Flight Status Data Analysis and Classification

Caleb McCurdy, Trevor Sauerbrey, and UE Wang

502 Data Mining, University of San Diego

Background

For those who live in the United States, flying is one of the main transportation options. People may discover that the likelihood of a flight being delayed or canceled varies depending on the airport you depart from, the time you fly, the airline you use, etc.

We will utilize the EDA approach to learn each factor's impact on the flight and the classification method to predict the flight status in order to gain a better understanding of the factors connected to flight cancellation and delays.

Inventory of resources

Data Source: Kaggle, Flight Status Prediction. (Kaggle, 2022)

Software Access: Jupyter Notebook, Google Docs, Google Slides

Data Visualization Resources: Jupyter Notebook packages like seaborn, and matplotlib will be used to deliver intuitive graphs for this project.

Data Collection

Kaggle's data set includes flight logs from 2018 through July 2022. The spread of COVID-19 (CDC, 2020) began in late 2019, causing each state to implement a variety of travel restrictions and measures to prevent and reduce the spread. To ensure that the data is consistent, we will take the data from a year in advance.

19 of the 61 variables in the dataset will be used in our study. There was a reduction in data storage from 2.5 GB to 787 MB. We will also include an additional 5 columns in order to more effectively categorize the variables required for our project. Our final model will only use 6 predictive variables for our 2 target features.

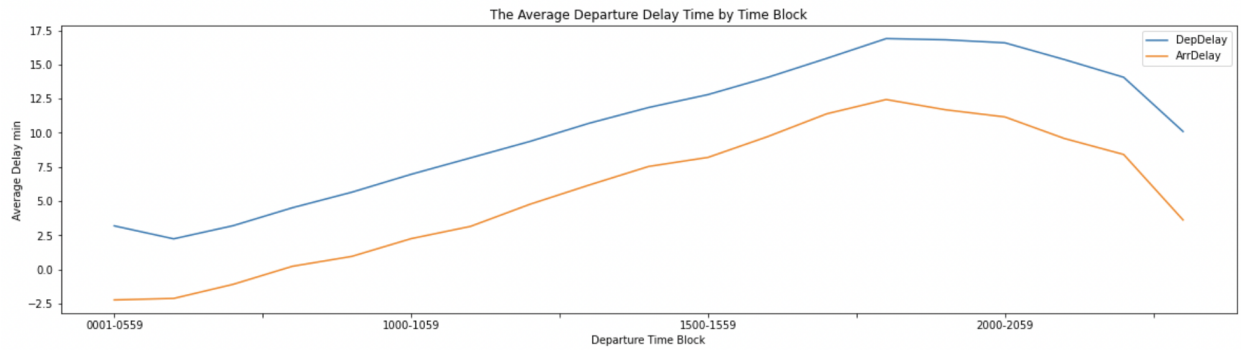
Data Description/Observations

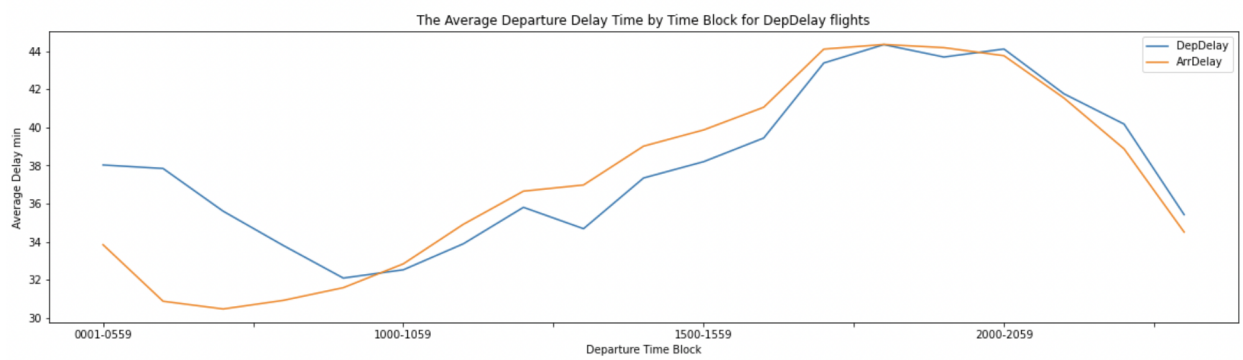
1. Departure Delay & Time Block

The flights that have been canceled would not have the exact time block or a minute-long departure delay. We will not include any data with missing values and only study the delayed data from departure in this section.

By comparing the median delay minutes for all flights versus the median delay minutes for flights that are delayed, we can clearly see that arriving flights from midnight to nine o’clock in the morning may land earlier than the scheduled time. In general, the departure delays are longer than the delayed arrivals. The amount of time that planes are delayed can always be reduced by

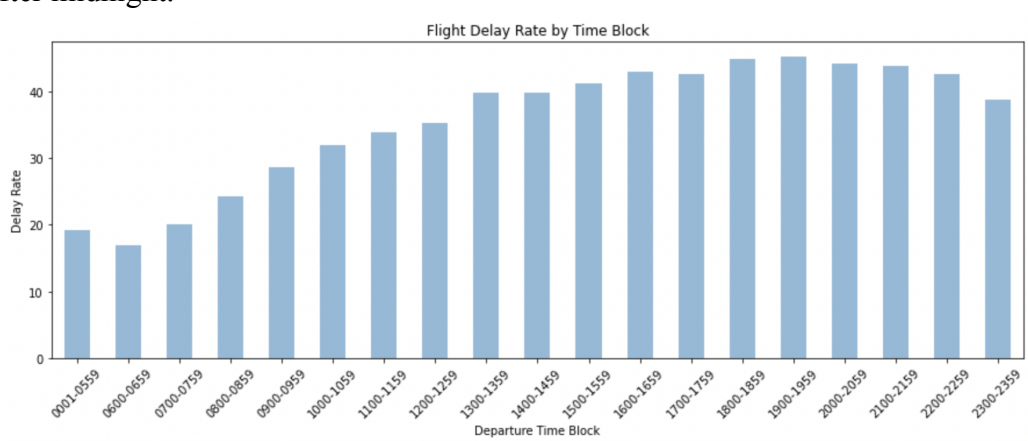
“making up time” in the air, however, the likelihood of delay reduction is greatest between midnight and 10 am then decreases after that. Between 3 PM and 12 AM, the average airplane departure delay is over 10 minutes. At all times, the average departure delay is more than 30 minutes. After 3 PM, the trend trails off, returning to the high-rise stage at midnight.





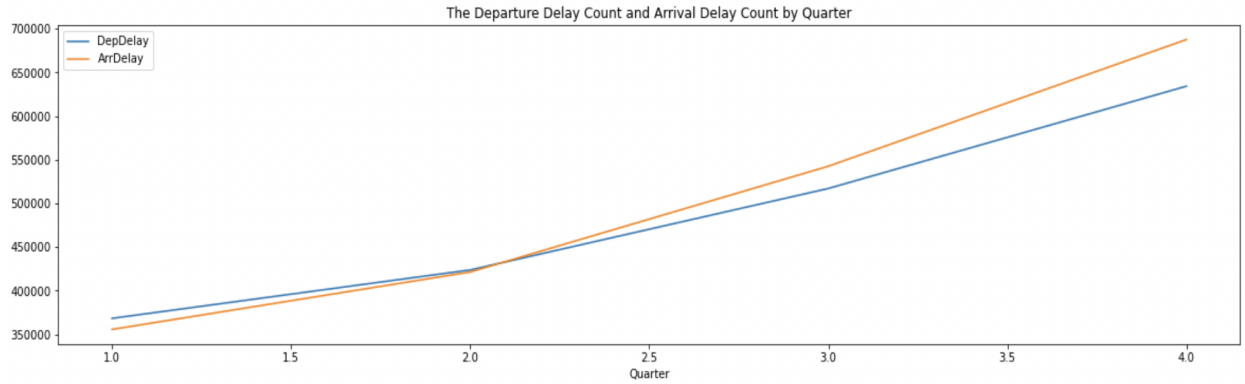
2. Departure Delay Rate by Time Block

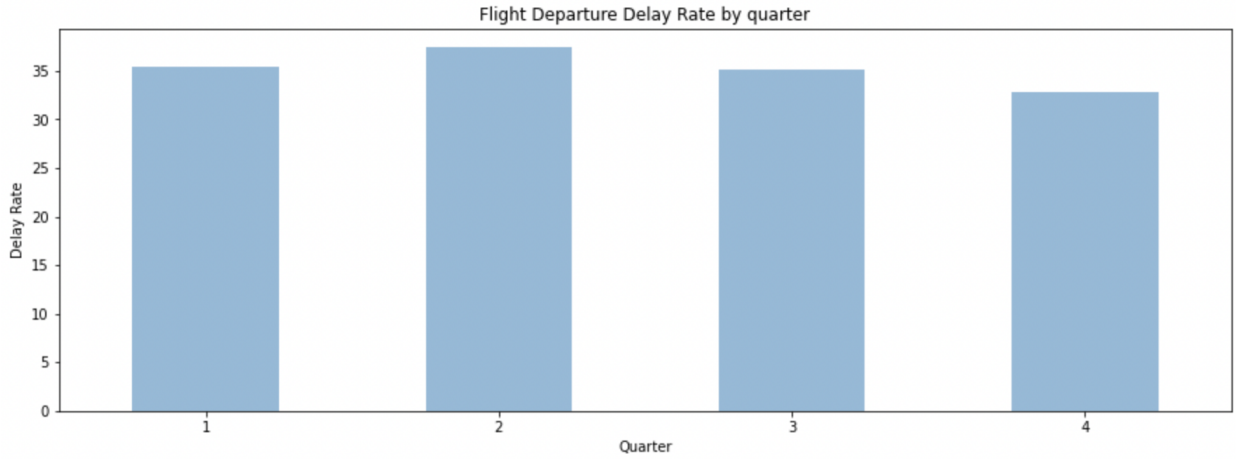
We can determine that there is consistently a countrywide probability of flight delays of more than 15% by dividing the number of delayed flights by the total number of flights and plotting the results according to the time block. The delay rate will reach 30% after 10 AM and continue to rise until 7 PM. The number of overnight flights is extremely low, thus there will be fewer delays after midnight.



3. Quarter

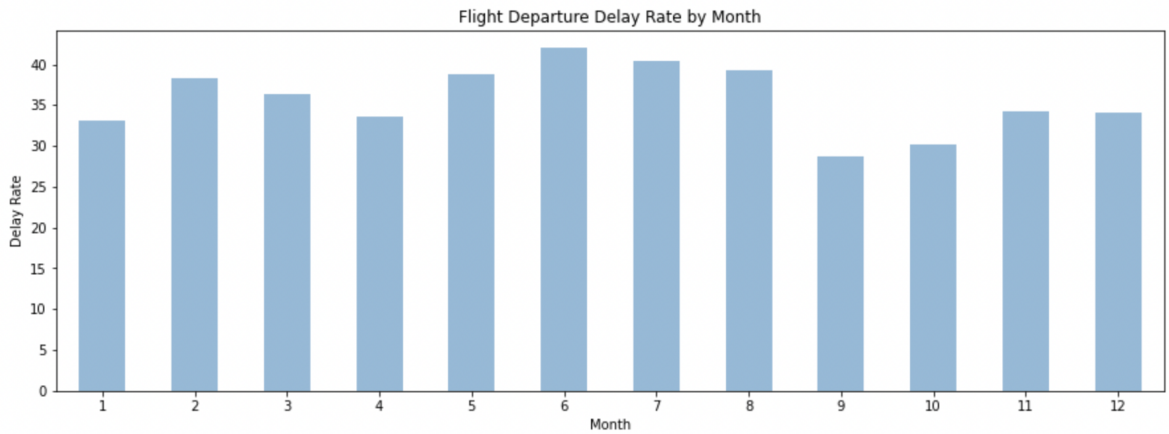
The number of flights varies according to the seasons. To determine whether the delay rate would differ between quarters, we will use the provided variable "Quarter". First, we count the number of flights that have been delayed based on seasonality. It is not surprising that there are more flight delays in the second part of the year. However, if we divide the number of flights with delays by the total number of flights for each quarter, we can observe that the rate of delays has reduced. Despite the fact that there were more holidays in the third and fourth quarters, there were more flights but lower delay rates.



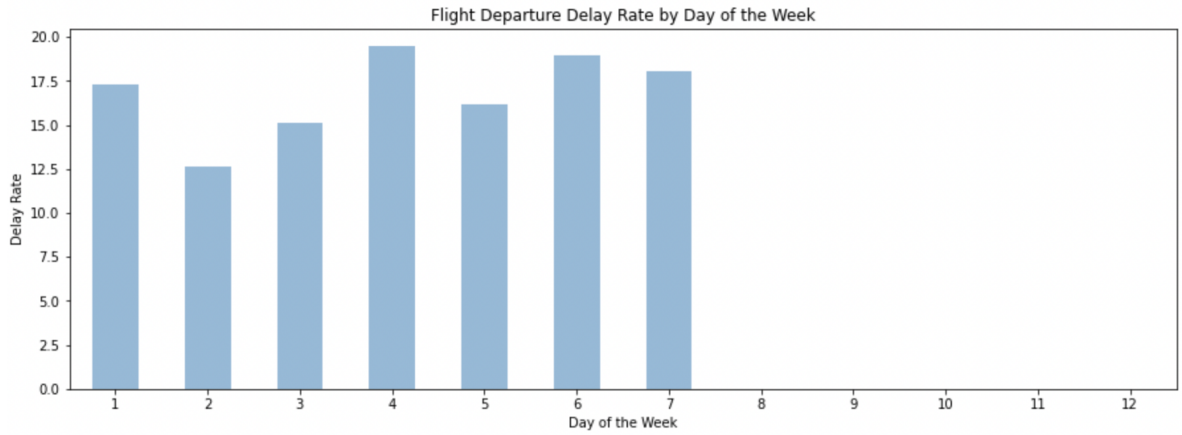


4. Month, Day of the Week, and Day of the Month

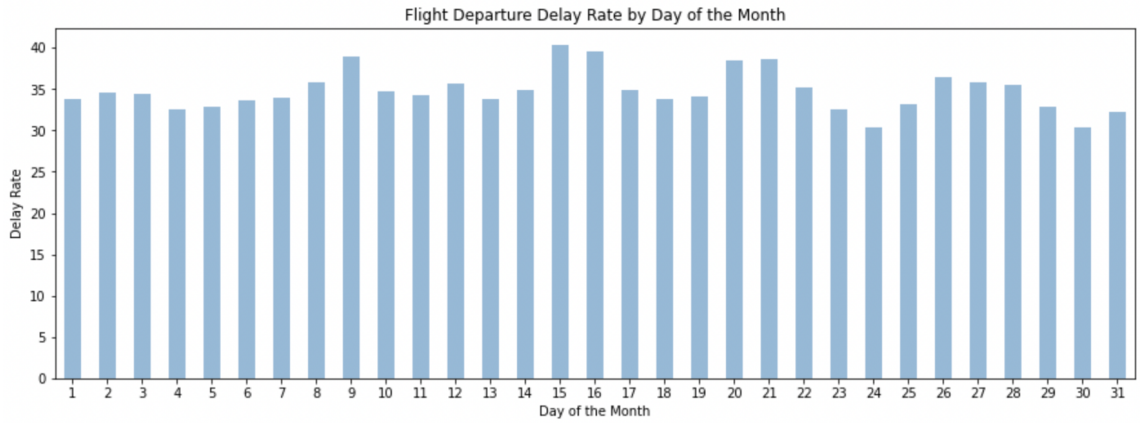
We will examine the monthly delay rate to determine if there have been any changes as each month has a different amount of holidays. According to the graph below that illustrates the flight departure delay rate by month, September has the lowest delay rate of the entire year. The outcome is consistent with our analysis of the last variable, which showed that the fourth quarter has the lowest delay rate. All three of the second quarter's months have the top three daily rates of the whole year. The overall quarter-by-quarter delay rate is greater than 25%.



The month is then divided into its days of the week, as seen below, where Tuesday has the week's lowest delay rate. Because people tend to travel more on Thursdays than on Mondays, there is an increase in the rate of airplane departure delays. Since the delay rate varies by the day of the week, this factor is good to contain in our modeling.

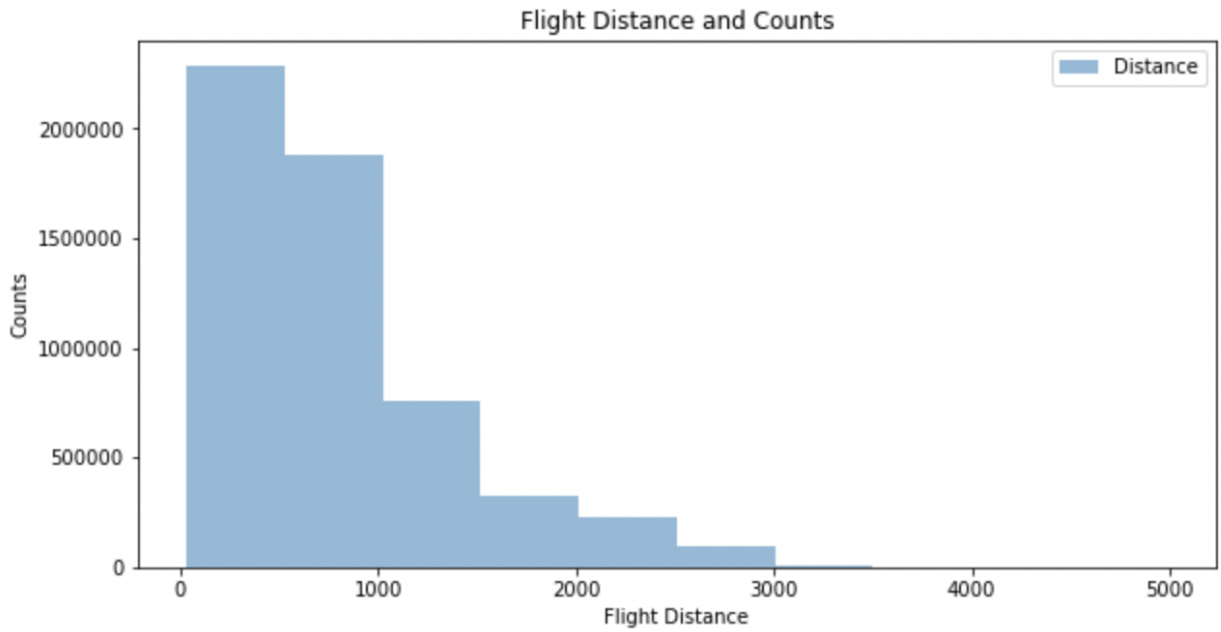


We utilize the departure delay flights divided by the total flights per month to determine which day of the month has the greatest delay rate. Unfortunately, there is no clear distinction between the days, and since the holidays and days of each month are not all constant, we will consider the day of the month component to be variable year-to-year and avoid using it in the modeling stage.



5. Distance

According to flight distance counts, the majority of flights are under 1000 miles. Domestic flights of lengths greater than 1500 miles are quite rare. When a flight is delayed, the flight's distance also is taken into consideration because it might shorten the delay time in the air. To categorize the distance by levels of 300 miles, we add the Distance_Level variable. Additionally, categorize the arrival and departure delays by 15-minute intervals. We then analyze each level independently to determine whether the airtime has an impact on the delay time.



Fifty-one and a half percent (50.5%) of the "DepDelay 15min" group on delayed flights can still arrive ahead of schedule. Forty-one and a half percent (40.5%) of flights in the 16–30 minute delay group can arrive late by no more than 15 minutes. Fifty-seven and a half percent (57.5%) of the flights in the depDelay 31-60 minutes group still arrive late by the same amount of time. Almost eighty-eight percent (87.8%) of flights that left with a delay of more than 60 minutes still arrive with a delay of more than 60 minutes. According to the table below, there is a good chance that we will arrive ahead of schedule as long as the departure delay is within 30 minutes. However, we will consider the distance element in order to examine it further.

ArrDelay_level	ArrDelay 16-30 min	ArrDelay 31-60 min	ArrDelay < 15 min	ArrDelay > 61	Early Arrival < 0
DepDelay_level					
DepDelay 16-30 min	36.630000	12.490000	40.470000	1.640000	8.770000
DepDelay 31-60 min	26.920000	57.550000	5.270000	9.750000	0.510000
DepDelay < 15 min	8.540000	2.650000	36.860000	0.450000	51.500000
DepDelay > 61	0.200000	11.940000	0.020000	87.840000	0.000000

We choose the data that shows flights arrive early with departure delays of no more than 15 minutes. The actual average arrival time for this block of flight is actually in advance of the scheduled arrival time. The more distance, the more time a flight will arrive ahead of schedule. The "Distance < 300-mile" group, for example, has an average departure delay of 4.6 minutes but an average arrival delay of -6 minutes, which suggests that even with an average departure delay of 4.6 minutes, the trip time can be cut by 10 minutes when the flight distance is within 300 miles. Comparatively, the average departure delay on flights longer than 2700 miles is 5.5 minutes, but the whole travel time is cut by 18 minutes.

	Flight_counts	Avg_DepDelay min	Avg_ArrDelay min
Distance 300-600 mile	122183	4.744375	-6.249274
Distance 600-900 mile	99231	5.098104	-7.705818
Distance 900-1200 mile	81472	5.315998	-8.817876
Distance < 300	59909	4.682034	-5.944516
Distance 1200-1500 mile	35710	5.385690	-10.021395
Distance 1500-1800 mile	33938	5.411780	-10.970181
Distance 2400-2700 mile	15605	5.511246	-13.176546
Distance 1800-2100 mile	15182	5.506982	-11.415821
Distance 2100-2400 mile	14618	5.489602	-12.384389
Distance > 2700 mile	2943	5.504247	-12.791030

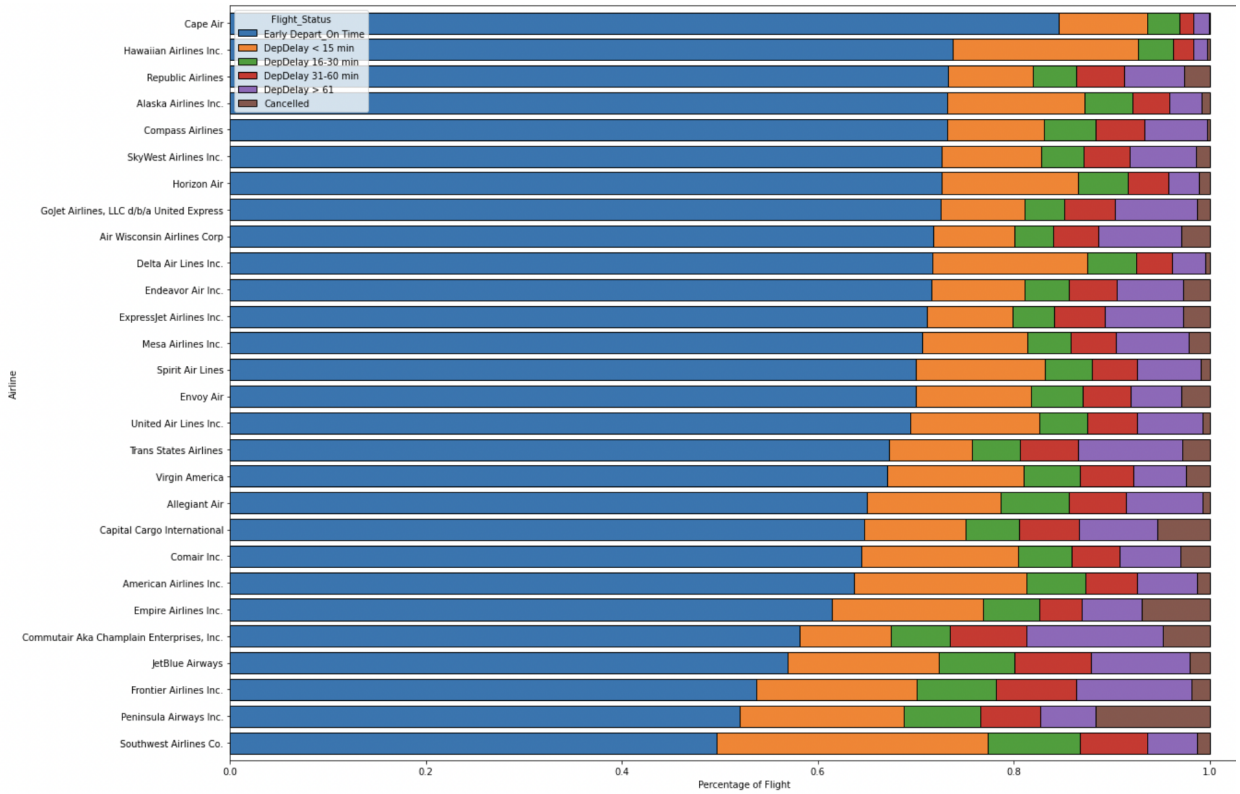
This same phenomenon remains consistent when observing the other departure delay blocks as well.

6. Airline

Another element that can help determine whether a flight will be delayed or canceled is the airline hosting the flight. People frequently assume that low-cost airlines have poor on-time performance; nevertheless, we can calculate their delayed rate by dividing the total number of delayed flights by the total number of flights for each airline. The decreasing chart below demonstrates that, out of all 28 airlines, Southwest Airlines performs the worst, with a delay rate over 50%. Cape Air had the best performance of any airline, with a delay rate of only 15.23% during the entire year.

Airline	Airline_delay Counts	Airline_Total Counts	Rate
Southwest Airlines Co.	660551	1331609	49.605477
Frontier Airlines Inc.	53187	117523	45.256673
JetBlue Airways	124396	297758	41.777551
Peninsula Airways Inc.	456	1121	40.677966
Commutair Aka Champlain Enterprises, Inc.	17658	45509	38.801116
Empire Airlines Inc.	519	1411	36.782424
American Airlines Inc.	133816	378722	35.333569
Allegiant Air	32787	95192	34.443020
Comair Inc.	37928	113211	33.502045
Capital Cargo International	13165	41818	31.481659
Virgin America	5346	17153	31.166560
Trans States Airlines	19862	64664	30.715700
United Air Lines Inc.	184285	614915	29.969183
Spirit Air Lines	50951	174129	29.260491
Delta Air Lines Inc.	128432	458840	27.990585
Envoy Air	34305	123437	27.791505
Mesa Airlines Inc.	41453	149840	27.664842
ExpressJet Airlines Inc.	43341	161955	26.761137
Compass Airlines	11886	44680	26.602507
Horizon Air	22174	83936	26.417747
GoJet Airlines, LLC d/b/a United Express	16482	62437	26.397809
Endeavor Air Inc.	30470	115924	26.284462
SkyWest Airlines Inc.	136931	522086	26.227671
Alaska Airlines Inc.	47753	182705	26.136668
Hawaiian Airlines Inc.	21714	83375	26.043778
Air Wisconsin Airlines Corp	24816	95565	25.967666
Republic Airlines	48630	197448	24.629269
Cape Air	252	1655	15.226586

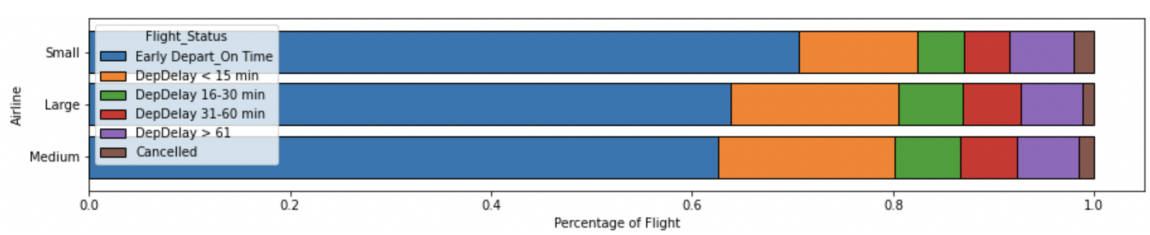
We also wanted to look into the different levels of delays and cancellations for each of these airlines which can be shown in the following figure. Despite having the largest delay rate, Southwest Airlines does not experience as many large delays or canceled flights when compared to other airlines, suggesting that short delays account for a big portion of flights for Southwest Airlines.



7. Airport_Size

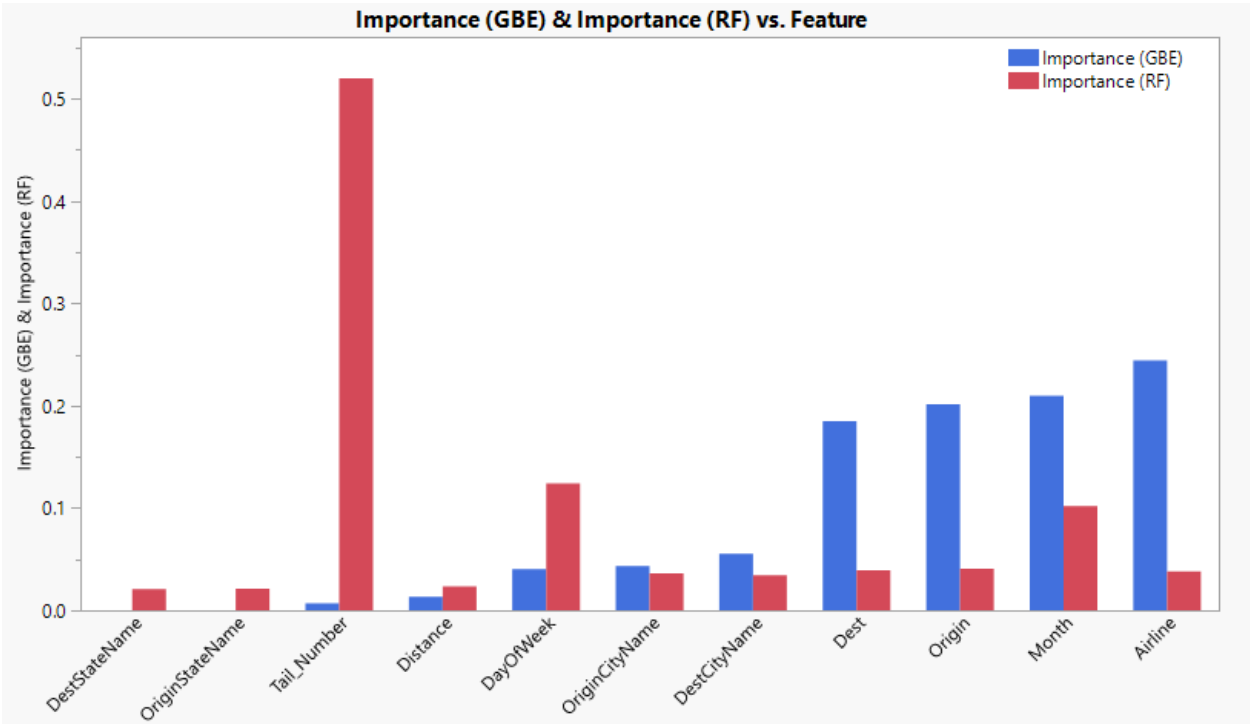
People commonly believe that busier airports experience more delays. We divide the airports into three categories: small, medium, and large. A small airport is one with less than 50 daily flights. We refer to airports as medium if an airport experiences between 50 and 500 flights per day. Large airports are categorized as those with over 500 daily flights. The small airports appear to have a somewhat lower delay rate, whereas the medium-sized and large-sized airports have delay rates that are nearly identical when we use the number of delayed flights and the total flights of each airport.

Airport_Size	Origin_Delay Counts	Origin_Total Counts	Rate
Medium	1354572	3719842	36.414772
Large	331191	934772	35.430137
Small	257783	924004	27.898472



Data Reduction

Once we finished our data description and observations of variables that we believed to be important, we performed some data cleaning and reduction by getting rid of variables that were redundant, did not appear to apply for our model, or ones with low importance. For example, the airport IDs were redundant with the airport codes as they are referring to the same place. Departure delays would not play a role in the canceled flights as these are mutually exclusive events.



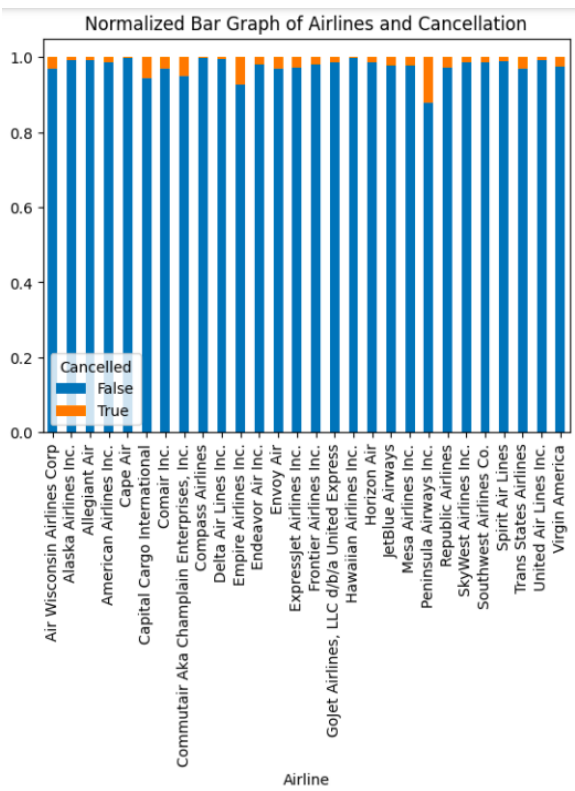
After running feature importance, we are able to further reduce our dataset to only include the following predictive variables: month, day of week, origin, destination, airline, and distance. Tail number is a variable that we considered keeping around, but because of its abnormal importance behavior and linkage to only the individual airplane, we have decided to scrap it. The city and state names did not have as high of feature importance as the airport codes (origin and dest) while being redundant so we scrapped these variables as well. From our new reduced columns, we can explore the relationships between the predictive variables to each of the target features.

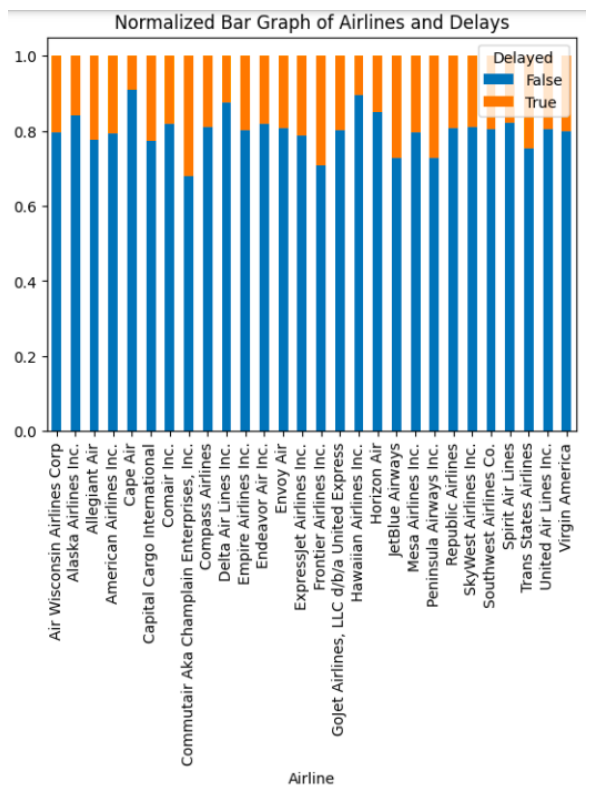
Data Exploration

The target features for this project are the presence, or absence, of flight cancellation and flight delay. Thus, data exploration should be taken on how other features relate to these target features. When looking at our six predictive variables, three of them are of type object. We want to look at their summary statistics, as done in the following figure.

	Airline	Origin	Dest
count	5689512	5689512	5689512
unique	28	370	370
top	Southwest Airlines Co.	ORD	ORD
freq	1352552	289451	289405

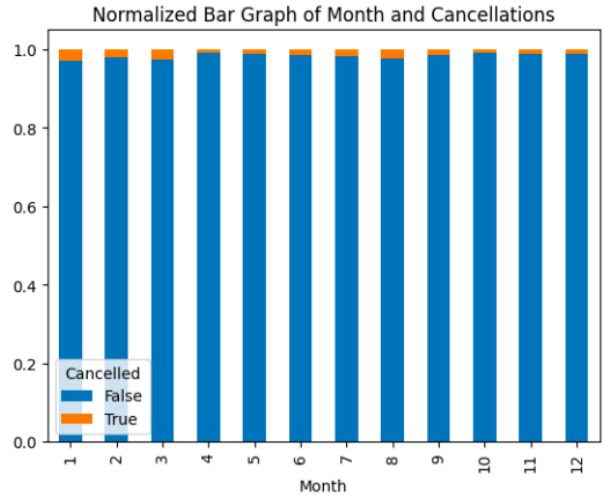
Because of the large number of unique classes for the origin and destination variables, a visualization of them would be unreadable and so we will not provide one. For the airline variable, however, we created a normalized bar graph with overlay for each of the cancellation and delay target features:

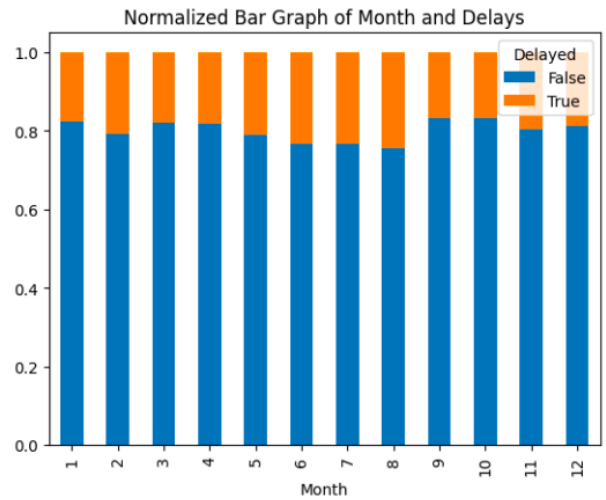




As illustrated above, flight delays give us a much easier relationship to visualize with the predictive variable than cancellations do. With both delays and cancellations, we are able to see how some airlines experience these phenomenons at a higher rate than others. Peninsula Airways Inc. experiences cancellations at the greatest rate in this data set, while Commutair Aka Champlain Enterprises Inc. experiences delays at the greatest rate.

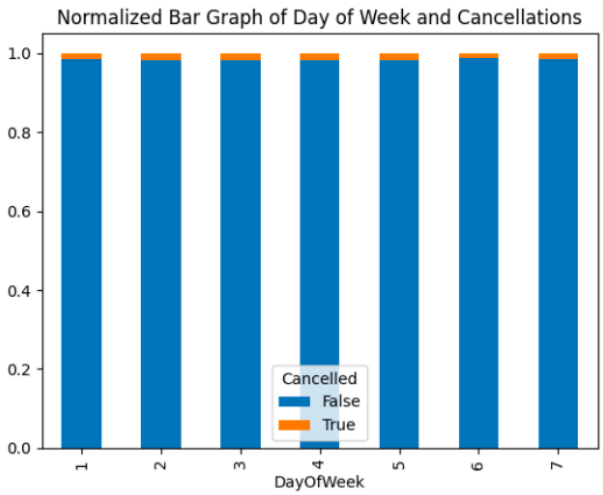
Moving onto the other predictive variables, we want to look at how the month, day of the week, and flight distance affect the cancellation and delay rates. First, we do this for the month which has 12 unique classes (January through December, labeled as 1-12):

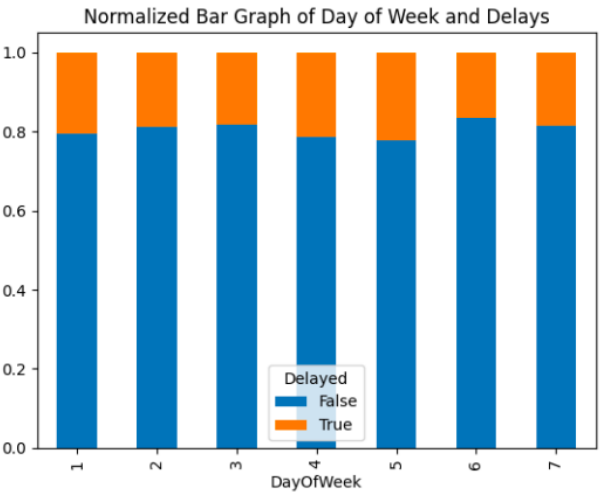




Just as with the airlines, we get a much better sense of a relationship with the month for the delays than the cancellations. However, we are able to notice some relationships still. Months at the beginning of the year appear to experience more cancellations than other months, save for August. This could be for many reasons, such as weather differences but we will not speculate further as to why these relationships exist. For delays, the months near the summer appear to have more delays. Once again we will not speculate as to why this is true, but it is interesting to see different months topping the delay rates than we saw for cancellation rates.

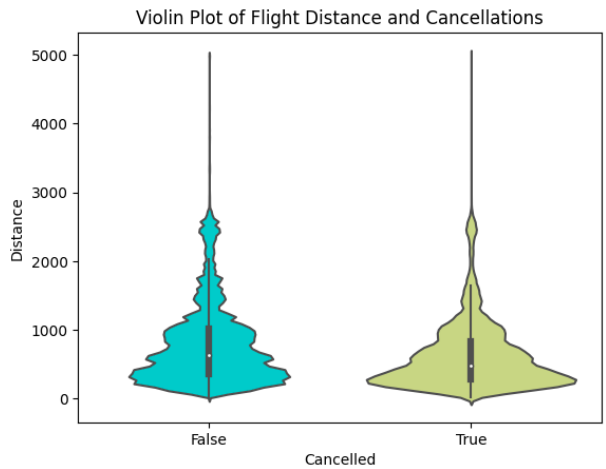
Next, we will look at the days of the week as it relates to our target features. We have created normalized bar graphs for this:

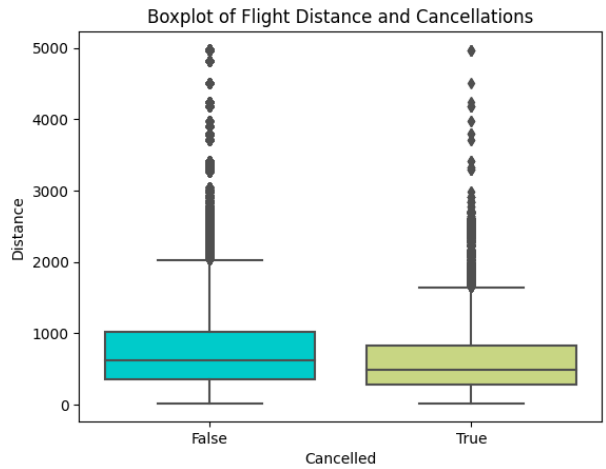




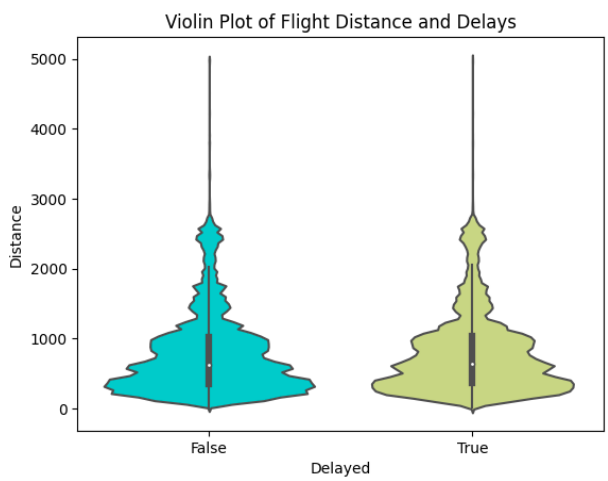
Day of the week does not have as distinguishable of results as month did. However, we are still able to gain some information from the visualizations. The sixth day of the week appears to have the lowest rate of cancellations and delays. Day 5 experiences the highest rate of delays as well as one of the highest cancellation rates.

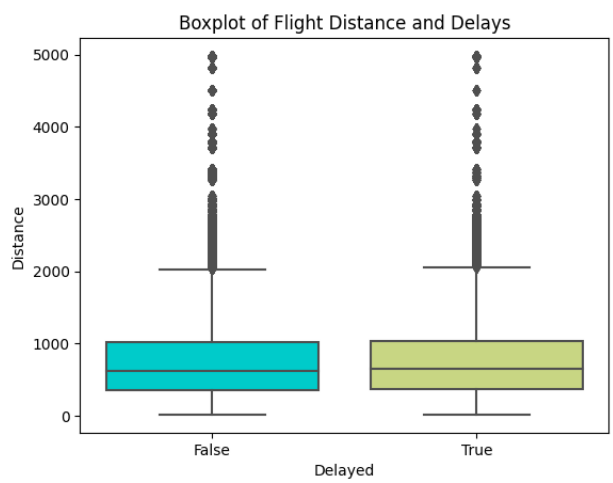
Finally, we look into how the flight distance relates to delay and cancellation. We will start with how distance relates to cancellations, using a violin plot and boxplot:





From these two plots, we are able to see that cancellations have more associativity with shorter flights than longer ones. This is why the density of the cancellations is greater near the bottom of the violin as well as the interquartile range of the boxplot being lower. The non-cancelled flights have a wider stem near the top of its colored section due to its higher density in longer flights. Next, we look at the relationship between distance and flight delays, once again using a violin plot and boxplot:





Unlike for the cancellations, it does not appear that the distance tells us much as it relates to flight delays. Both the violin plots and boxplots look almost identical for delays and non-delays.

The densities are very similar for each distance, and the interquartile ranges are at the same levels.

This concludes the exploratory data analysis section of our project for the relationships between predictive variables and our target features.

Data Classification & Prediction

Dependant (Target) variable 1: Canceled Flights

For our first model, we use the aforementioned predictive variables to help classify flights as canceled (true) or not (false). Unlike for delay times, there is no threshold to consider for flight cancellations, the flight is either canceled or it happens. For this model, we use all the flights in our data set as they can all be classified according to our target variable.

Random Forest Classifier results from Python H20 module:

Reported on training data (Random Forest):

```
ModelMetricsBinomial: drf
** Reported on train data. **

MSE: 0.011830237320947356
RMSE: 0.10876689441621175
LogLoss: 0.0869265743777181
Mean Per-Class Error: 0.3958186218872365
AUC: 0.8184418693768559
AUCPR: 0.06234795075579152
Gini: 0.6368837387537118
```

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.09709883430855945

	False	True	Error	Rate
False	4349146.0	131592.0	0.0294	(131592.0/4480738.0)
True	41472.0	12934.0	0.7623	(41472.0/54406.0)
Total	4390618.0	144526.0	0.0382	(173064.0/4535144.0)

Here is an evaluation of the random forest classifier on the canceled training data set in Python H2O. The accuracy metric for this is 96.2%.

Reported on test data (Random Forest):

```
ModelMetricsBinomial: drf
** Reported on validation data. **

MSE: 0.011633441153490584
RMSE: 0.10785843107282149
LogLoss: 0.06545000136265723
Mean Per-Class Error: 0.380140206213011
AUC: 0.8365075381885055
AUCPR: 0.06478836745411508
Gini: 0.6730150763770111
```

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.08785157345500601

	False	True	Error	Rate
False	1082225.0	38156.0	0.0341	(38156.0/1120381.0)
True	9759.0	3679.0	0.7262	(9759.0/13438.0)
Total	1091984.0	41835.0	0.0423	(47915.0/1133819.0)

We validated our model by running the classifier on the test data set as well and compared the results. Here is an evaluation of the random forest classifier on the canceled test data set in Python H2O. The accuracy metric for this is 95.8% which is only slightly lower than our training data set accuracy, but not different enough to suggest overfitting.

Gradient Boosting Estimator results from Python H20 module:

Reported on training data (Gradient Boosting Estimator):

```
ModelMetricsBinomial: gbm
** Reported on train data. **

MSE: 0.011474057389676558
RMSE: 0.10711702660957574
LogLoss: 0.05565988290421229
Mean Per-Class Error: 0.4099988799947512
AUC: 0.8278987873137784
AUCPR: 0.06385882556769851
Gini: 0.6557975746275568
```

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.05347271502732791

	False	True	Error	Rate
False	4372274.0	108692.0	0.0243	(108692.0/4480966.0)
True	43126.0	11070.0	0.7957	(43126.0/54196.0)
Total	4415400.0	119762.0	0.0335	(151818.0/4535162.0)

Here is an evaluation of the gradient boosting estimator on the canceled training data set in Python H2O. The accuracy metric for this is 96.7%.

Reported on test data (Gradient Boosting Estimator):

```
ModelMetricsBinomial: gbm
** Reported on validation data. **

MSE: 0.01159053579104582
RMSE: 0.10765935069024808
LogLoss: 0.05655401947934198
Mean Per-Class Error: 0.41899350976288574
AUC: 0.8183887266070857
AUCPR: 0.05885153270841317
Gini: 0.6367774532141715
```

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.054927692010963355

	False	True	Error	Rate
False	1094231.0	25923.0	0.0231	(25923.0/1120154.0)
True	11121.0	2527.0	0.8148	(11121.0/13648.0)
Total	1105352.0	28450.0	0.0327	(37044.0/1133802.0)

We once again validated our model by running the estimator on the test data set as well and compared results. Here is an evaluation of the gradient boost estimator on the canceled test data set in Python H2O. The accuracy metric for this is also 96.7% which is the same as for the training data set.

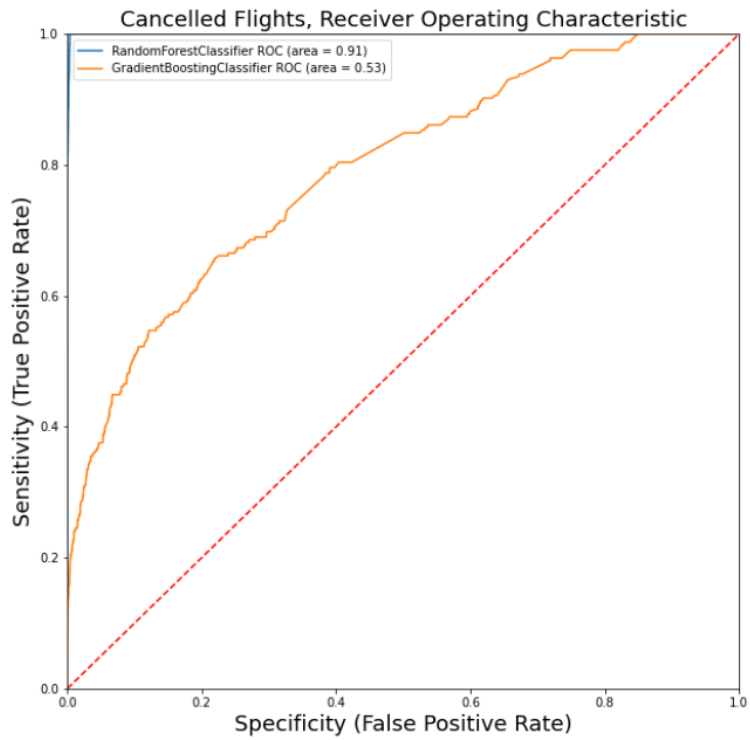
Cross-Validation Metrics Summary:

	mean	
accuracy	0.9667488	0
auc	0.8177446	0
err	0.0332512	0
err_count	30162.8	
f0point5	0.0990282	0
f1	0.1198966	0
f2	0.1528436	0
lift_top_group	9.561763	0
logloss	0.0562802	0
max_per_class_error	0.8112214	0
mcc	0.1135862	0
mean_per_class_accuracy	0.5824683	0
mean_per_class_error	0.4175317	0
mse	0.0115079	0
pr_auc	0.0591155	0
precision	0.0888289	0
r2	0.0253604	0
recall	0.1887786	0
rmse	0.1072747	0
specificity	0.976158	0

Above are some cross-validation metrics for the Python H2O module of flight cancellation.

Results from scikit learn using one-hot encoding:

We provide an ROC curve for each of the classification methods that we have discussed for canceled flights as well as some of the key evaluation metrics.



Classifier	Accuracy	F1 Score	ROC	Precision
RandomForestClassifier	99.7	0.878	0.904	0.961
GradientBoostingClassifier	98.8	0.129	0.535	0.895

Classifier	Recall	Log Loss	Sensitivity	Specificity
RandomForestClassifier	0.808	0.095	1.0	0.808
GradientBoostingClassifier	0.069	0.397	1.0	0.069

Based on the ROC curve, we believe the random forest classifier to be the better performer on predicting the cancellation of flights in 2018.

Dependant (Target) variable 2: Delayed Flights

We decided to use the flight's arrival delay time as our target variable for the second model instead of the flight's departure delay time since the customers we chose to inform are the flight attendees, or the passengers. Thinking from their perspective, we chose arrival delay time as that has a larger impact on people's schedules as compared to departure delay times. The departure delay time becomes irrelevant as long as the plane arrives at its destination at the right time. We used an arrival delay time of 15 minutes as the threshold to discern whether or not a flight was delayed (1) or on time (0). For this model, we only use the flights from our data set that were not canceled as we intend to classify the occurring flights as delayed or not.

Random Forest Classifier results from Python H20 module:

Random Forest (train data):

```
ModelMetricsBinomial: drf
** Reported on train data. **

MSE: 0.14493263834478887
RMSE: 0.3807001948315615
LogLoss: 0.5988064001937393
Mean Per-Class Error: 0.4095890657533847
AUC: 0.6214257221313118
AUCPR: 0.2469121924640666
Gini: 0.2428514442626235
```

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.1564336197319548

	False	True	Error	Rate
False	397705.0	297044.0	0.4276	(297044.0/694749.0)
True	56394.0	87607.0	0.3916	(56394.0/144001.0)
Total	454099.0	384651.0	0.4214	(353438.0/838750.0)

Here is an evaluation of the random forest classifier on the delayed training data set in Python H2O. The accuracy metric for this is 57.9%.

Random Forest (test data):

```
ModelMetricsBinomial: drf
** Reported on validation data. **

MSE: 0.14297237923154965
RMSE: 0.37811688567366264
LogLoss: 0.47309042367324583
Mean Per-Class Error: 0.40686839478203 44
AUC: 0.6266494099034042
AUCPR: 0.2506573958451317
Gini: 0.25329881980680846
```

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.1623066155576723

	False	True	Error	Rate
False	101856.0	71951.0	0.414	(71951.0/173807.0)
True	14358.0	21558.0	0.3998	(14358.0/35916.0)
Total	116214.0	93509.0	0.4115	(86309.0/209723.0)

We validated our model by running the classifier on the test data set as well and compared the results. Here is an evaluation of the random forest classifier on the delayed test data set in Python H2O. The accuracy metric for this is 58.8% which is only slightly lower than our training data set accuracy, but not different enough to suggest overfitting.

Random Forest Performance Metrics Summary:

Cross-Validation Metrics Summary:

	mean
accuracy	0.5764105
auc	0.6213890
err	0.4235895
err_count	71065.8
f0point5	0.2589384
f1	0.3298907
f2	0.4544230
lift_top_group	2.0566883
logloss	0.4854471
max_per_class_error	0.4299811
mcc	0.1341623
mean_per_class_accuracy	0.5886558
mean_per_class_error	0.4113442
mse	0.1449850
pr_auc	0.2468159
precision	0.2264679
r2	-0.0195098
recall	0.6072926
rmse	0.3807687
specificity	0.5700189

Above are some cross-validation metrics for the Python H2O module of flight delays using the random forest classifier.

Gradient Boosting Estimator results from Python H20 module:

Gradient Boosting Estimator (train data):

```
ModelMetricsBinomial: gbm
** Reported on train data. **

MSE: 0.1360686821627974
RMSE: 0.3688748868692438
LogLoss: 0.4375848994841443
Mean Per-Class Error: 0.390772258156763
AUC: 0.6533083807869204
AUCPR: 0.2776297135725935
Gini: 0.3066167615738409
```

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.17681683707189025

	False	True	Error	Rate
False	438915.0	255915.0	0.3683	(255915.0/694830.0)
True	59514.0	84507.0	0.4132	(59514.0/144021.0)
Total	498429.0	340422.0	0.376	(315429.0/838851.0)

Here is an evaluation of the gradient boosting estimator on the canceled training data set in Python H2O. The accuracy metric for this is 62.4%.

Gradient Boosting Estimator (test data):

```
ModelMetricsBinomial: gbm
** Reported on validation data. **

MSE: 0.136640835455121
RMSE: 0.36964961173403255
LogLoss: 0.43970547467355137
Mean Per-Class Error: 0.39827057101386426
AUC: 0.6424396357877957
AUCPR: 0.2656605475735208
Gini: 0.28487927157559145
```

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.17432689654396077

	False	True	Error	Rate
False	106974.0	66833.0	0.3845	(66833.0/173807.0)
True	14798.0	21118.0	0.412	(14798.0/35916.0)
Total	121772.0	87951.0	0.3892	(81631.0/209723.0)

We once again validated our model by running the estimator on the test data set as well and compared results. Here is an evaluation of the gradient boost estimator on the canceled test data set in Python H2O. The accuracy metric for this is also 61.1% which is the same as for the training data set.

Gradient Boosting Estimator Performance Metrics Summary:

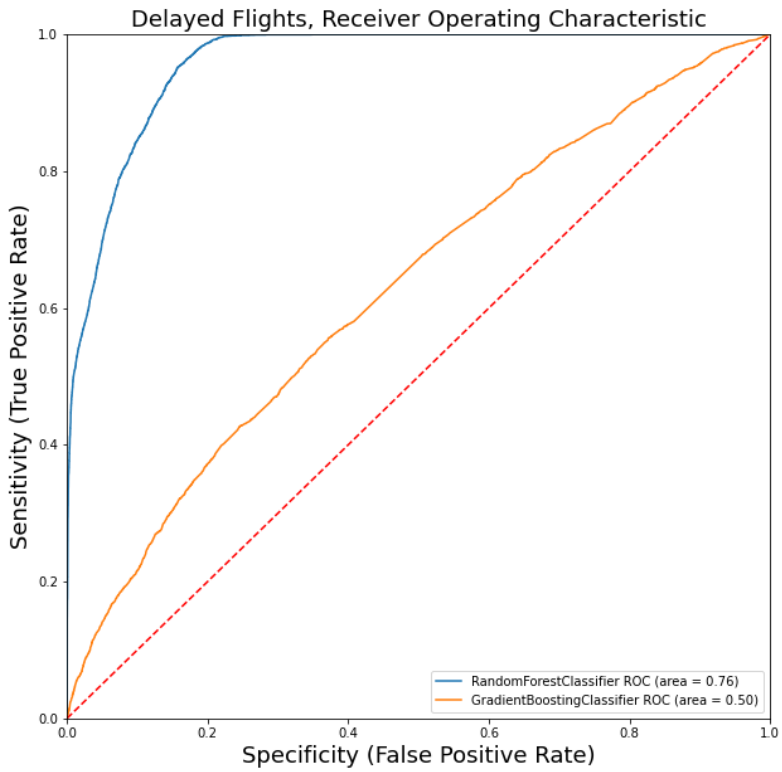
Cross-Validation Metrics Summary

	mean
accuracy	0.6021152
auc	0.6406967
err	0.3978849
err_count	66750.0
f0point5	0.2711391
f1	0.3411351
f2	0.4600439
lift_top_group	2.3808634
logloss	0.4407735
max_per_class_error	0.4158868
mcc	0.1542209
mean_per_class_accuracy	0.6011753
mean_per_class_error	0.3988247
mse	0.137028
pr_auc	0.2644579
precision	0.2385275
r2	0.0364465
recall	0.5997447
rmse	0.3701718
specificity	0.6026058

Above are some cross-validation metrics for the Python H2O module of flight delays using the gradient boosting estimator.

Results from scikit learn using one-hot encoding:

We provide an ROC curve for each of the classification methods that we have discussed for canceled flights as well as some of the key evaluation metrics.



Classifier	Accuracy	F1 Score	ROC	Precision
RandomForestClassifier	90.60000000000001	0.671	0.766	0.859
GradientBoostingClassifier	82.5	0.002	0.5	1.0

Recall	Log Loss	Sensitivity	Specificity	True Positives
0.551	3.26	0.981	0.551	16184
0.001	6.037	1.0	0.001	16501

Based on the ROC curve, we believe the random forest classifier to be the better performer in predicting the delay of flights in 2018.

References

Kaggle. (2022).Flight Status Prediction. Retrieved from: https://www.kaggle.com/datasets/robikscube/flight-delay-dataset-20182022?resource=download&select=raw.

CDC. (2022). CDC Museum COVID-19 Timeline. Retrieved from: https://www.cdc.gov/museum/timeline/covid19.html

Attachment

USD MSADS 502 Final Project - Team 3

Caleb McCurdy, Trevor Sauerbrey, and UE Wang

Part 1 Data preparation

```
[1]: # Import the required libraries
     import numpy as np
     import pandas as pd
     import matplotlib as mpl
     import matplotlib.pyplot as plt
     %matplotlib inline
     from sklearn.model_selection import train_test_split

     import warnings
     warnings.filterwarnings('ignore')
     pd.set_option('display.max_columns', 500)
     pd.set_option('display.max_rows', 500)
     pd.set_option('display.max_colwidth', 1000)

     %> Matplotlib inline

     ! pip install pandas-profiling[notebook,jupyter] --user --upgrade
```

```
[2]: ! pip install -U ydata-profiling[notebook,jupyter]
```

```
[3]: ! rm -rf ~/.local/share/jupyter/lab/staging
     ! jupyter lab build
     ! jupyter nbextension enable --py widgets --sys-prefix
     ! jupyter nbextension enable --py ydata_profiling --sys-prefix
     ! jupyter nbextension enable --py --sys-prefix qgrid
     ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[4]: ! pip install -U ydata-profiling[notebook,jupyter] --user
```

```
[5]: ! rm -rf ~/.local/share/jupyter/lab/staging
     ! jupyter lab build
     ! jupyter nbextension enable --py widgets --sys-prefix
     ! jupyter nbextension enable --py --sys-prefix qgrid
     ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[6]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[7]: ! rm -rf ~/.local/share/jupyter/lab/staging
     ! jupyter lab build
     ! jupyter nbextension enable --py widgets --sys-prefix
     ! jupyter nbextension enable --py --sys-prefix qgrid
     ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[8]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[9]: ! rm -rf ~/.local/share/jupyter/lab/staging
     ! jupyter lab build
     ! jupyter nbextension enable --py widgets --sys-prefix
     ! jupyter nbextension enable --py --sys-prefix qgrid
     ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[10]: ! pip install -U ydata-profiling[notebook,jupyter] --user
```

```
[11]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[12]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[13]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[14]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[15]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[16]: ! pip install -U ydata-profiling[notebook,jupyter] --user
```

```
[17]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[18]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[19]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[20]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[21]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[22]: ! pip install -U ydata-profiling[notebook,jupyter] --user
```

```
[23]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[24]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[25]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[26]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[27]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[28]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[29]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[30]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[31]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[32]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[33]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[34]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[35]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[36]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[37]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[38]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[39]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[40]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[41]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[42]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[43]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[44]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[45]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[46]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[47]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[48]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[49]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[50]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[51]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[52]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[53]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[54]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[55]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[56]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[57]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[58]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[59]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[60]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[61]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[62]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[63]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[64]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[65]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[66]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[67]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[68]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[69]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[70]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[71]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[72]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[73]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[74]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[75]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[76]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[77]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[78]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[79]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[80]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[81]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[82]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[83]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[84]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[85]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[86]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[87]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[88]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[89]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[90]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[91]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[92]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[93]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[94]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[95]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[96]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[97]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[98]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[99]: ! rm -rf ~/.local/share/jupyter/lab/staging
      ! jupyter lab build
      ! jupyter nbextension enable --py widgets --sys-prefix
      ! jupyter nbextension enable --py --sys-prefix qgrid
      ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[100]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[101]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[102]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[103]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[104]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[105]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[106]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[107]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[108]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[109]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[110]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[111]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[112]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[113]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[114]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[115]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[116]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[117]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[118]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[119]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[120]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[121]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[122]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[123]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[124]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[125]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[126]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[127]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[128]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[129]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[130]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[131]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[132]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[133]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[134]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[135]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[136]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[137]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[138]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[139]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[140]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[141]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[142]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[143]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[144]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[145]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[146]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[147]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[148]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[149]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[150]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[151]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[152]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[153]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[154]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[155]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[156]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[157]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[158]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[159]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[160]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[161]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[162]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[163]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[164]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[165]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[166]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[167]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[168]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[169]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[170]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[171]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[172]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[173]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[174]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[175]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[176]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[177]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[178]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[179]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[180]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[181]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[182]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[183]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[184]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[185]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[186]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[187]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[188]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[189]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[190]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

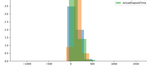
```
[191]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[192]: ! pip install pandas-profiling[notebook,jupyter] --user
```

```
[193]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       ! jupyter nbextension enable --py --sys-prefix qgrid
       ! jupyter nbextension enable --py --sys-prefix ipyleaflet
```

```
[194]: ! pip install -U pandas-profiling[notebook,jupyter] --user
```

```
[195]: ! rm -rf ~/.local/share/jupyter/lab/staging
       ! jupyter lab build
       ! jupyter nbextension enable --py widgets --sys-prefix
       
```



10. Delay (Departure Delay & Arrival Delay)

```
# plot 10 pie diagram for delay (Arrival Delay, Departure Delay and without delay data)
delay_table = pd.concat([df['ArrDelay'], df['DepDelay'], df['ArrDelay'].mask(df['ArrDelay'] > 0)])
```

```
# plot 10 pie diagram for delay (Arrival Delay, Departure Delay and without delay data)
delay_table = pd.concat([df['ArrDelay'], df['DepDelay'], df['ArrDelay'].mask(df['ArrDelay'] > 0)])
delay_table.groupby(delay_table > 0).count().plot.pie(figsize=(6,6),
                                                      labels=['Departure Delay', 'Arrival Delay'],
                                                      colors=['#ff9999','#66b3ff'],
                                                      autopct=lambda x: '{:.0f}'.format(x*delay_table.shape[0]/100))
plt.show();
```

```
# Departure Delay Level
df['Delay_level'] = pd.cut(df['DepDelay'], bins=[-1, 15, 60, 180, 1440], labels=['Delay_level_1', 'Delay_level_2', 'Delay_level_3', 'Delay_level_4'])
df.groupby('Delay_level').count[['DepDelay']].plot.bar(figsize=(6,6));
```

```
DepDelay_level  DepDelay 15-60  DepDelay 60-180  DepDelay >180  Delay_level_1
DepDelay 15-60          0.000            0.000          0.000          0.000
DepDelay 60-180         0.000            0.000          0.000          0.000
DepDelay >180           0.000            0.000          0.000          0.000
Delay_level_1           0.000            0.000          0.000          0.000
Delay_level_2           0.000            0.000          0.000          0.000
Delay_level_3           0.000            0.000          0.000          0.000
Delay_level_4           0.000            0.000          0.000          0.000
```

Departure Delay only

```
# plot 11 pie diagram for departure delay (flights only)
df['DepDelay'].plot.pie(figsize=(6,6));
```

```
# plot 11 pie diagram for departure delay (flights only)
df['DepDelay'].plot.pie(figsize=(6,6));
df.groupby('Delay_level').count[['DepDelay']].plot.bar(figsize=(6,6));
```

```
DepDelay_level  DepDelay 15-60  DepDelay 60-180  DepDelay >180  Delay_level_1
DepDelay 15-60          0.000            0.000          0.000          0.000
DepDelay 60-180         0.000            0.000          0.000          0.000
DepDelay >180           0.000            0.000          0.000          0.000
Delay_level_1           0.000            0.000          0.000          0.000
Delay_level_2           0.000            0.000          0.000          0.000
Delay_level_3           0.000            0.000          0.000          0.000
Delay_level_4           0.000            0.000          0.000          0.000
```

```
# plot 12 bar diagram for delay (Arrival Delay and Departure Delay)
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
```

```
# plot 12 bar diagram for delay (Arrival Delay and Departure Delay)
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
```

```
DepDelay_level  DepDelay 15-60  DepDelay 60-180  DepDelay >180  Delay_level_1
DepDelay 15-60          0.000            0.000          0.000          0.000
DepDelay 60-180         0.000            0.000          0.000          0.000
DepDelay >180           0.000            0.000          0.000          0.000
Delay_level_1           0.000            0.000          0.000          0.000
Delay_level_2           0.000            0.000          0.000          0.000
Delay_level_3           0.000            0.000          0.000          0.000
Delay_level_4           0.000            0.000          0.000          0.000
```

```
# plot 13 bar diagram for delay (Arrival Delay and Departure Delay) with percentage value
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
```

```
DepDelay_level  DepDelay 15-60  DepDelay 60-180  DepDelay >180  Delay_level_1
DepDelay 15-60          0.000            0.000          0.000          0.000
DepDelay 60-180         0.000            0.000          0.000          0.000
DepDelay >180           0.000            0.000          0.000          0.000
Delay_level_1           0.000            0.000          0.000          0.000
Delay_level_2           0.000            0.000          0.000          0.000
Delay_level_3           0.000            0.000          0.000          0.000
Delay_level_4           0.000            0.000          0.000          0.000
```

```
# plot 14 bar diagram for delay (Arrival Delay and Departure Delay) with percentage value
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
```

```
DepDelay_level  DepDelay 15-60  DepDelay 60-180  DepDelay >180  Delay_level_1
DepDelay 15-60          0.000            0.000          0.000          0.000
DepDelay 60-180         0.000            0.000          0.000          0.000
DepDelay >180           0.000            0.000          0.000          0.000
Delay_level_1           0.000            0.000          0.000          0.000
Delay_level_2           0.000            0.000          0.000          0.000
Delay_level_3           0.000            0.000          0.000          0.000
Delay_level_4           0.000            0.000          0.000          0.000
```

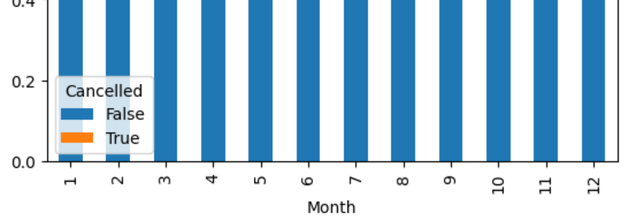
```
# plot 15 bar diagram for delay (Arrival Delay and Departure Delay) with percentage value
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
```

```
DepDelay_level  DepDelay 15-60  DepDelay 60-180  DepDelay >180  Delay_level_1
DepDelay 15-60          0.000            0.000          0.000          0.000
DepDelay 60-180         0.000            0.000          0.000          0.000
DepDelay >180           0.000            0.000          0.000          0.000
Delay_level_1           0.000            0.000          0.000          0.000
Delay_level_2           0.000            0.000          0.000          0.000
Delay_level_3           0.000            0.000          0.000          0.000
Delay_level_4           0.000            0.000          0.000          0.000
```

```
# plot 16 bar diagram for delay (Arrival Delay and Departure Delay) with percentage value
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
```

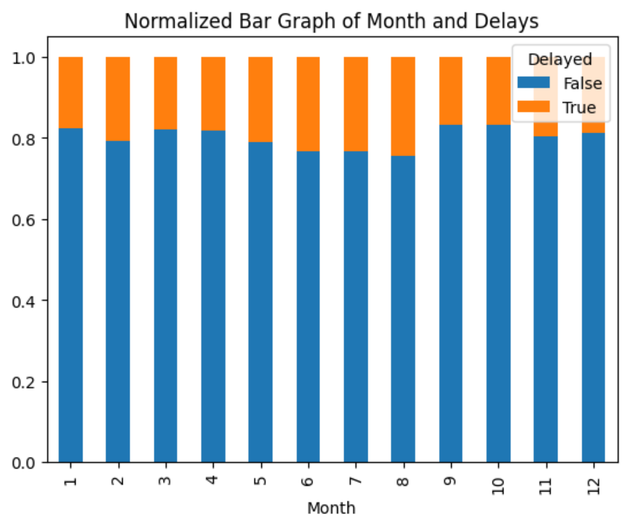
```
DepDelay_level  DepDelay 15-60  DepDelay 60-180  DepDelay >180  Delay_level_1
DepDelay 15-60          0.000            0.000          0.000          0.000
DepDelay 60-180         0.000            0.000          0.000          0.000
DepDelay >180           0.000            0.000          0.000          0.000
Delay_level_1           0.000            0.000          0.000          0.000
Delay_level_2           0.000            0.000          0.000          0.000
Delay_level_3           0.000            0.000          0.000          0.000
Delay_level_4           0.000            0.000          0.000          0.000
```

```
# plot 17 bar diagram for delay (Arrival Delay and Departure Delay) with percentage value
df.groupby('Delay_level').count[['ArrDelay', 'DepDelay']].plot.bar(figsize=(6,6));
```



In [75]:
```
# create a normalized bar graph of arrival delay rates for each month
crosstab_04 = pd.crosstab(flights_del['Month'], flights_del['Delayed'])
crosstab_norm4 = crosstab_04.div(crosstab_04.sum(1),axis = 0)
crosstab_norm4.plot(kind='bar', stacked = True, title = 'Normalized Bar Graph of Month and Delays')
```

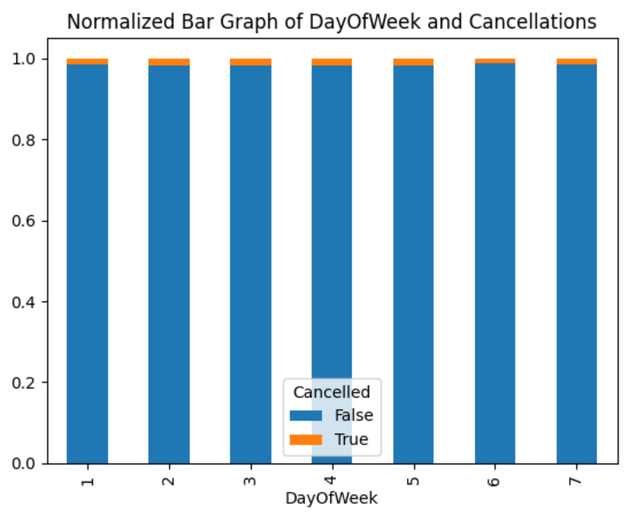
Out[75]:
```
<AxesSubplot: title={'center': 'Normalized Bar Graph of Month and Delays'}, xlabel='Month'>
```



5. Day Of Week

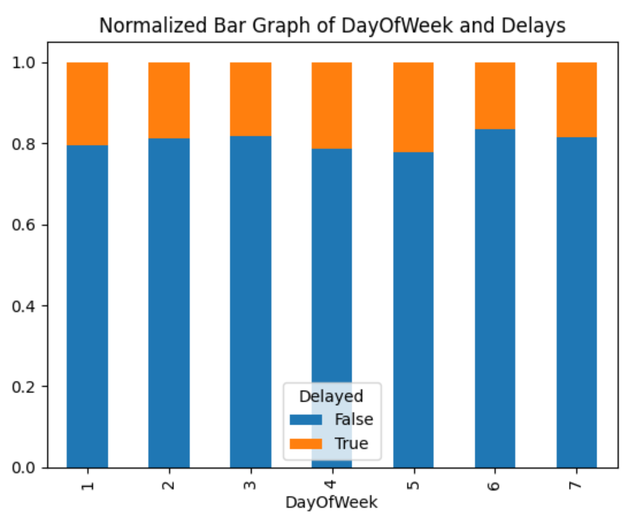
In [76]:
```
# create a normalized bar graph of cancellation rates for each day of the week
crosstab_05 = pd.crosstab(flights_can['DayOfWeek'], flights_can['Cancelled'])
crosstab_norm5 = crosstab_05.div(crosstab_05.sum(1),axis = 0)
crosstab_norm5.plot(kind='bar', stacked = True, title = 'Normalized Bar Graph of DayOfWeek and Cancellations')
```

Out[76]:
```
<AxesSubplot: title={'center': 'Normalized Bar Graph of DayOfWeek and Cancellations'}, xlabel='DayOfWeek'>
```



In [77]:
```
# create a normalized bar graph of arrival delay rates for each day of the week
crosstab_06 = pd.crosstab(flights_del['DayOfWeek'], flights_del['Delayed'])
crosstab_norm6 = crosstab_06.div(crosstab_06.sum(1),axis = 0)
crosstab_norm6.plot(kind='bar', stacked = True, title = 'Normalized Bar Graph of DayOfWeek and Delays')
```

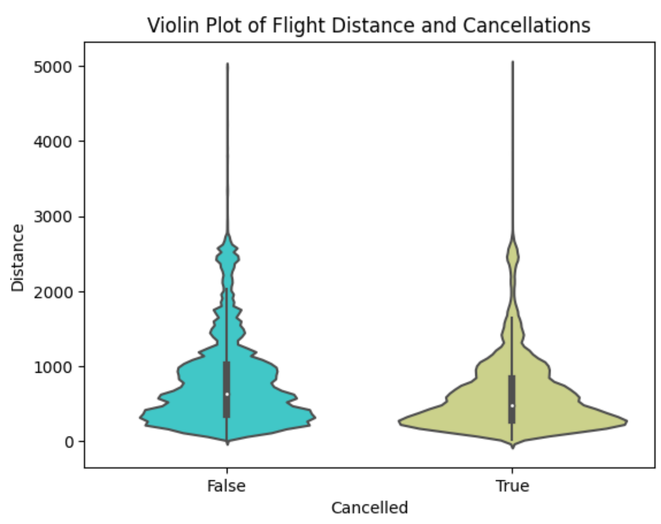
Out[77]:
```
<AxesSubplot: title={'center': 'Normalized Bar Graph of DayOfWeek and Delays'}, xlabel='DayOfWeek'>
```



6. Distance

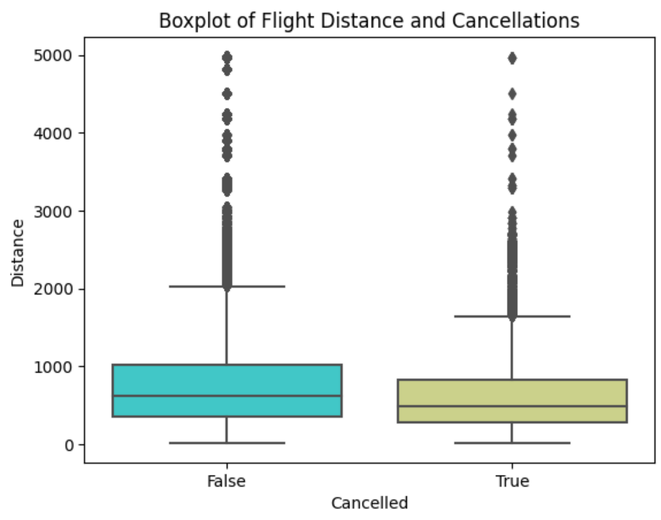
In [78]:
```
# create a violin plot of relationship between cancellation rates and distance
sns.violinplot(x="Cancelled", y="Distance", data=flights_can,
               palette='rainbow').set(title='Violin Plot of Flight Distance and Cancellations')
```

Out[78]:
```
[Text(0.5, 1.0, 'Violin Plot of Flight Distance and Cancellations')]
```



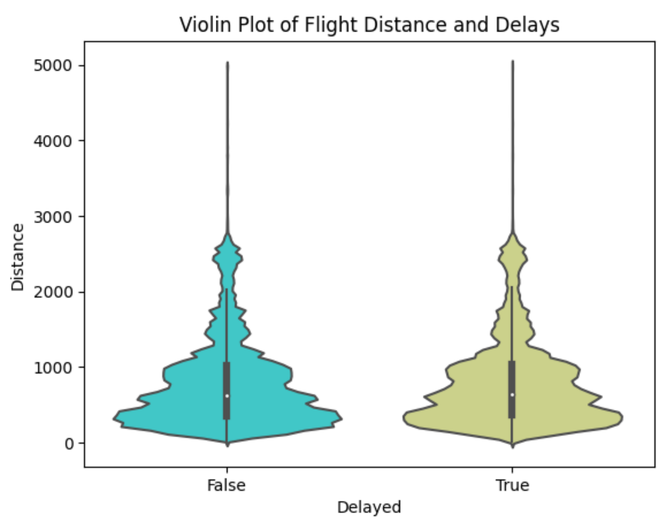
In [79]:
```
# create a boxplot of relationship between cancellation rates and distance
sns.boxplot(x="Cancelled", y="Distance", data=flights_can,
            palette='rainbow').set(title='Boxplot of Flight Distance and Cancellations')
```

Out[79]:
```
[Text(0.5, 1.0, 'Boxplot of Flight Distance and Cancellations')]
```



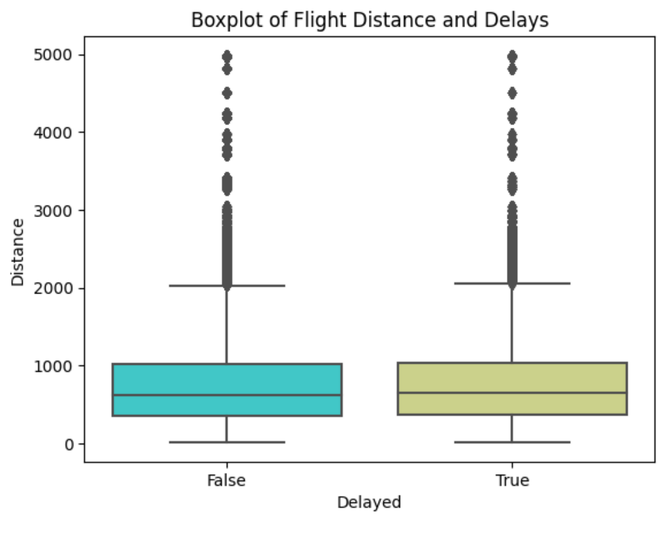
In [80]:
```
# create a violin plot of relationship between arrival delay rates and distance
sns.violinplot(x="Delayed", y="Distance", data=flights_del,
               palette='rainbow').set(title='Violin Plot of Flight Distance and Delays')
```

Out[80]:
```
[Text(0.5, 1.0, 'Violin Plot of Flight Distance and Delays')]
```



In [81]:
```
# create a boxplot of relationship between arrival delay rates and distance
sns.boxplot(x="Delayed", y="Distance", data=flights_del,
            palette='rainbow').set(title='Boxplot of Flight Distance and Delays')
```

Out[81]:
```
[Text(0.5, 1.0, 'Boxplot of Flight Distance and Delays')]
```



Classification Modeling using Scikit learn

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import os, sys, gc
from sklearn import tableize
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier,GradientBoostingClassifier
from sklearn.ensemble import GaussianNB, BernoulliNB, MultinomialNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis, QuadraticDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV, PassiveAggressiveClassifier
from sklearn.metrics import cohen_kappa_score,mean_squared_error, accuracy_score, log_loss, roc_curve,auc,roc_auc_score, hamming_loss, f1score,recall_score, precision_score, recall_score
from sklearn.metrics import classification_report, accuracy_score, precision_recall_fscore_support, confusion_matrix
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,
from sklearn.preprocessing import OrdinalEncoder, MinMaxScaler, LabelEncoder, OneHotEncoder, MinMaxScaler
from sklearn.utm import UTC, LinearUTC, NaUTC
from sklearn.utm.utmscale import UTMScaleNorth
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import SMOTE
import h2o
from h2o.automlearn import H2ORandomForestEstimator
from h2o.automlearn import H2OGradientBoosting
from h2o.automlearn.gbm import H2OGradientBoostingEstimator
```

In [ ]:
```
canceled_data = pd.read_csv("C:\FEDFUND_512_Fall_Applied_Data_Mining\512_data_mining_project\canceled_flight_data.csv")
canceled_data.head()
```

Out[ ]:

	Airline	Origin	Dest	Canceled	Distance	Month	DayOfWeek	
0	Endeavor Air Inc.	ABY	ATL	False	145.0	January	Tuesday	
1	Endeavor Air Inc.	ABY	ATL	False	145.0	January	Wednesday	
2	Endeavor Air Inc.	ABY	ATL	False	145.0	January	Thursday	
3	Endeavor Air Inc.	ABY	ATL	False	145.0	January	Friday	
4	Endeavor Air Inc.	ABY	ATL	False	145.0	January	Saturday	

In [ ]:
```
# Get number of non values for each column of dataframe
for i, col in enumerate(canceled_data.columns):
    cnt = canceled_data.loc[:, col].shape[0]
    print(cnt,end=' ')
```

In [ ]:
```
canceled_data.shape
```

In [ ]:
```
canceled_data.dtypes
```

In [ ]:
```
canceled_data.nunique()
```

In [ ]:
```
canceled_data.index()
```

In [ ]:
```
canceled_flights_w_dummies = pd.get_dummies(canceled_data, drop_first=True)
```

In [ ]:
```
# print(canceled_flights_w_dummies.columns)
```

In [ ]:
```
scaler = MinMaxScaler()
y_scaled = scaler.fit_transform(DONPLM)

X = DONPLM.drop(columns='Canceled')
y = DONPLM['Canceled']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, shuffle=True)
```

In [ ]:
```
# add the models to the list that you want to view on the ROC plot
models = [
    {
        'label': 'DecisionTreeClassifier',
        'model': DecisionTreeClassifier(),
    },
    {
        'label': 'ExtraTreeClassifier',
        'model': ExtraTreeClassifier(),
    },
    {
        'label': 'RandomForestClassifier',
        'model': RandomForestClassifier(),
    },
    {
        'label': 'AdaBoostClassifier',
        'model': AdaBoostClassifier(),
    },
    {
        'label': 'GradientBoostingClassifier',
        'model': GradientBoostingClassifier(),
    },
    {
        'label': 'GaussianNB',
        'model': GaussianNB(),
    },
    {
        'label': 'BernoulliNB',
        'model': BernoulliNB(),
    },
    {
        'label': 'MultinomialNB',
        'model': MultinomialNB(),
    },
    {
        'label': 'LinearDiscriminantAnalysis',
        'model': LinearDiscriminantAnalysis(),
    },
    {
        'label': 'LogisticRegression',
        'model': LogisticRegression(),
    },
    {
        'label': 'QuadraticDiscriminantAnalysis',
        'model': QuadraticDiscriminantAnalysis(),
    }
]
```

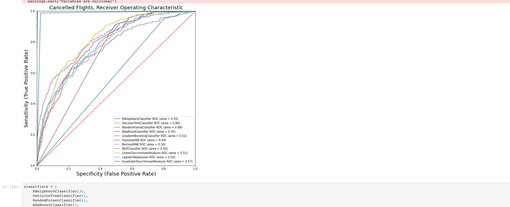
```
from joblib import dump,load
cvfbsamp,*_=joblib.dump(X_y,0,1,9)
```

```
# Below for-loop iterates through your models list
for m in models:
    model = m['model'] # select the model
    model.fit(X_train, y_train)
    y_predicted, probs = m['model'].predict(X_test)
# Compute False positive rate, and True positive rate
    fpr, tpr, thresholds = metrics.roc_curve(y_test,model.predict_proba(X_test)[:,1])
# Calculate Area under the curve to display on the plot
    auc = metrics.roc_auc_score(y_test,model.predict(X_test))
# Now, plot the computed values
    plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (m['label'], auc))
# Custom settings for the plot
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('Specificity (False Positive Rate)', fontsize=15)
plt.ylabel('Sensitivity (True Positive Rate)', fontsize=15)

plt.title('Canceled Flights, Receiver Operating Characteristic', fontsize=20)
plt.legend(loc='best')
plt.show() # Display
```

```
C:\Users\rsenot.saas\envs\MYpy64-1970\python-3.6.9.amd64\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\rsenot.saas\envs\MYpy64-1970\python-3.6.9.amd64\lib\site-packages\sklearn\discriminant_analysis.py:836: UserWarning: Variables are collinear
  warnings.warn("Variables are collinear")
```



In [ ]:
```
classifiers = [
    DecisionTreeClassifier(),
    ExtraTreeClassifier(),
    RandomForestClassifier(),
    AdaBoostClassifier(),
    GradientBoostingClassifier(),
    GaussianNB(),
    BernoulliNB(),
    MLPClassifier(
    hidden_layer_sizes=(100, 100),),
    LinearDiscriminantAnalysis(),
    LogisticRegression(),
    QuadraticDiscriminantAnalysis(),
]

log_cols=["Classifier", "Accuracy", "F1 Score", "ROC", "Precision", "Recall", "Log Loss", "Sensitivity", "Specificity","True Positives"]
log = pd.DataFrame(columns=log_cols)

for clf in classifiers:
    clf.fit(X, y)
    name = clf.__class__.__name__

    print("****")
    print(name)
    print("****Results****")

    train_predictions = clf.predict(X_test)
    acc = accuracy_score(y_test, train_predictions)
    print("Accuracy: {:%}".format(acc))
    print("Accuracy: {:.4f}".format(acc))

    fbeta = fbeta_score(y_test, train_predictions, beta=1)
    fbeta = fbeta_round(f)
    print("F1 Score: {:".format(fbeta)

    roc = roc_auc_score(y_test, train_predictions)
    roc = roc_round(r)
    print("ROC (ROC) Score: {:".format(roc)

    precision = precision_score(y_test, train_predictions, average='binary')
    precision = precision_round(p)
    print("Precision Score: {:".format(precision)

    recall = recall_score(y_test, train_predictions)
    recall = recall_round(r)
    print("Recall Score: {:".format(recall))

    TP = confusion_matrix(y_test,train_predictions).ravel()[3]
    TN = confusion_matrix(y_test,train_predictions).ravel()[0]
    sensitivity = TP / (TP + FN)
    sensitivity = sensitivity_round(s)
    print("Sensitivity Score: {:".format(sensitivity)

    TP = confusion_matrix(y_test,train_predictions).ravel()[3]
    TN = confusion_matrix(y_test,train_predictions).ravel()[0]
    specificity = TN / (TN + FP)
    specificity = specificity_round(s)
    print("Specificity Score: {:".format(specificity)

    true_positives = confusion_matrix(y_test,train_predictions).ravel()[3]

    print("\nCanceled train score: {np.round(clf.score(X_train), 2)}\n")
    print("Canceled test score: {np.round(clf.score(X_test), 2)}\n")
    print("Classification report: \n{classification_report(y_test, train_predictions)}")
    print("\nConfusion matrix: \n{confusion_matrix(y_test,train_predictions)}")

    ll = log_loss(y_test, train_predictions)
    ll = ll_round(l)

    log_entry = pd.DataFrame([[name, acc*100, fbeta, roc, precision, recall, ll, sensitivity, specificity, true_positives]], columns=log_cols)
    log = pd.concat([log,log_entry])
```

```
-----------------------------------------------------------------------------
DecisionTreeClassifier
****Results****
Accuracy: 96.0959%
Accuracy: 0.9610
F1 Score: {:
ROC (ROC) Score: 0.546
Precision Score: 0.676
Recall Score: 0.112
Sensitivity Score: 0.999
Specificity Score: 0.092

Canceled train score: 0.9987
Canceled test score: 0.9601

Classification report:
{1}9750    0{1}
{1}  237    2{1}

-----------------------------------------------------------------------------
ExtraTreeClassifier
****Results****
Accuracy: 95.7935%
Accuracy: 0.9579
ROC (ROC) Score: 0.476
Precision Score: 0.000
Recall Score: 0.752
Sensitivity Score: 1.0
Specificity Score: 0.752

Canceled train score: 0.9571
Canceled test score: 0.9568
Confusion matrix:
{1}9750    0{1}
{1}  58  191{1}

-----------------------------------------------------------------------------
RandomForestClassifier
****Results****
Accuracy: 96.7999%
Accuracy: 0.9680
ROC (ROC) Score: 0.655
Precision Score: 0.955
Recall Score: 0.312
Sensitivity Score: 1.0
Specificity Score: 0.748

Canceled train score: 0.9971
Canceled test score: 0.9666
Confusion matrix:
{1}9750    0{1}
{1}  174  135{1}

-----------------------------------------------------------------------------
AdaBoostClassifier
****Results****
Accuracy: 96.6959%
F1 Score: 0.
Recall Score: 0.5
Precision Score: 0.5
Recall Score: 0.0
Sensitivity Score: 1.0
Specificity Score: 0.0

C:\Users\rsenot.saas\envs\MYpy64-1970\python-3.6.9.amd64\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use 'zero_division' parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Canceled train score: 0.9683
Canceled test score: 0.9679
Confusion matrix:
{1}9750    0{1}
{1}  250    0{1}

-----------------------------------------------------------------------------
GradientBoostingClassifier
****Results****
Accuracy: 96.9959%
F1 Score: 0.
ROC (ROC) Score: 0.52
Precision Score: 0.5
Recall Score: 0.04
Sensitivity Score: 1.0
Specificity Score: 0.04

Canceled train score: 0.9697
Canceled test score: 0.969
Confusion matrix:
{1}9750    0{1}
{1}  240   10{1}

-----------------------------------------------------------------------------
GaussianNB
****Results****
Accuracy: 62.7999%
F1 Score: 0.
ROC (ROC) Score: 0.487
Precision Score: 0.02
Recall Score: 0.02
Sensitivity Score: 0.974
Specificity Score: 0.956

Canceled train score: 0.625
Canceled test score: 0.6252
Confusion matrix:
{1}6235 3485{1}
{1}   11  239{1}

-----------------------------------------------------------------------------
BernoulliNB
****Results****
Accuracy: 96.7999%
F1 Score: 0.
Recall Score: 0.504
Precision Score: 0.2
Recall Score: 0.0
Sensitivity Score: 1.0
Specificity Score: 0.008

Canceled train score: 0.9679
Canceled test score: 0.9672
Confusion matrix:
{1}9742    8{1}
{1}  250    0{1}

-----------------------------------------------------------------------------
MLPClassifier
****Results****
Accuracy: 96.6959%
F1 Score: 0.
ROC (ROC) Score: 0.5
Precision Score: 0.0
Recall Score: 0.0

C:\Users\rsenot.saas\envs\MYpy64-1970\python-3.6.9.amd64\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use 'zero_division' parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Sensitivity Score: 1.0
Specificity Score: 0.0

Canceled train score: 0.9683
Canceled test score: 0.9675
Confusion matrix:
{1}9750    0{1}
{1}  250    0{1}

-----------------------------------------------------------------------------
MLPClassifier
****Results****
Accuracy: 96.6959%
F1 Score: 0.001
ROC (ROC) Score: 0.522
Precision Score: 0.5
Recall Score: 0.004
Sensitivity Score: 1.0
Specificity Score: 0.044

Canceled train score: 0.499
Canceled test score: 0.4979
Confusion matrix:
{1}9747    3{1}
{1}  249    1{1}

-----------------------------------------------------------------------------
LinearDiscriminantAnalysis
****Results****
Accuracy: 96.6959%
F1 Score: 0.046
ROC (ROC) Score: 0.514
Precision Score: 0.5
Recall Score: 0.024
Sensitivity Score: 1.0
Specificity Score: 0.052

Canceled train score: 0.9667
Canceled test score: 0.9663
Confusion matrix:
{1}9744    6{1}
{1}  244    6{1}

C:\Users\rsenot.saas\envs\MYpy64-1970\python-3.6.9.amd64\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\rsenot.saas\envs\MYpy64-1970\python-3.6.9.amd64\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use 'zero_division' parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
-----------------------------------------------------------------------------
LogisticRegression
****Results****
Accuracy: 96.6959%
F1 Score: 0.0
Recall Score: 0.5
Precision Score: 0.0
Recall Score: 0.0
Sensitivity Score: 1.0
Specificity Score: 0.0

Canceled train score: 0.9683
Canceled test score: 0.9675
Confusion matrix:
{1}9750    0{1}
{1}  250    0{1}

C:\Users\rsenot.saas\envs\MYpy64-1970\python-3.6.9.amd64\lib\site-packages\sklearn\discriminant_analysis.py:836: UserWarning: Variables are collinear
  warnings.warn("Variables are collinear")
-----------------------------------------------------------------------------
QuadraticDiscriminantAnalysis
****Results****
Accuracy: 20.7999%
F1 Score: 0.
ROC (ROC) Score: 0.575
Precision Score: 0.015
Recall Score: 0.964
Sensitivity Score: 0.187
Specificity Score: 0.952

Canceled train score: 0.2053
Canceled test score: 0.2069
Confusion matrix:
{1}2030 7720{1}
{1}   10  240{1}
```

In [ ]:
```
# Save df as .csv
def render_mpl_table(data=df, col_width=3.0, row_height=0.625, font_size=14,
                     header_color='#40466e', row_colors=['#f1f1f', '#ffffff'], edge_color='w',
                     bbox=[0, 0, 1, 1], header_columns=0,
                     ax=None, **kwargs):
    if ax is None:
        size = (np.array(data.shape[::-1]) + np.array([1, 1])) * np.array([col_width, row_height])
        fig, ax = plt.subplots(figsize=size)
        ax.axis('off')
    mpl_table = ax.table(cellText=data.values,bbox=bbox, colLabels=data.columns, **kwargs)
    mpl_table.auto_set_font_size(False)
    mpl_table.set_fontsize(font_size)

    for k, cell in mpl_table._cells.items():
        cell.set_edgecolor(edge_color)
        if k[0] == 0 or k[1] < header_columns:
            cell.set_text_props(weight='bold', color='w')
            cell.set_facecolor(header_color)
        else:
            cell.set_facecolor(row_colors[k[0]%len(row_colors) ])
    return ax.get_figure(), ax

fig,ax = render_mpl_table(log, header_columns=0, col_width=2.0)
ax.set_title('Canceled Flights, Classification Models Performance Metrics Table', fontsize=20)
fig.savefig('table_mpl.png')
```

Canceled Flights, Classification Models Performance Metrics Table

Classifier	Accuracy	F1 Score	ROC	Precision	Recall	Log Loss	Sensitivity	Specificity	True Positives
DecisionTreeClassifier	96.0	0.252	0.546	0.676	0.092	0.411	0.999	0.092	19750
ExtraTreeClassifier	96.7	0.858	0.476	1.0	0.752	0.107	1.0	0.752	19750
RandomForestClassifier	96.7	0.651	0.655	0.955	0.708	0.116	1.0	0.708	19750
AdaBoostClassifier	96.6	0.0	0.5	0.0	0.0	0.432	1.0	0.0	19750
GradientBoostingClassifier	96.9	0.077	0.52	1.0	0.04	0.424	1.0	0.04	19750
GaussianNB	62.7	0.04	0.487	0.02	0.956	16.051	0.618	0.956	9250
BernoulliNB	96.7	0.015	0.504	0.2	0.008	0.442	1.0	0.008	19750
MLPClassifier	96.6	0.0	0.5	0.0	0.0	0.432	1.0	0.0	19750
MLPClassifier	96.6	0.001	0.522	0.786	0.044	0.428	1.0	0.044	19750
LinearDiscriminantAnalysis	96.6	0.046	0.514	0.5	0.024	0.342	0.996	0.052	19679
LogisticRegression	96.6	0.0	0.5	0.0	0.0	0.432	1.0	0.0	19750
QuadraticDiscriminantAnalysis	20.7	0.029	0.575	0.015	0.952	27.393	0.187	0.952	3660

In [ ]:
```
```

Classification Modeling using H2O Python

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import numpy
import datetime as dtm
import h2o, os, sys, shutil, requests
import h2o.utils.shutil
from h2o.estimators import DeepLearningClassifier, XGBoostClassifier, GradientBoostingClassifier
from h2o.search.grid.grid_search import H2OGridSearch, RandomDiscrete, CartesianOrder
from h2o.automl import H2OAutoML, get_leaderboard, get_extended_leaderboard, get_leaderboard_with_customMetrics
from h2o.exceptions import H2ONotFoundArrayException, H2OValueRecognitionException
from h2o.sklearn.wrapper import H2ODummyEncoder, H2OSklearnFeatureUnion, H2OSklearnStackedEnsemble, H2OSklearnTransformer
from h2o.sklearn.wrapper import H2OSklearnEstimator, H2OSklearnClassifier, H2OSklearnRegressor
from h2o.sklearn.utils import accuracy_score, auc_score, log_loss, r2_score, mse, mae, rmse, rmsle, f1_score, precision_score, recall_score
from h2o.sklearn.utils import mean_absolute_error, mean_squared_error, median_absolute_error, mean_residual_deviance, DkullbackLeibler, log_loss, gini
from h2o.sklearn.utils import classification_report, accuracy_score, precision_recall_fscore_support, confusion_matrix
from h2o.sklearn.utils import cross_validate, cross_validate_score
from h2o.sklearn.utils import GridSearchCV, RandomizedSearchCV, BaseEstimator, ClassifierMixin
from h2o.sklearn.utils import MOJo, ticsmeOPT, MojoPT
from h2o.sklearn.utils import H2OPipeline, H2OFeatureUnion
from h2o.sklearn.utils import MojoPipeline, MojoFeatureUnion
from h2o.sklearn.utils import H2OTransformerMixin, H2OPredictMixin
from h2o.labels.eval_confusion_matrix import confusion_matrix
from imblearn.over_sampling.ADASYN import ADASYN

import h2o
from h2o.estimators.gbm_v3 import H2OGradientBoostingEstimator
from h2o.estimators.xgboost_v3 import H2OXGBoostEstimator
from h2o.estimators.deeplearning_v3 import H2ODeepLearningEstimator
from h2o.estimators.glm_v3 import H2OGeneralizedLinearEstimator
from h2o.sklearn.wrapper import H2OGradientBoostingClassifier
from imblearn.over_sampling.ADASYN import ADASYN
```

H2O Analysis of features

In [2]:
```
h2o.init()
h2o.connect()
```

```
Checking whether there is an H2O instance running at http://localhost:54321 ..... connected.
--------------------------  ------------------------------------------------------------
H2O_cluster_uptime:         3 hours 14 min
H2O_cluster_timezone:       America/Los_Angeles
H2O_data_parsing_timezone:  UTC
H2O_cluster_version:        3.38.0.3
H2O_cluster_name:           fit_dean
H2O_cluster_total_nodes:    1
H2O_cluster_free_memory:    6.793 Gb
H2O_cluster_total_cores:    16
H2O_cluster_allowed_cores:  16
H2O_cluster_status:         locked, healthy
H2O_connection_url:         http://localhost:54321
H2O_connection_proxy:       {"http": null, "https": null}
H2O_internal_security:      False
Python_version:             3.8.8 final
--------------------------  ------------------------------------------------------------
Connecting to H2O server at http://localhost:54321 ..... successful.
--------------------------  ------------------------------------------------------------
H2O_cluster_uptime:         3 hours 14 min
H2O_cluster_timezone:       America/Los_Angeles
H2O_data_parsing_timezone:  UTC
H2O_cluster_version:        3.38.0.3
H2O_cluster_name:           fit_dean
H2O_cluster_total_nodes:    1
H2O_cluster_free_memory:    6.793 Gb
H2O_cluster_total_cores:    16
H2O_cluster_allowed_cores:  16
H2O_cluster_status:         locked, healthy
H2O_connection_url:         http://localhost:54321
H2O_connection_proxy:       {"http": null, "https": null}
H2O_internal_security:      False
Python_version:             3.8.8 final
```


```
H2OConnection to http://localhost:54321, no session!
```


In [3]:
```
df = h2o.import_file("weather_data_train.csv")
df_test = h2o.import_file("weather_data_test.csv")
```

```
Parse progress: |(done) 100%
```

In [4]:
```
df.describe()
```

Rows:100854
Cols:7

	Airlne	Origin	Dest	Cancelled	Distance	Month	DayOfWeek
type	enum	enum	enum	enum	int	enum	enum
mins					79.0		
mean					786.399937404843		
maxs					4983.0		
sigma					584.397491876167		
zeros					0		
missing	0	0	0	0	0	0	0
0	Endeavor Air Inc.	ORF	ATL	False	745.0	January	Tuesday
1	Endeavor Air Inc.	ORF	ATL	False	745.0	January	Wednesday
2	Endeavor Air Inc.	ORF	ATL	False	745.0	January	Friday
3	Endeavor Air Inc.	ORF	ATL	False	745.0	January	Saturday
4	Endeavor Air Inc.	ORF	ATL	False	745.0	January	Sunday
5	Endeavor Air Inc.	ORF	ATL	False	745.0	January	Monday
6	Endeavor Air Inc.	ORF	ATL	False	745.0	January	Tuesday
7	Endeavor Air Inc.	ORF	ATL	False	745.0	January	Wednesday
8	Endeavor Air Inc.	ATL	ABR	False	190.0	January	Wednesday

[100854 rows x 7 column]

In [5]:
```
# scaling numerical data
df["Distance"] = df["Distance"].apply(lambda x: float, axis= True)
```

In [6]:
```
df.describe()
```

Rows:100854
Cols:7

	Airlne	Origin	Dest	Cancelled	Distance	Month	DayOfWeek
type	enum	enum	enum	enum	real	enum	enum
mins					-1.29226692387755		
mean					-1.25266992237884e-16		
maxs					3.57476029375996		
sigma					1.00000000000000e+00		
zeros					0		
missing	0	0	0	0	0	0	0
0	Endeavor Air Inc.	ORF	ATL	False	-1.08027168857984e+00	January	Tuesday
1	Endeavor Air Inc.	ORF	ATL	False	-1.08027168857984e+00	January	Wednesday
2	Endeavor Air Inc.	ORF	ATL	False	-1.08027168857984e+00	January	Friday
3	Endeavor Air Inc.	ORF	ATL	False	-1.08027168857984e+00	January	Saturday
4	Endeavor Air Inc.	ORF	ATL	False	-1.08027168857984e+00	January	Sunday
5	Endeavor Air Inc.	ORF	ATL	False	-1.08027168857984e+00	January	Monday
6	Endeavor Air Inc.	ORF	ATL	False	-1.08027168857984e+00	January	Tuesday
7	Endeavor Air Inc.	ORF	ATL	False	-1.08027168857984e+00	January	Wednesday
8	Endeavor Air Inc.	ATL	ABR	False	-1.08027168857984e+00	January	Wednesday

[100854 rows x 7 column]

In [7]:
```
# Splitting our data into train and training set
train, valid = df.split_frame(ratios = [.8])

X = ['Airline', 'Origin', 'Dest', 'Distance', 'Month', 'DayOfWeek']

y = 'Cancelled'
y = y.lower()
```

In [8]:
```
# Defining our models

gbm = H2OGradientBoostingEstimator(ntrees=50, stopping_metric='AUC',
                   stopping_rounds=3,
                   stopping_tolerance=0.01,
                   stopping_metric='AUC',
                   score_each_iteration=True)
```

In [9 finished]: gbm.train(x=X,y=y,training_frame=train, validation_frame = valid)

```
# Plotting variable importance using Binned Pressed
v1 = gbm.varimp(use_pandas=True)
v1.head(10).sort_values('rel_imp')
v2 = v1.sort_values('rel_imp', ascending=False)
v3 = v2.set_index('name')

v3.sort_values(by = 'rel Variable Importance').plot(kind='barh', sort=True, figsize = (25,15), fontsize = 35)
```

```
# Gradient Boosting Estimator Model
model = H2OGradientBoostingEstimator(distribution='bernoulli', ntrees=50, max_depth= 4, min_rows=1, learn_rate= 0.1, nfolds=5)

model.train(x, y, training_frame=train, validation_frame=valid)
```

```
gbm Grid Progress: |(done) 100%
```

```
Model Details
=============
H2OGradientBoostingEstimator : Gradient Boosting Machine
Model Key: GBM_model_python_1678891881988_824

Model Summary:
```

number_of_trees	number_of_internal_trees	model_size_in_bytes	min_depth	max_depth	mean_depth	min_leaves	max_leaves	mean_leaves
44.0	44.0	14586.0	3.0	3.0	3.0000182	12.0	8.0	12.04908

```
ModelMetrics(nfolds=5): gbm
** Reported on cross-validation data. **

MSE: 0.07134873285761024
RMSE: 0.26711024422000344
LogLoss: 0.2424164267924188
Mean Per-Class Error: 0.42891549762885174
AUC: 0.7577192000000001
AUCPR: 0.4548511217860211
Gini: 0.5154384000000002

Confusion Matrix (vertical) for max F1 @ threshold = 0.03543775002732791:
```

	False	True	Error	Rate
False	67215.0	15884.0	0.1243	=15884/82099
True	4215.0	1575.0	0.7421	=4215/5790
Total	67430.0	17459.0	0.2108	=19999/87889

```
Maximum Metrics: Maximum metrics at their respective thresholds
```

metric	threshold	value	idx
max f1	0.035437	0.172721	351
max f2	0.014735	0.363910	393
max f0point5	0.076346	0.116341	183
max accuracy	0.576346	0.933641	0
max precision	0.576346	0.666667	12
max recall	0.000025	1.0	399
max specificity	0.576346	1.0	0
max absolute_mcc	0.035437	0.150365	351
max min_per_class_accuracy	0.014735	0.704616	393
max mean_per_class_accuracy	0.012108	0.752701	399
max tns	0.576346	84099.0	0
max fns	0.576346	5494.0	0
max fps	0.000025	84099.0	399
max tps	0.000025	5790.0	399
max tnr	0.576346	1.0	0
max fnr	0.576346	0.949051	0
max fpr	0.000025	1.0	399
max tpr	0.000025	1.0	399

```
GainsLift Table: Avg response rate: 1.25 %, avg score: 1.25 %
```

group	cumulative_data_fraction	lift	cumulative_lift	response_rate	score	cumulative_response_rate	cumulative_score	capture_rate	cumulative_capture_rate	gain	cumulative_gain	kolmogorov_smirnov
1	0.010023	3.206355	3.206355	0.015893	0.121535	0.116393	0.121535	0.032127	0.032127	220.635501	220.635501	0.031512
2	0.020046	2.608696	2.907525	0.012931	0.080598	0.014397	0.101067	0.026256	0.058383	160.869565	190.752533	0.056895
3	0.030069	2.050351	2.621729	0.010164	0.064250	0.013000	0.086266	0.020725	0.079108	105.035088	162.172933	0.076966
4	0.040092	1.863354	2.436957	0.009237	0.055968	0.008821	0.076900	0.018635	0.097743	86.335404	143.695652	0.094177
5	0.050008	1.620464	2.321442	0.008076	0.047533	0.007477	0.069972	0.016174	0.113916	62.046372	132.144206	0.108236
6	0.100015	1.313793	2.067617	0.006545	0.038168	0.006334	0.053953	0.065544	0.179461	31.379311	106.761688	0.171872
7	0.150007	0.921060	1.685997	0.004587	0.031597	0.003475	0.044513	0.045855	0.225316	-7.894026	68.599688	0.202455
8	0.200005	0.807486	1.507500	0.004021	0.026811	0.003242	0.038187	0.040351	0.265667	-19.251400	50.750000	0.225000
9	0.300007	0.621847	1.238616	0.003097	0.021908	0.002611	0.031502	0.062197	0.327863	-37.815303	23.861600	0.263267
10	0.400009	0.518208	1.077500	0.002580	0.019127	0.002317	0.027411	0.051813	0.379677	-48.179200	7.750000	0.283333
11	0.500011	0.414567	0.966000	0.002064	0.017488	0.002117	0.024900	0.041522	0.421199	-58.543300	-3.400000	0.290000
12	0.600013	0.414567	0.897000	0.002064	0.016492	0.001917	0.023488	0.041522	0.462721	-58.543300	-10.300000	0.290000
13	0.700015	0.414567	0.841000	0.002064	0.015800	0.001917	0.022600	0.041522	0.504242	-58.543300	-15.900000	0.290000
14	0.800017	0.414567	0.793000	0.002064	0.015298	0.001917	0.021900	0.041522	0.545763	-58.543300	-20.700000	0.290000
15	0.900019	0.414567	0.750000	0.002064	0.014900	0.001917	0.021300	0.041522	0.587284	-58.543300	-25.000000	0.290000
16	1.000000	0.412284	0.700000	0.002053	0.014600	0.001906	0.020800	0.041284	0.628568	-58.771600	-30.000000	0.290000

```
ModelMetrics(nfolds=5): gbm
** Reported on cross-validation data. **

MSE: 0.07134873285761024
RMSE: 0.26711024422000344
LogLoss: 0.2424164267924188
Mean Per-Class Error: 0.42891549762885174
AUC: 0.7577192000000001
AUCPR: 0.4548511217860211
Gini: 0.5154384000000002

Confusion Matrix (vertical) for max F1 @ threshold = 0.035437500273279:
```

	False	True	Error	Rate
False	67215.0	15884.0	0.1243	=15884/82099
True	4215.0	1575.0	0.7421	=4215/5790
Total	71430.0	17459.0	0.2108	=20099/87889

```
Maximum Metrics: Maximum metrics at their respective thresholds
```

metric	threshold	value	idx
max f1	0.035437	0.172721	351
max f2	0.014735	0.363910	393
max f0point5	0.076346	0.116341	183
max accuracy	0.576346	0.933641	0
max precision	0.576346	0.666667	12
max recall	0.000025	1.0	399
max specificity	0.576346	0.999987	0
max absolute_mcc	0.035437	0.150365	351
max min_per_class_accuracy	0.014735	0.704616	393
max mean_per_class_accuracy	0.012108	0.752701	399
max tns	0.576346	84098.0	0
max fns	0.576346	5494.0	0
max fps	0.000025	84099.0	399
max tps	0.000025	5790.0	399
max tnr	0.576346	0.999987	0
max fnr	0.576346	0.949051	0
max fpr	0.000025	1.0	399
max tpr	0.000025	1.0	399

```
GainsLift Table: Avg response rate: 6.59 %, avg score: 1.87 %
```

group	cumulative_data_fraction	lift	cumulative_lift	response_rate	score	cumulative_response_rate	cumulative_score	capture_rate	cumulative_capture_rate	gain	cumulative_gain	kolmogorov_smirnov
1	0.010023	3.206355	3.206355	0.015893	0.121535	0.116393	0.121535	0.032127	0.032127	220.635501	220.635501	0.031512
2	0.020046	2.608696	2.907525	0.012931	0.080598	0.014397	0.101067	0.026256	0.058383	160.869565	190.752533	0.056895
3	0.030069	2.050351	2.621729	0.010164	0.064250	0.013000	0.086266	0.020725	0.079108	105.035088	162.172933	0.076966
4	0.040092	1.863354	2.436957	0.009237	0.055968	0.008821	0.076900	0.018635	0.097743	86.335404	143.695652	0.094177
5	0.050008	1.620464	2.321442	0.008076	0.047533	0.007477	0.069972	0.016174	0.113916	62.046372	132.144206	0.108236
6	0.100015	1.313793	2.067617	0.006545	0.038168	0.006334	0.053953	0.065544	0.179461	31.379311	106.761688	0.171872
7	0.150007	0.921060	1.685997	0.004587	0.031597	0.003475	0.044513	0.045855	0.225316	-7.894026	68.599688	0.202455
8	0.200005	0.807486	1.507500	0.004021	0.026811	0.003242	0.038187	0.040351	0.265667	-19.251400	50.750000	0.225000
9	0.300007	0.621847	1.238616	0.003097	0.021908	0.002611	0.031502	0.062197	0.327863	-37.815303	23.861600	0.263267
10	0.400009	0.518208	1.077500	0.002580	0.019127	0.002317	0.027411	0.051813	0.379677	-48.179200	7.750000	0.283333
11	0.500011	0.414567	0.966000	0.002064	0.017488	0.002117	0.024900	0.041522	0.421199	-58.543300	-3.400000	0.290000
12	0.600013	0.414567	0.897000	0.002064	0.016492	0.001917	0.023488	0.041522	0.462721	-58.543300	-10.300000	0.290000
13	0.700015	0.414567	0.841000	0.002064	0.015800	0.001917	0.022600	0.041522	0.504242	-58.543300	-15.900000	0.290000
14	0.800017	0.414567	0.793000	0.002064	0.015298	0.001917	0.021900	0.041522	0.545763	-58.543300	-20.700000	0.290000
15	0.900019	0.414567	0.750000	0.002064	0.014900	0.001917	0.021300	0.041522	0.587284	-58.543300	-25.000000	0.290000
16	1.000000	0.412284	0.700000	0.002053	0.014600	0.001906	0.020800	0.041284	0.628568	-58.771600	-30.000000	0.290000

Cross-Validation Metrics Summary:

	mae	rmse	mse	nobs	r2	mean_residual_deviance	rmsle	aic	pr_auc	lift_top_group	mean_per_class_error	logloss
accuracy	0.067444	0.258177	0.066642	87889.0	-0.07969	0.066642	0.07969	0.067176				
auc	0.067444	0.267192	0.071393	87889.0	-0.07969	0.071393	0.07969	0.067176	0.454851	3.206355	0.429155	0.242416
aucpr	0.015512	0.125177	0.015703	87889.0	0.038798	0.015703	0.038798	0.023902				
aucpr	0.015512	0.125177	0.015703	87889.0	0.038798	0.015703	0.038798	0.023902	0.000000	3.206355	0.000000	0.000000
mean_per_class_error	0.429155	0.000000	0.000000	87889.0	1.000000	0.000000	1.000000	0.429155	0.429155	0.000000	0.429155	0.429155
rmse	0.258177	0.000000	0.000000	87889.0	1.000000	0.000000	1.000000	0.258177	0.258177	0.000000	0.258177	0.258177
mse	0.066642	0.000000	0.000000	87889.0	1.000000	0.000000	1.000000	0.066642	0.066642	0.000000	0.066642	0.066642
nobs	87889.0	0.000000	0.000000	87889.0	NaN	0.000000	NaN	87889.0	87889.0	87889.0	87889.0	87889.0
r2	-0.07969	0.000000	0.000000	87889.0	-0.07969	0.000000	-0.07969	-0.07969	-0.07969	-0.07969	-0.07969	-0.07969
rmsle	0.067176	0.000000	0.000000	87889.0	-0.07969	0.000000	0.07969	0.067176	0.067176	0.000000	0.067176	0.067176
aic	0.067176	0.000000	0.000000	87889.0	-0.07969	0.000000	0.07969	0.067176	0.067176	0.000000	0.067176	0.067176
lift_top_group	3.206355	0.000000	0.000000	87889.0	0.000000	0.000000	0.000000	3.206355	3.206355	0.000000	3.206355	3.206355
pr_auc	0.015512	0.125177	0.015703	87889.0	0.038798	0.015703	0.038798	0.023902	0.000000	3.206355	0.000000	0.000000

Scoring History:

timestamp	duration	number_of_trees	training_rmse	training_logloss	training_auc	training_pr_auc	training_lift	training_classification_error	validation_rmse	validation_logloss	validation_auc	validation_pr_auc	validation_lift	validation_classification_error
2022-12-12 19:34	2 min 32.842 sec	0.0	0.258937	0.687553	0.5	0.011512	1.0	0.988498	0.258177	0.687553	0.5	0.011512	1.0	0.988498
2022-12-12 19:34	2 min 37.829 sec	1.0	0.248303	0.612673	0.701113	0.021743	3.206355	0.127631	0.247867	0.612423	0.727208	0.028241	3.606177	0.127631
2022-12-12 19:34	2 min 42.853 sec	2.0	0.248207	0.611967	0.707134	0.022179	3.206355	0.127631	0.247866	0.612419	0.726857	0.028047	3.606177	0.127631
2022-12-12 19:34	2 min 23.724 sec	3.0	0.248121	0.608908	0.710273	0.024104	3.606177	0.124114	0.247866	0.612419	0.726824	0.028008	3.272451	0.124455
2022-12-12 19:34	2 min 19.489 sec	4.0	0.248104	0.608292	0.711006	0.024104	3.606177	0.124114	0.247866	0.612419	0.726821	0.028001	3.272451	0.124455
2022-12-12 19:34	2 min 41.087 sec	5.0	0.247449	0.593860	0.733359	0.028398	3.606177	0.120345	0.247843	0.611945	0.727045	0.027945	3.272451	0.124455
2022-12-12 19:34	2 min 41.687 sec	6.0	0.247408	0.592864	0.734197	0.028529	3.606177	0.120345	0.247842	0.611941	0.727045	0.027945	3.272451	0.124455
2022-12-12 19:34	2 min 41.687 sec	44.0	0.247175	0.586498	0.747988	0.031435	3.606177	0.120345	0.247842	0.611941	0.727045	0.027945	3.272451	0.124455

Variable Importances:

variable	relative_importance	scaled_importance	percentage
Dest	1962.000347	1.000	0.203560
Airline	1931.858375	0.985	0.200437
Origin	1754.440164	0.894	0.182000
Month	1654.513254	0.843	0.171639
DayOfWeek	1.000000	0.000	0.000104
Distance	0.000000	0.000	0.000000

In [10]:
```
# Graph
# ntree = 50
ntree = 50
h2o.display.toggle_user_tips(1) # switch on/off this setup.

gbm = gbm.fit(x=X,y=y,ntrees=ntree)
gbm
```

```
Model Details
=============
H2OGradientBoostingEstimator : Gradient Boosting Machine
Model Key: GBM_model_python_1678891881988_824

Model Summary:
```

number_of_trees	number_of_internal_trees	model_size_in_bytes	min_depth	max_depth	mean_depth	min_leaves	max_leaves	mean_leaves
44.0	44.0	14586.0	3.0	3.0	3.0000182	12.0	8.0	12.04908

```
ModelMetrics(nfolds=5): gbm
** Reported on cross-validation data. **

MSE: 0.07134873285761024
RMSE: 0.26711024422000344
LogLoss: 0.2424164267924188
Mean Per-Class Error: 0.42891549762885174
AUC: 0.7577192000000001
AUCPR: 0.4548511217860211
Gini: 0.5154384000000002

Confusion Matrix (vertical) for max F1 @ threshold = 0.03543775002732791:
```

	False	True	Error	Rate
False	67215.0	15884.0	0.1243	=15884/82099
True	4215.0	1575.0	0.7421	=4215/5790
Total	71430.0	17459.0	0.2108	=20099/87889

```
Maximum Metrics: Maximum metrics at their respective thresholds
```

metric	threshold	value	idx
max f1	0.035437	0.172721	351
max f2	0.014735	0.363910	393
max f0point5	0.076346	0.116341	183
max accuracy	0.576346	0.933641	0
max precision	0.576346	0.666667	12
max recall	0.000025	1.0	399
max specificity	0.576346	0.999987	0
max absolute_mcc	0.035437	0.150365	351
max min_per_class_accuracy	0.014735	0.704616	393
max mean_per_class_accuracy	0.012108	0.752701	399
max tns	0.576346	84098.0	0
max fns	0.576346	5494.0	0
max fps	0.000025	84099.0	399
max tps	0.000025	5790.0	399
max tnr	0.576346	0.999987	0
max fnr	0.576346	0.949051	0
max fpr	0.000025	1.0	399
max tpr	0.000025	1.0	399

```
GainsLift Table: Avg response rate: 6.59 %, avg score: 1.87 %
```

group	cumulative_data_fraction	lift	cumulative_lift	response_rate	score	cumulative_response_rate	cumulative_score	capture_rate	cumulative_capture_rate	gain	cumulative_gain	kolmogorov_smirnov
1	0.010023	3.206355	3.206355	0.015893	0.121535	0.116393	0.121535	0.032127	0.032127	220.635501	220.635501	0.031512
2	0.020046	2.608696	2.907525	0.012931	0.080598	0.014397	0.101067	0.026256	0.058383	160.869565	190.752533	0.056895
3	0.030069	2.050351	2.621729	0.010164	0.064250	0.013000	0.086266	0.020725	0.079108	105.035088	162.172933	0.076966
4	0.040092	1.863354	2.436957	0.009237	0.055968	0.008821	0.076900	0.018635	0.097743	86.335404	143.695652	0.094177
5	0.050008	1.620464	2.321442	0.008076	0.047533	0.007477	0.069972	0.016174	0.113916	62.046372	132.144206	0.108236
6	0.100015	1.313793	2.067617	0.006545	0.038168	0.006334	0.053953	0.065544	0.179461	31.379311	106.761688	0.171872
7	0.150007	0.921060	1.685997	0.004587	0.031597	0.003475	0.044513	0.045855	0.225316	-7.894026	68.599688	0.202455
8	0.200005	0.807486	1.507500	0.004021	0.026811	0.003242	0.038187	0.040351	0.265667	-19.251400	50.750000	0.225000
9	0.300007	0.621847	1.238616	0.003097	0.021908	0.002611	0.031502	0.062197	0.327863	-37.815303	23.861600	0.263267
10	0.400009	0.518208	1.077500	0.002580	0.019127	0.002317	0.027411	0.051813	0.379677	-48.179200	7.750000	0.283333
11	0.500011	0.414567	0.966000	0.002064	0.017488	0.002117	0.024900	0.041522	0.421199	-58.543300	-3.400000	0.290000
12	0.600013	0.414567	0.897000	0.002064	0.016492	0.001917	0.023488	0.041522	0.462721	-58.543300	-10.300000	0.290000
13	0.700015	0.414567	0.841000	0.002064	0.015800	0.001917	0.022600	0.041522	0.504242	-58.543300	-15.900000	0.290000
14	0.800017	0.414567	0.793000	0.002064	0.015298	0.001917	0.021900	0.041522	0.545763	-58.543300	-20.700000	0.290000
15	0.900019	0.414567	0.750000	0.002064	0.014900	0.001917	0.021300	0.041522	0.587284	-58.543300	-25.000000	0.290000
16	1.000000	0.412284	0.700000	0.002053	0.014600	0.001906	0.020800	0.041284	0.628568	-58.771600	-30.000000	0.290000

```
ModelMetrics(nfolds=5): gbm
** Reported on cross-validation data. **

MSE: 0.07134873285761024
RMSE: 0.26711024422000344
LogLoss: 0.2424164267924188
Mean Per-Class Error: 0.42891549762885174
AUC: 0.7577192000000001
AUCPR: 0.4548511217860211
Gini: 0.5154384000000002

Confusion Matrix (vertical) for max F1 @ threshold = 0.035437500273279:
```

	False	True	Error	Rate
False	67215.0	15884.0	0.1243	=15884/82099
True	4215.0	1575.0	0.7421	=4215/5790
Total	71430.0	17459.0	0.2108	=20099/87889

```
Maximum Metrics: Maximum metrics at their respective thresholds
```

metric	threshold	value	idx
max f1	0.035437	0.172721	351
max f2	0.014735	0.363910	393
max f0point5	0.076346	0.116341	183
max accuracy	0.576346	0.933641	0
max precision	0.576346	0.666667	12
max recall	0.000025	1.0	399
max specificity	0.576346	0.999987	0
max absolute_mcc	0.035437	0.150365	351
max min_per_class_accuracy	0.014735	0.704616	393
max mean_per_class_accuracy	0.012108	0.752701	399
max tns	0.576346	84098.0	0
max fns	0.576346	5494.0	0
max fps	0.000025	84099.0	399
max tps	0.000025	5790.0	399
max tnr	0.576346	0.999987	0
max fnr	0.576346	0.949051	0
max fpr	0.000025	1.0	399
max tpr	0.000025	1.0	399

```
GainsLift Table: Avg response rate: 6.59 %, avg score: 1.87 %
```

group	cumulative_data_fraction	lift	cumulative_lift	response_rate	score	cumulative_response_rate	cumulative_score	capture_rate	cumulative_capture_rate	gain	cumulative_gain	kolmogorov_smirnov
1	0.010023	3.206355	3.206355	0.015893	0.121535	0.116393	0.121535	0.032127	0.032127	220.635501	220.635501	0.031512
2	0.020046	2.608696	2.907525	0.012931	0.080598	0.014397	0.101067	0.026256	0.058383	160.869565	190.752533	0.056895
3	0.030069	2.050351	2.621729	0.010164	0.064250	0.013000	0.086266	0.020725	0.079108	105.035088	162.172933	0.076966
4	0.040092	1.863354	2.436957	0.009237	0.055968	0.008821	0.076900	0.018635	0.097743	86.335404	143.695652	0.094177
5	0.050008	1.620464	2.321442	0.008076	0.047533	0.007477	0.069972	0.016174	0.113916	62.046372	132.144206	0.108236
6	0.100015	1.313793	2.067617	0.006545	0.038168	0.006334	0.053953	0.065544	0.179461	31.379311	106.761688	0.171872
7	0.150007	0.921060	1.685997	0.004587	0.031597	0.003475	0.044513	0.045855	0.225316	-7.894026	68.599688	0.202455
8	0.200005	0.807486	1.507500	0.004021	0.026811	0.003242	0.038187	0.040351	0.265667	-19.251400	50.750000	0.225000
9	0.300007	0.621847	1.238616	0.003097	0.021908	0.002611	0.031502	0.062197	0.327863	-37.815303	23.861600	0.263267
10	0.400009	0.518208	1.077500	0.002580	0.019127	0.002317	0.027411	0.051813	0.379677	-48.179200	7.750000	0.283333
11	0.500011	0.414567	0.966000	0.002064	0.017488	0.002117	0.024900	0.041522	0.421199	-58.543300	-3.400000