPO optimized metric to be : auc
[02:41:58] 1308472x85 matrix with 111220120 entries loaded from /opt/ml/input/data/train?format=csv&label
_column=0&delimiter=,
[02:41:58] 163559x85 matrix with 13902515 entries loaded from /opt/ml/input/data/validation?format=csv&la
bel_column=0&delimiter=,
[2024-11-04 02:41:58.642 ip-10-2-95-24.ec2.internal:7 INFO json_config.py:91] Creating hook from json_con
fig at /opt/ml/input/config/debughookconfig.json.
[2024-11-04 02:41:58.643 ip-10-2-95-24.ec2.internal:7 INFO hook.py:201] tensorboard_dir has not been set
for the hook. SMDebug will not be exporting tensorboard summaries.
[2024-11-04 02:41:58.643 ip-10-2-95-24.ec2.internal:7 INFO profiler_config_parser.py:102] User has disabl
```

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

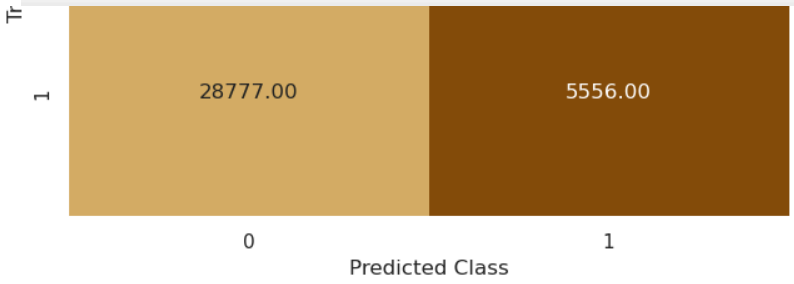
Get the predicted target and test labels.

```
print(target_predicted.head())

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

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

test_labels = test.iloc[:,0]

print(target_predicted.head())
```

```
     target
0  0.092488
1  0.090602
2  0.601977
3  0.049336
4  0.117867
   target
0       0
1       0
2       1
3       0
4       0
```

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

```
# Enter your code here
plot_confusion_matrix(test_labels, target_predicted)
```



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.

```
]: # Write your answers here
   # I do feel like the model performance meets the business goal.
   # Tried to keep it as simple possible ran into alot of issues on things I feel I shouldn't
   # The biggest challenge I faced was classifying the data correctly
   # Not to sure why I couldn't train my three datasets and peform a batch on them
   # The biggest thing I learned is correctly structuring and gaining more understanding how to decipher the
```

Data Collection
We gathered historical data on domestic flights, including on-time performance metrics and weather conditions for the busiest US airports. This dataset contained essential details such as departure/arrival times, origin and destination airports, and delay status, as well as weather metrics like temperature, precipitation, and wind speed.

Data Preprocessing

- Data Cleaning: Missing values in crucial fields like weather data and delay status were handled through imputation or removal.
- Feature Engineering: Key features, including weather conditions, time of day, and month, were created to capture influential factors affecting delays.
- Encoding: Categorical variables (e.g., airport codes, airlines) were one-hot encoded to make them usable for the ML model.
- Data Splitting: The data was divided into training, validation, and test sets to ensure effective model evaluation.

Model Selection
Several classification models, including logistic regression, decision trees, and gradient boosting, were evaluated. The random forest classifier was ultimately chosen due to its high accuracy and feature interpretability.

```
# Setting up SageMaker Estimator for model training
import sagemaker
from sagemaker.sklearn import SKLearn

sagemaker_session = sagemaker.Session()
sklearn_estimator = SKLearn(entry_point='train.py',  # Training script
                            framework_version='0.23-1',
                            instance_type='ml.m5.large',
                            role='your_sagemaker_execution_role')
sklearn_estimator.fit({'train': train_data, 'validation': val_data})
```

Model Training
The model was trained on Amazon SageMaker's managed infrastructure. The validation set helped fine-tune hyperparameters and mitigate overfitting. Key metrics, such as accuracy, precision, and recall, were used to assess performance.

Deployment
The trained model was deployed as an endpoint on SageMaker for real-time predictions. It processes incoming flight and weather data to predict delays at booking time.

```
# Deploying the model

predictor = sklearn_estimator.deploy(initial_instance_count=1,
instance_type='ml.m4.xlarge')
```