Flight Delay Prediction

Feiran Ji

Problem and Data

- Source: the Bureau of Transportation Statistics
- Each row is one flight, including the information of date/time, flight number, carrier, origin airport, destination airport, distance, elapse time. We are going to predict if the flight is going to be delayed for more than 15 minutes.

```
data.head(5)
```

	MONTH	DAY_OF_WEEK	FL_DATE	UNIQUE_CARRIER	FL_NUM	ORIGIN	ORIGIN_CITY_NAME	DEST	DEST_CITY_NAME	CRS_DEP_TIME	ARR_DEL15	CRS_ELAPSED_TIME	DISTANCE	Unnamed: 13
0	2.0	6.0	2017-02-25	B6	28.0	MCO	Orlando, FL	EWR	Newark, NJ	1000.0	0.0	156.0	937.0	NaN
1	2.0	7.0	2017-02-26	B6	28.0	MCO	Orlando, FL	EWR	Newark, NJ	739.0	0.0	153.0	937.0	NaN
2	2.0	1.0	2017-02-27	B6	28.0	MCO	Orlando, FL	EWR	Newark, NJ	1028.0	0.0	158.0	937.0	NaN
3	2.0	2.0	2017-02-28	B6	28.0	MCO	Orlando, FL	EWR	Newark, NJ	739.0	0.0	153.0	937.0	NaN
4	2.0	3.0	2017-02-01	B6	33.0	BTV	Burlington, VT	JFK	New York, NY	1907.0	0.0	90.0	266.0	NaN

Data cleaning

- Delete last column “Unnamed: 13”
- NA in response and CRS_ELAPSED TIME.
  - Delete rows that have null value

```
data.isnull().sum()
```

```
MONTH                    0
DAY_OF_WEEK              0
FL_DATE                  0
UNIQUE_CARRIER           0
FL_NUM                   0
ORIGIN                   0
ORIGIN_CITY_NAME         0
DEST                     0
DEST_CITY_NAME           0
CRS_DEP_TIME             0
ARR_DEL15            71020
CRS_ELAPSED_TIME        10
DISTANCE                 0
Unnamed: 13        5129354
dtype: int64
```

Now…

- Split based on date and look at training set, and do some EDA.

EDA - Response

```
train['ARR_DEL15'].value_counts().to_frame()
```

	ARR_DEL15
0.0	3332525
1.0	714142

EDA - Other categorical features

- 12 unique carriers, 6937 Flight numbers, 311 origin cities, 310 destination cities

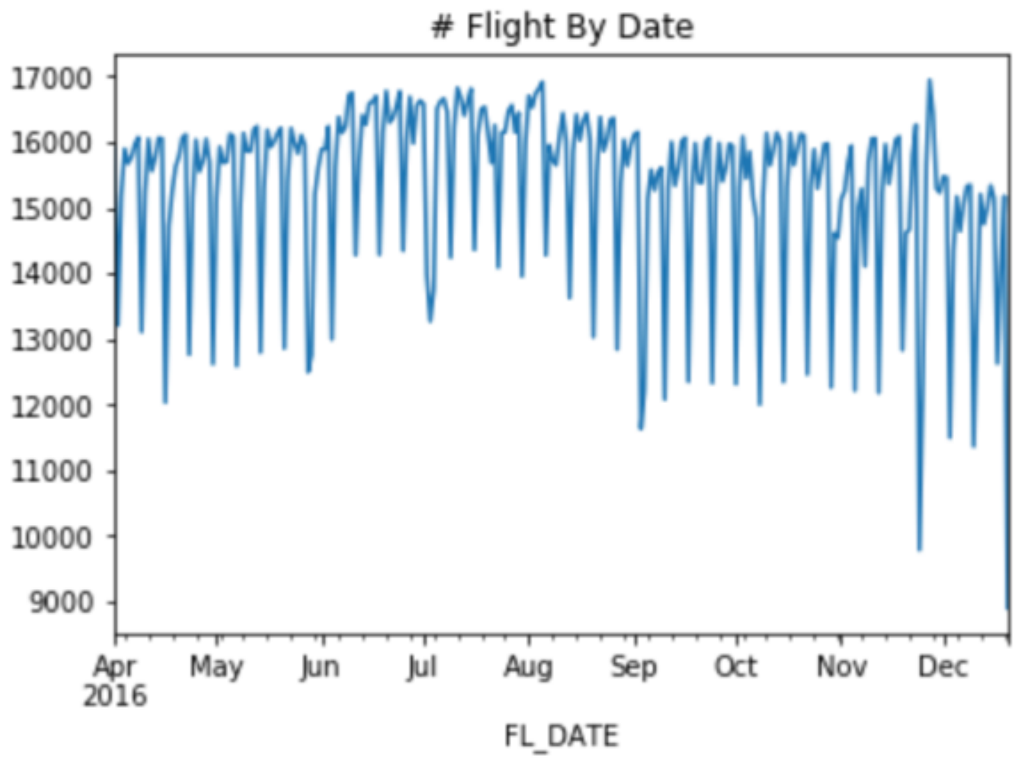
```
train['UNIQUE_CARRIER'].value_counts()

WN    931615
DL    678203
AA    651964
OO    436305
UA    400024
EV    341840
B6    202329
AS    128418
NK     99093
F9     70517
HA     55853
VX     50506
```

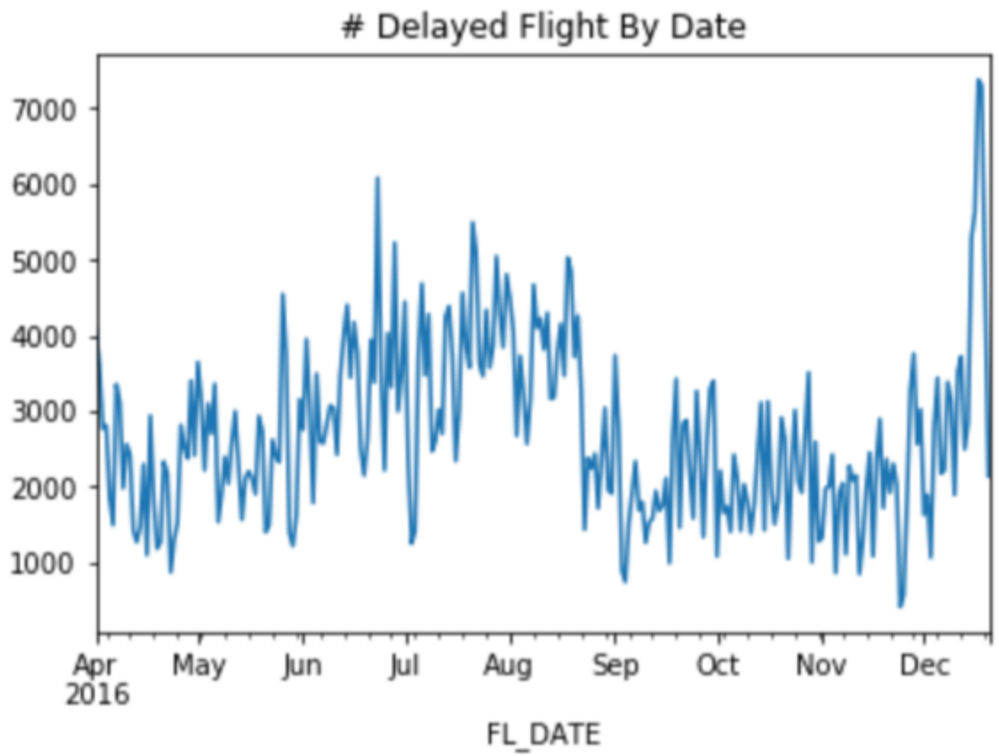
```
train['FL_NUM'].value_counts()

6809.0     1
6801.0     1
6677.0     1
6795.0     1
6678.0     1
6720.0     1
6789.0     1
6525.0     1
6696.0     1
6518.0     1
6512.0     1
6500.0     1
6605.0     1
6494.0     1
6487.0     1
6870.0     1
6871.0     1
6829.0     1
Name: FL_NUM, Length: 6937, dtype: int64
```

EDA – Time Range

FL_DATE	
2016-04-30	456702
2016-05-31	475499
2016-06-30	480916
2016-07-31	491198
2016-08-31	489965
2016-09-30	452558
2016-10-31	467486
2016-11-30	448911
2016-12-31	288435

- All fight information is from Apr 2016 to Feb 2017
- Training set: Apr 2016 – Dec 2016





Feature Engineering

- Date/time related
  - We already have day of week and month
  - We will add day of month and delete month. Because we are predicting Jan 2017 and Feb 2017 but we don’t have data from Jan and Feb 2016.
- Categorical features
  - We will convert most categorical features into numeric features with label encoding. Because we are using tree model, so we don’t need to use one hot encoding.
  - For some categorical features with less classes, we conducted target encoding to convert them.

Model & Performance

In [32]:
```
def set_rf_samples(n):
    """ Changes Scikit learn's random forests to give each tree a random sample of
    n random rows.
    """
    forest._generate_sample_indices = (lambda rs, n_samples:↔
set_rf_samples(50000)
```

In [36]:
```
rf = RandomForestClassifier(n_estimators = 20, max_features = 0.5, min_samples_leaf = 50, n_jobs=-1)
rf.fit(X_train, y_train)
report_score(rf)
```

```
Precision score for val: 0.333 ; Recall score for val: 0.0
Accuracy for train: 0.824 ; Accuracy for val: 0.805
Log Loss for train: 0.43 Log Loss for val: 0.482
```

In [39]:
```
rf.predict(X_val).sum()
```

Out[39]:
```
6.0
```

Upsampling

In [48]:
```
up_train = train.loc[train['ARR_DEL15']==1,:]
train_upsampled = train.append(up_train).append(up_train)
```

In [50]:
```
train.shape
```

Out[50]:
```
(4046667, 19)
```

In [53]:
```
train_upsampled.shape
```

Out[53]:
```
(5474951, 19)
```

Model & Performance

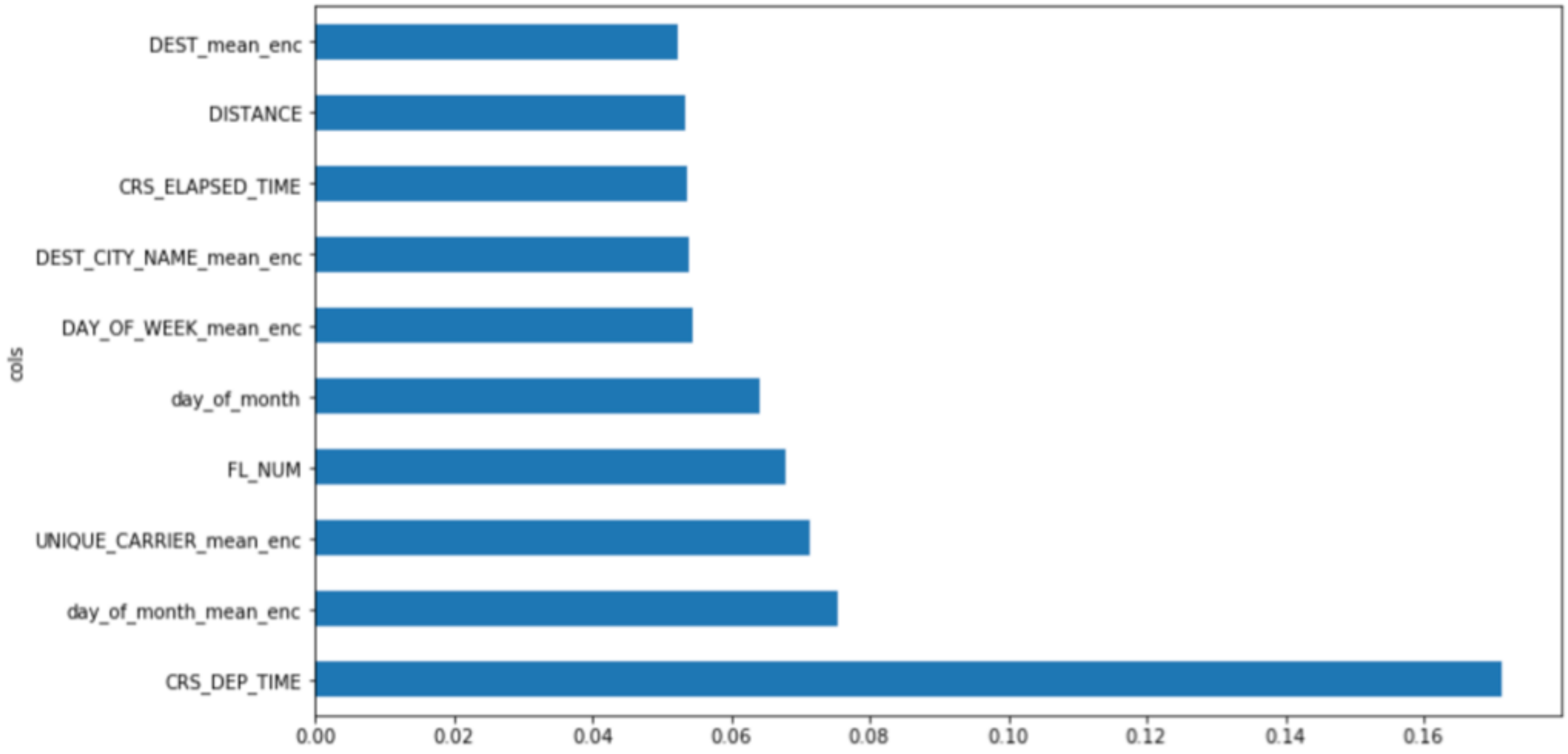
```
rf = RandomForestClassifier(n_estimators = 20, max_features = 0.5, min_samples_leaf = 50, n_jobs=-1)
rf.fit(X_train, y_train)
report_score(rf)
```

```
Precision score for val: 0.284 ; Recall score for val: 0.285
Accuracy for train: 0.671 ; Accuracy for val: 0.721
Log Loss for train: 0.609 Log Loss for val: 0.56
```

```
In [62]: for tn in [20,50,100]:
             for ms in [3,10,50]:
                 print('number of trees:', tn, '; min samples per leaf:', ms)
                 rf = RandomForestClassifier(n_estimators = tn, max_features = 0.5, min_samples_leaf = ms, n_jobs=-1)
                 rf.fit(X_train, y_train)
                 report_score(rf)
```

```
number of trees: 20 ; min samples per leaf: 3
Precision score for val: 0.277 ; Recall score for val: 0.302
Accuracy for train: 0.684 ; Accuracy for val: 0.711 ; F1 score for val: 0.289
Log Loss for train: 0.596 Log Loss for val: 0.569
number of trees: 20 ; min samples per leaf: 10
Precision score for val: 0.294 ; Recall score for val: 0.285
Accuracy for train: 0.681 ; Accuracy for val: 0.728 ; F1 score for val: 0.29
Log Loss for train: 0.599 Log Loss for val: 0.558
number of trees: 20 ; min samples per leaf: 50
Precision score for val: 0.296 ; Recall score for val: 0.28
Accuracy for train: 0.671 ; Accuracy for val: 0.73 ; F1 score for val: 0.288
Log Loss for train: 0.61 Log Loss for val: 0.555
number of trees: 50 ; min samples per leaf: 3
Precision score for val: 0.299 ; Recall score for val: 0.286
Accuracy for train: 0.696 ; Accuracy for val: 0.73 ; F1 score for val: 0.292
Log Loss for train: 0.585 Log Loss for val: 0.557
number of trees: 50 ; min samples per leaf: 10
Precision score for val: 0.297 ; Recall score for val: 0.275
Accuracy for train: 0.686 ; Accuracy for val: 0.732 ; F1 score for val: 0.285
Log Loss for train: 0.595 Log Loss for val: 0.554
number of trees: 50 ; min samples per leaf: 50
Precision score for val: 0.294 ; Recall score for val: 0.271
Accuracy for train: 0.672 ; Accuracy for val: 0.732 ; F1 score for val: 0.282
Log Loss for train: 0.609 Log Loss for val: 0.555
```

Feature Importance & Insights



Next Steps…

- More interpretation
- We only have one year data, so we cannot effectively use features like “month”, ”season”, etc. If we can pull more data from previous years, we may capture those information
- We may try clustering and create more features
- We may join data of weather to improve the performance because delay time may be highly correlated to weather.
- Try time series to capture the trend