Are We There Yet?

Modeling Late Flights from NYC

Larry Jackelen

Data Science Candidate
Delta Silver Medallion Holder

Problem Statement Background

Inconsistencies in a flight’s scheduled and experienced itinerary can have a butterfly affect on important business travel, vacations, time-sensitive situations, and everything else in its path!

User-side, understanding the possibility of a late flight can allow contingencies to be made.

Business-side, saving operations costs and headaches as well as maintaining a great customer experience.

No matter how you slice it, understanding how and when late flights happen is important

Problem Statement

Given flight dataset detailing flights from New York City airports (JFK, Newark, LaGuardia) from January 2013 until July 2013, create a model for predicting late flights.

Data

Flight dataset provided by LexisNexis

Training dataset of January - June 2013

Test dataset of July 2013

Individual flight information

- Origin and destination
- Flight time and distance
- Scheduled and actual (training only) departure and arrival times
- Carrier

172,000 flights

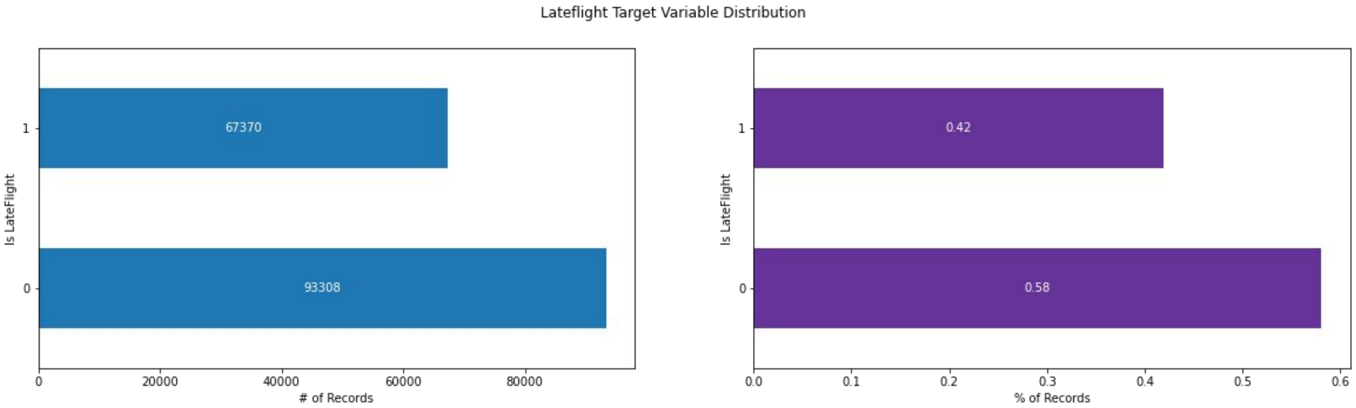
Exploratory Data Analysis

Target Variable

lateflight is defined as a flight arriving after the scheduled arrival time

This feature needed to be engineered in the training set using the variable “arrival delay” representing the amount of delay in minutes.

Checking for completeness of the “arrival delay” feature revealed some nulls (that couldn’t be imputed) leading to those flights being dropped due to lack of target variable value.

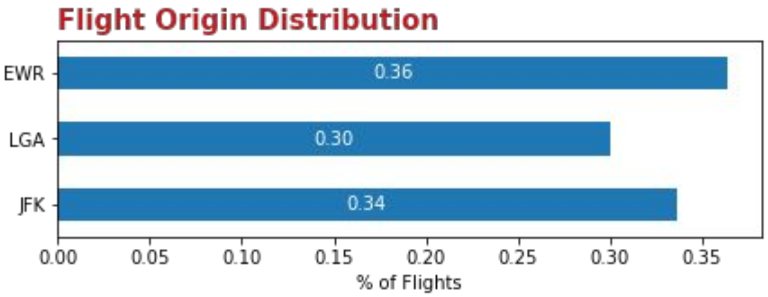


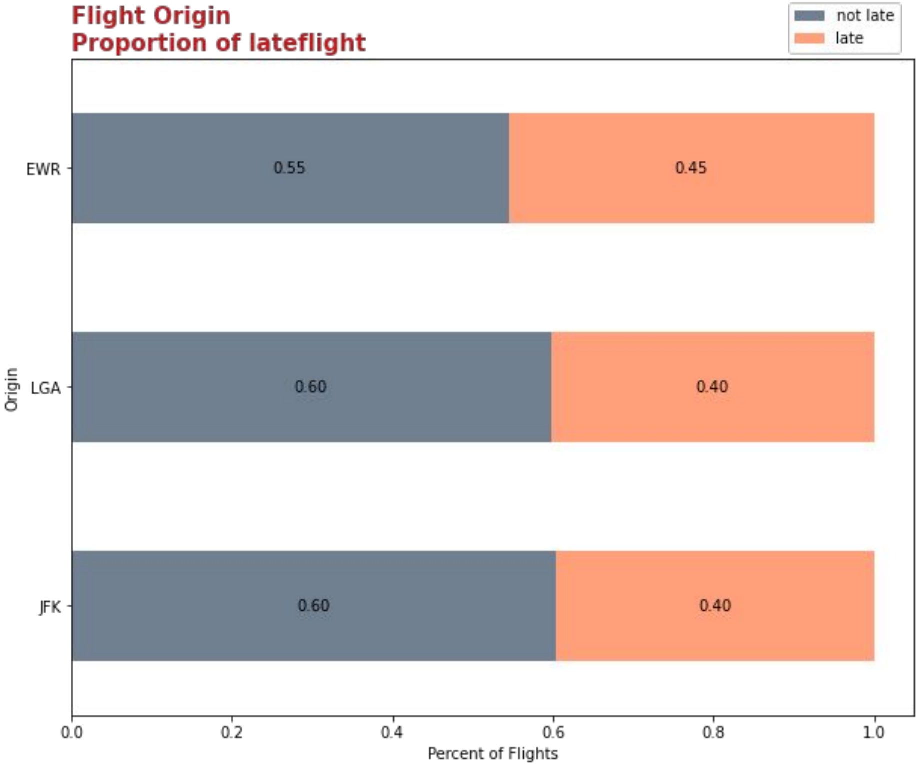
The Big Three

The dataset was roughly equally divided between the three origin airports in NYC

Newark has the most flights as well as the highest laterate

These features were dummied in the dataset



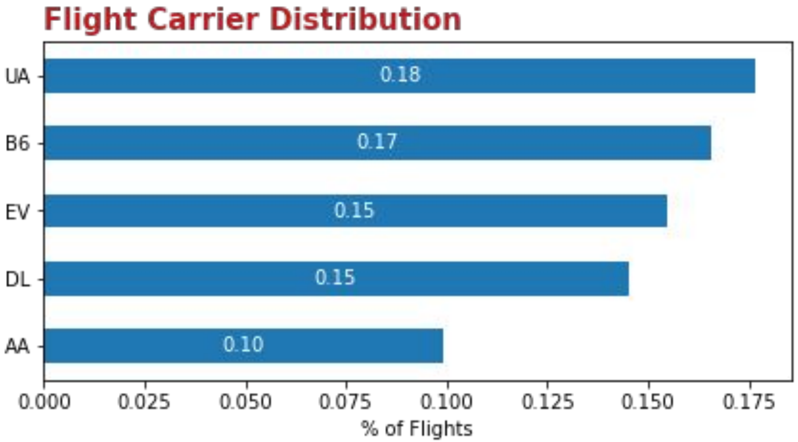


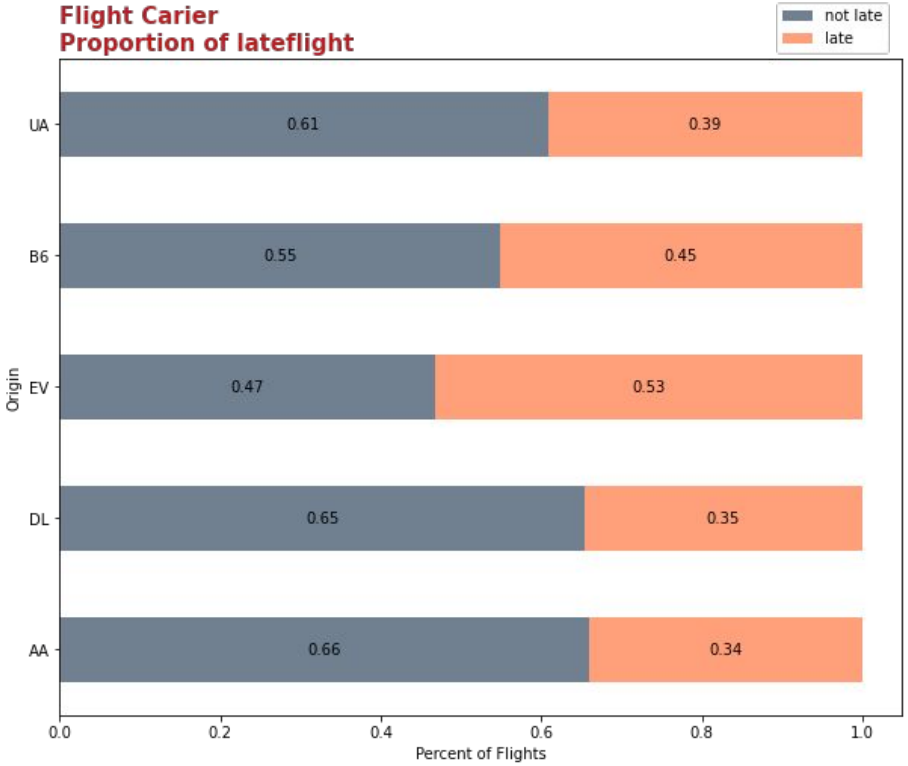
Carriers

The dataset had 5 carriers that comprised 10% or more of the flights

United had the most flights while ExpressJet Airlines saw the most late flights out of the 5

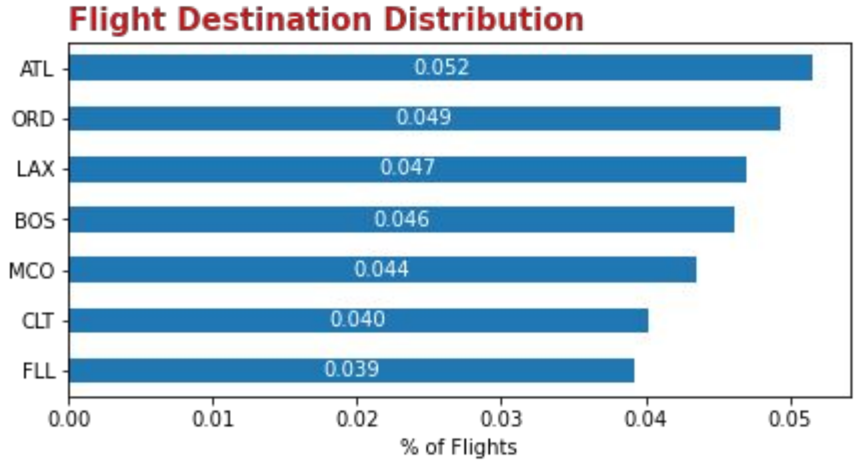
These larger carrier features were dummied in the dataset

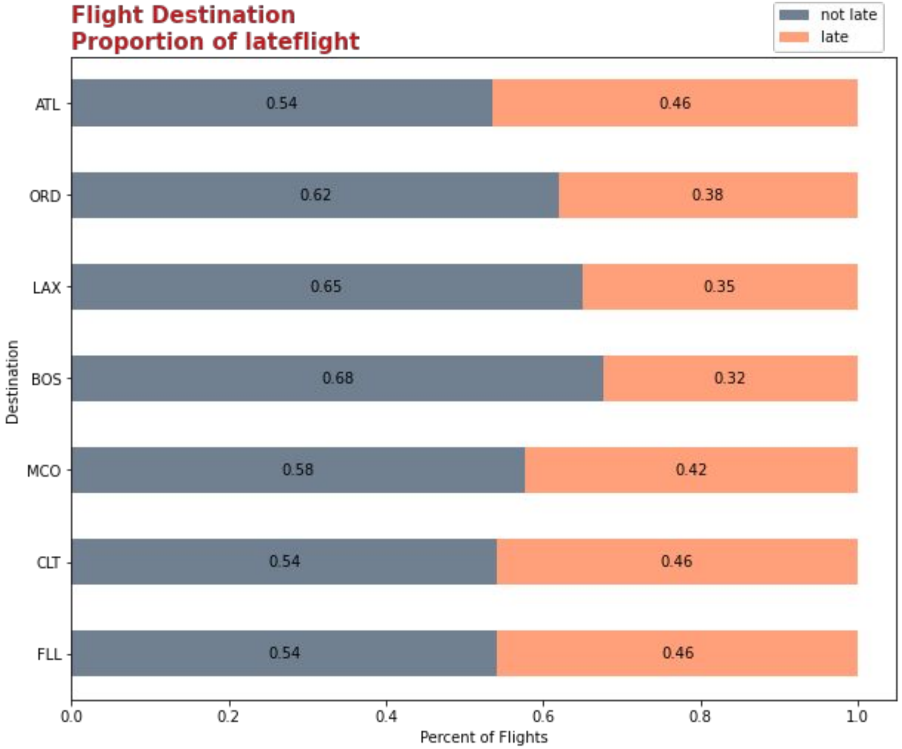




Popular Destinations

The largest destinations in the dataset topped off at 5%. To understand more about them rather than destinations entirely, destinations with a 4% or more share were dummied.





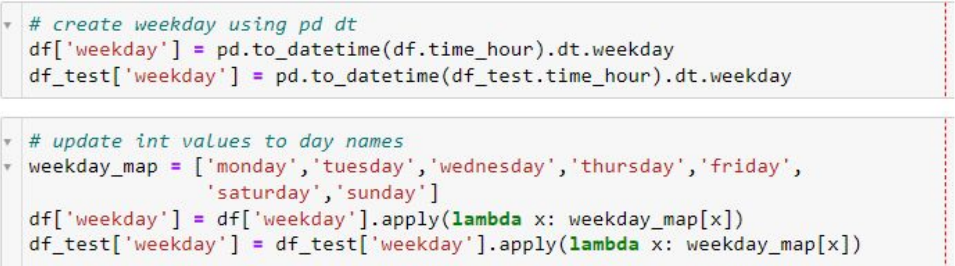
Meta-variables

Weekday

Rather than rely on the day number within a month, the weekday of the flights was considered

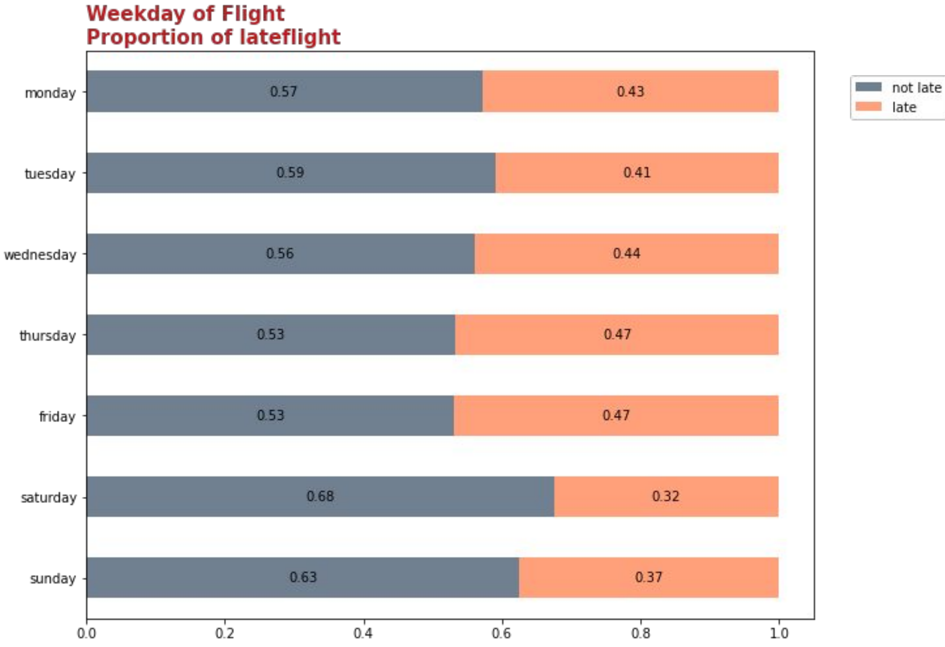
Saturdays were the lowest flight days while the rest were fairly even at 14-15% share.

Saturdays and Sundays saw the lowest number of late flights



```
# create weekday using pd dt
df['weekday'] = pd.to_datetime(df.time_hour).dt.weekday
df_test['weekday'] = pd.to_datetime(df_test.time_hour).dt.weekday
```

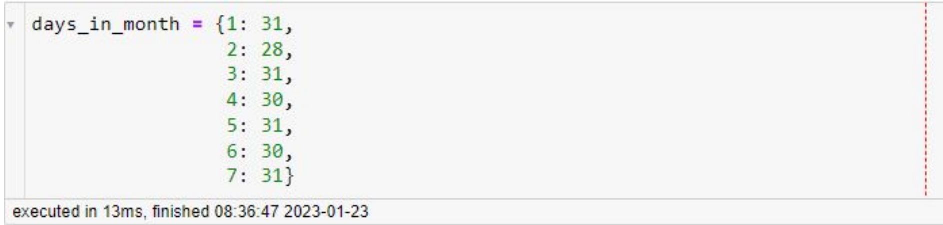
```
# update int values to day names
weekday_map = ['monday','tuesday','wednesday','thursday','friday',
              'saturday','sunday']
df['weekday'] = df['weekday'].apply(lambda x: weekday_map[x])
df_test['weekday'] = df_test['weekday'].apply(lambda x: weekday_map[x])
```



Month Penetration

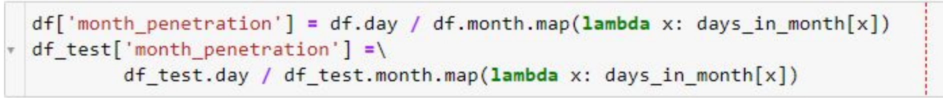
Similar to the addition of the weekday variable, month penetration was added to relate the flight's date to the lateness of the month.

This is different than just using day since there are an unequal number of days across all months



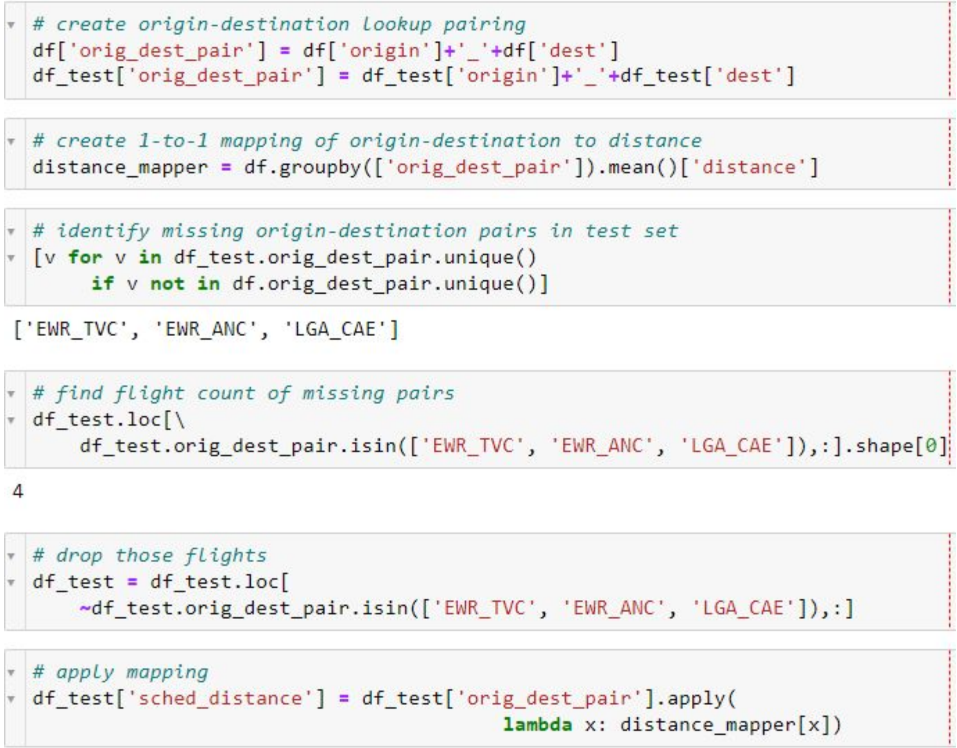
```
days_in_month = {1: 31,
                 2: 28,
                 3: 31,
                 4: 30,
                 5: 31,
                 6: 30,
                 7: 31}
```

executed in 13ms, finished 08:36:47 2023-01-23



```
df['month_penetration'] = df.day / df.month.map(lambda x: days_in_month[x])
df_test['month_penetration'] =\
        df_test.day / df_test.month.map(lambda x: days_in_month[x])
```

Scheduled Flight Distance



```
# create origin-destination lookup pairing
df['orig_dest_pair'] = df['origin']+'_'+df['dest']
df_test['orig_dest_pair'] = df_test['origin']+'_'+df_test['dest']
```

```
# create 1-to-1 mapping of origin-destination to distance
distance_mapper = df.groupby(['orig_dest_pair']).mean()['distance']
```

```
# identify missing origin-destination pairs in test set
[v for v in df_test.orig_dest_pair.unique()
     if v not in df.orig_dest_pair.unique()]
```

```
['EWR_TVC', 'EWR_ANC', 'LGA_CAE']
```

```
# find flight count of missing pairs
df_test.loc[\
    df_test.orig_dest_pair.isin(['EWR_TVC', 'EWR_ANC', 'LGA_CAE']),:].shape[0]
```

```
4
```

```
# drop those flights
df_test = df_test.loc[
    ~df_test.orig_dest_pair.isin(['EWR_TVC', 'EWR_ANC', 'LGA_CAE']),:]
```

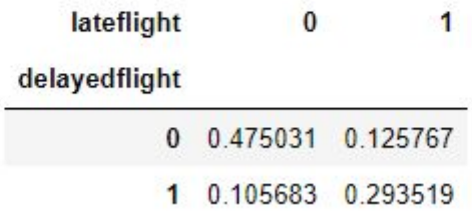
```
# apply mapping
df_test['sched_distance'] = df_test['orig_dest_pair'].apply(
                                        lambda x: distance_mapper[x])
```

Adding the distance of the flight is a numerical way of grouping destinations which were seen to have too many to dummy

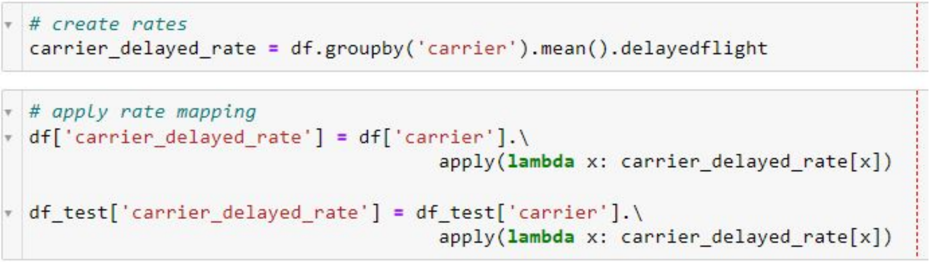
The test set did need to drop 4 records but overall had really good completeness with the training set

% Flights Delayed, Late by Carrier, Dest

Looking at the relationship between flights that were initially delayed and ultimately became late, understanding these rates at the origin, destination, and carrier level became important



lateflight	0	1
delayedflight		
0	0.475031	0.125767
1	0.105683	0.293519



```
# create rates
carrier_delayed_rate = df.groupby('carrier').mean().delayedflight
```

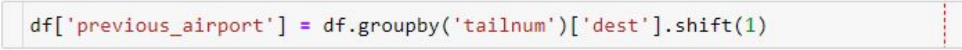
```
# apply rate mapping
df['carrier_delayed_rate'] = df['carrier'].\
                                    apply(lambda x: carrier_delayed_rate[x])

df_test['carrier_delayed_rate'] = df_test['carrier'].\
                                    apply(lambda x: carrier_delayed_rate[x])
```

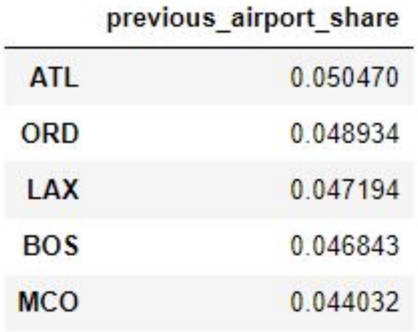
Previous Airport

There could be some value in understanding where the plane was from before arriving in NYC.

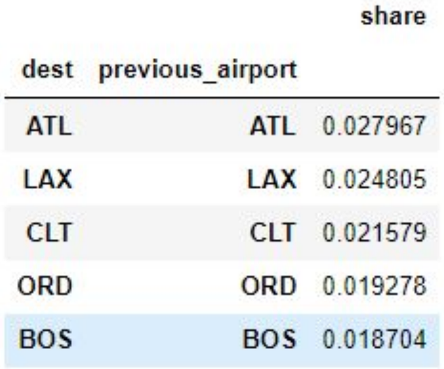
I went back and forth on implementing this variable. Ultimately I decided not to since it would remove 3% of the train dataset and still more records from an already tiny test set



```
df['previous_airport'] = df.groupby('tailnum')['dest'].shift(1)
```



	previous_airport_share
ATL	0.050470
ORD	0.048934
LAX	0.047194
BOS	0.046843
MCO	0.044032



		share
dest	previous_airport	
ATL	ATL	0.027967
LAX	LAX	0.024805
CLT	CLT	0.021579
ORD	ORD	0.019278
BOS	BOS	0.018704

Other Variables Considered

Holidays
Part of me wanted to add a holiday indicator, but the first half of the year really doesn't have that many good holidays that could measure up the fourth of july. If this was year over year, then I'd be more interested

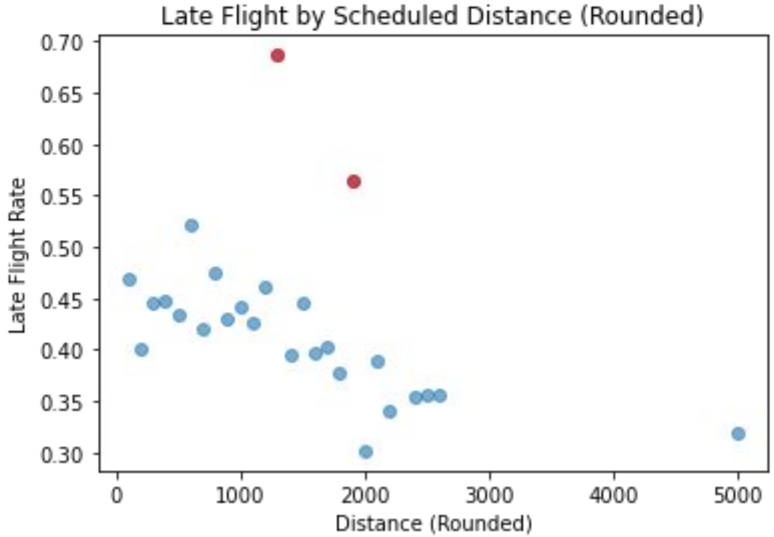
Expected flight time
Connect to geo data and get the appropriate timezone data to allow for expected time of flight by doing a UNIX subtraction

Relative Busyness of Terminal (Origin & Dest)
A flight could be perfectly on time if it wasn't initially delayed. A flight could take off on time and still come in late.

Data Pre-Processing and Modeling

Outliers

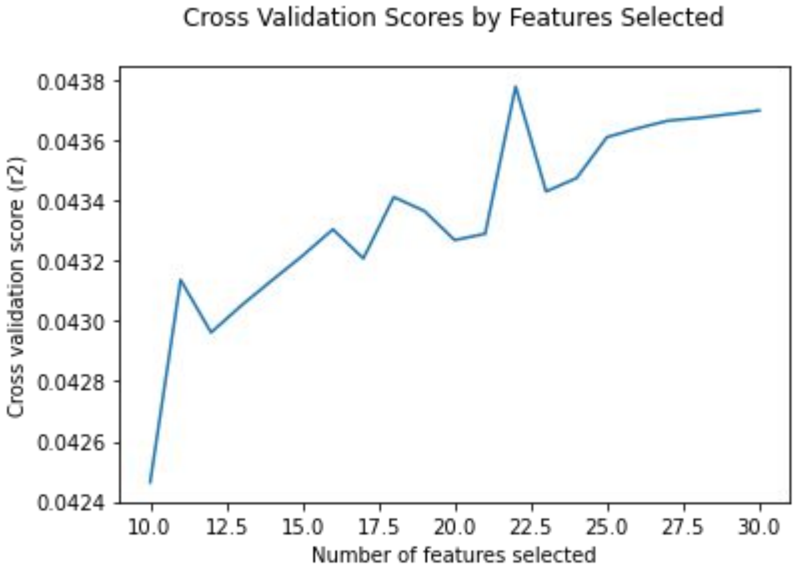
Outliers were looked for across the dataset, but the only action taken was in the scheduled distance with two distances showing much higher late rates and at an amount below the dummying level.



Pre-Processing

1. Standard Scaling

2. Recursive Feature Elimination
    a. Minimum Features Required: 10
    b. BayesianRidge() estimator
    c. 22 features kept, 8 dropped



Model Selection (Baseline = 58%)

Logistic Regression

Test Score of **59.46%**

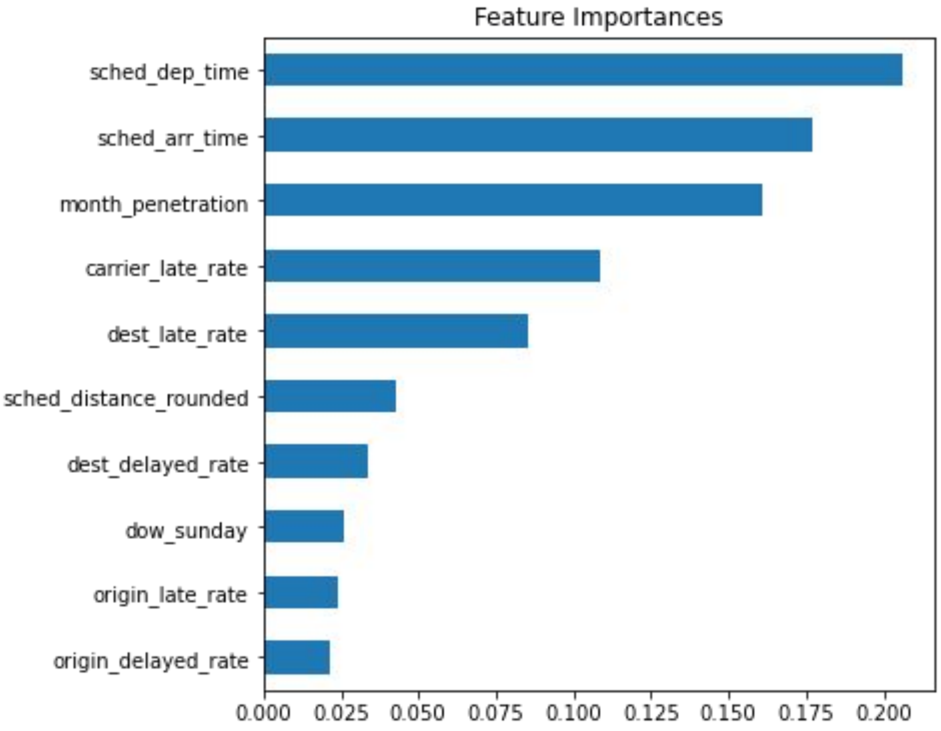
Applied Lasso for feature selection and got a test score of **59.47%**

Random Forest

Grid Searched Hyper-parameters to find:

- N_estimators = 500
- Max depth = 10

Test score of **60.5%**



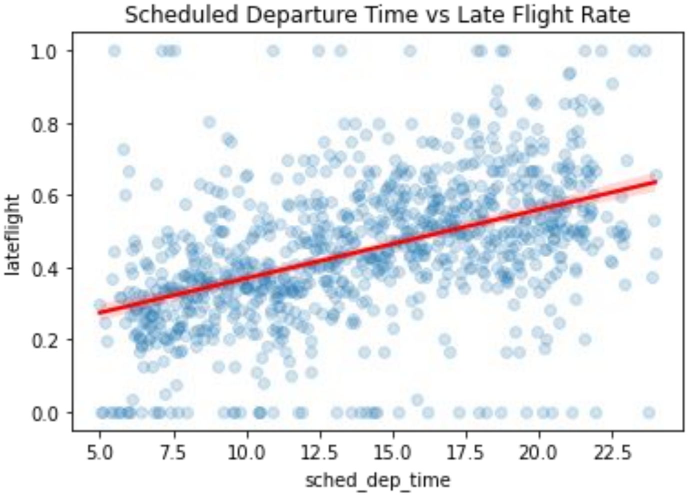
Results and Conclusions

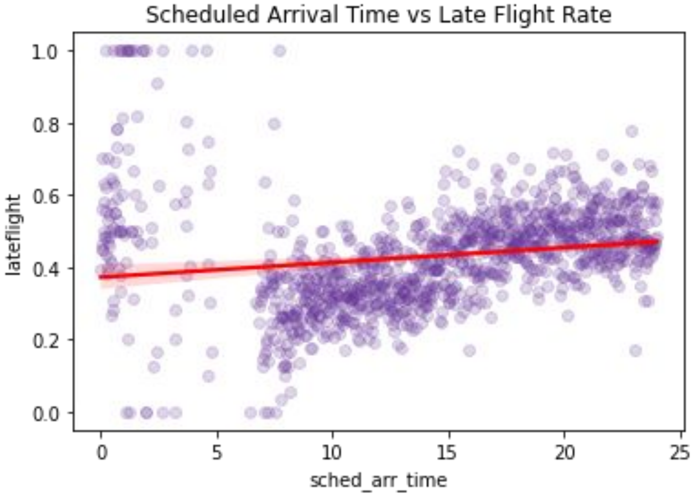
Model Results

The model notes the highest feature importances for the departure and arrival times of the flight.

Looking at the distributions of these features, the later the departure, the more often it is late.

Arrival time isn't as clear but it gets a lot more inconsistent in the early mornings

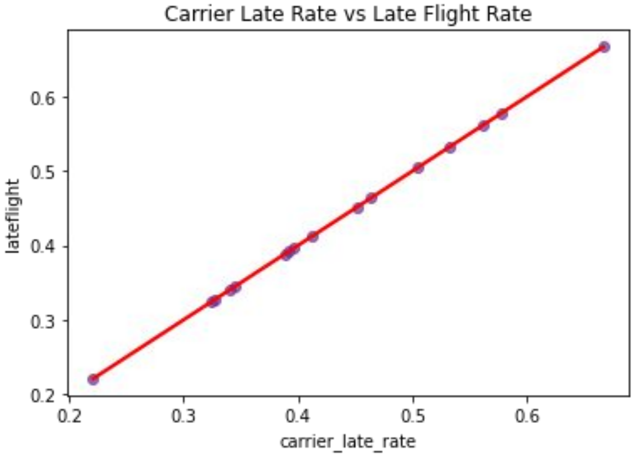


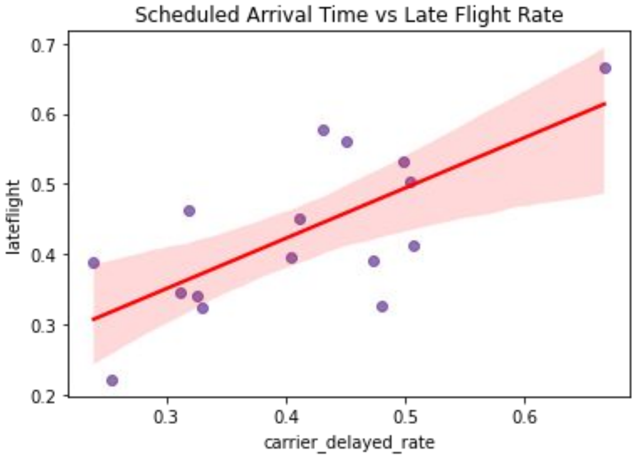


Model Results (cont)

The carrier selected, with their respective late and delayed rates, does give some insight as well. Carriers with higher late rates will tend to have higher expected late flight rates.

Carriers with higher delayed rates are not as clear, but there is a positive relationship





Using the Model

The model really struggles to beat the baseline and relies on a more opaque method to beat it by more than 2%

This model can be used if in the long run you'd like a better idea if your flight will be late rather than assuming the baseline of it never being late

As for business applications, much more info is needed. I would argue that the data given is not enough to create a good model for late flight.

Future Developments

Wind and weather data collection

Tail Speed

Time on Tarmac

Direction of travel

Test set is only 3% of train set, I would review metavariables that inherently leak data and possible do a more train-validation-test approach to help bridge this divide. This will allow for tuning of hyper-parameters for the model without leaking data from those metavariables in the train cross-validation set

Any questions?

Thank you for flying!