



Predicting Flight Departure Delays in an Air Traffic Network

Capstone Project 1
Miguel Montano

The Problem

Potential Clients

- Companies distributing use of the Aircraft Situation Display to Industry (ASDI)
- Air Route Traffic Control Centers (ARTCC) looking to make better predictions of AAR's and plan efficient holding patterns.
- Implementers of Ground Delay Programs at Domestic Airports
- Traffic Management Personnel seeking to get the earliest Expect Departure Clearance Time (EDCT)

Context - Cost of Delay

- \$40 billion in annual cost due to delays in 2015
- \$62.55 per minute average cost of aircraft block (taxi plus airborne) in 2016
- In 2016 nearly 40% of delays were due to the delayed arrival of the incoming aircraft, reflecting the high levels of interdependence in delay dynamics

Problem statement

Predict whether or not the departure delay on an Origin-Destination pair, with a 6-hour prediction horizon, will exceed a 15 min threshold; such that the future delay will fall into one of two classes, 'above threshold' (1) or 'below threshold' (0).

Methodology



Wrangling

1. Improve readability and address missing values
2. Aggregate to create mock network
3. Create target variable of departure delay state 6 hours in the future

Storytelling

1. Flight Frequency
2. Delay Distribution
3. On-Time Performance
4. Network Visualization

Inferential Statistics

1. Test for Normality and CLT in variables of interest
2. Regression Analysis
3. Hypothesis Tests
 - T-Test of Arrival and Departure Deviation Means
 - Permutation Test for Pearson's r between Median Arrival and Departure Delays

Binary Classification

1. Baseline Classifier using Logistic Regression
2. Resample data to address target class imbalance, both under- and over-sampling
3. Train Logistic Regression and Random Forest Classifiers under both resampling conditions

Wrangling

Surface Cleaning and
Pre-Processing

1. Import data and address missing values
 2. Compartmentalize columns and evaluate by category, then merge results into final Flights dataframe
 3. Create NetworkX Digraph from components
 4. Create mock air traffic network, a dataframe named Links_d, containing target class for model construction
-

Flights DataFrame

```
<class 'pandas.core.frame.DataFrame'>
```

```
MultiIndex: 1824403 entries, (ABE-ATL, 2016-01-01 07:00:00) to (YUM-PHX, 2016-12-31 19:15:00)
```

```
Data columns (total 39 columns):
```

| | | | |
|-----------------------|----------------|---------------------|---------|
| quarter | int64 | crs_dep_time | int64 |
| month | int64 | dep_time | float64 |
| day_of_month | int64 | dep_deviation | float64 |
| day_of_week | int64 | dep_delay | float64 |
| fl_date | object | wheels_on | float64 |
| Day_of_Week | object | taxi_in | float64 |
| Month | object | crs_arr_time | int64 |
| dt_index | datetime64[ns] | arr_time | float64 |
| hour_of_day | object | arr_deviation | float64 |
| unique_carrier | object | arr_delay | float64 |
| fl_num | int64 | crs_elapsed_time | float64 |
| origin_airport_id | int64 | actual_elapsed_time | float64 |
| origin_city_market_id | int64 | air_time | float64 |
| origin | object | distance | float64 |
| origin_city_name | object | | |
| origin_state_abr | object | | |
| origin_state_nm | object | | |
| dest_airport_id | int64 | | |
| dest_city_market_id | int64 | | |
| dest | object | | |
| dest_city_name | object | | |
| dest_state_abr | object | | |
| dest_state_nm | object | | |
| link | object | | |
| unique_carrier_nm | object | | |

```
dtypes: datetime64[ns](1), float64(12), int64(11), object(15)
```

```
memory usage: 554.5+ MB
```

Links_d DataFrame

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 882815 entries, (ANC-SEA, 2016-01-01 00:00:00) to (TPA-ATL, 2016-12-31 19:00:00)
Columns: 561 entries, crs_dep_time to dd_binary
dtypes: float64(15), int32(1), uint8(545)
memory usage: 566.6+ MB
None
Index(['crs_dep_time', 'dep_time', 'dep_deviation', 'dep_delay', 'wheels_on',
       'taxi_in', 'crs_arr_time', 'arr_time', 'arr_deviation', 'arr_delay',
       ...,
       'dest_city_name_San Diego', 'dest_city_name_San Jose',
       'dest_city_name_San Juan', 'dest_city_name_Seattle',
       'dest_city_name_St. Louis', 'dest_city_name_Tampa',
       'dest_city_name_Washington',
       'dest_city_name_West Palm Beach/Palm Beach', 'dd_in_6hrs', 'dd_binary'],
      dtype='object', length=561)
```

Storytelling

Exploratory Data Analysis

1. Flight Frequency
 - By Temporal Categories
 - By Commercial Airline Carrier
 - By Location Attributes
 2. Delay Distribution
 - Overall
 - By Commercial Airline Carrier
 - For Highest Traffic Airports
 3. On-Time Performance
 4. Network Visualization
-

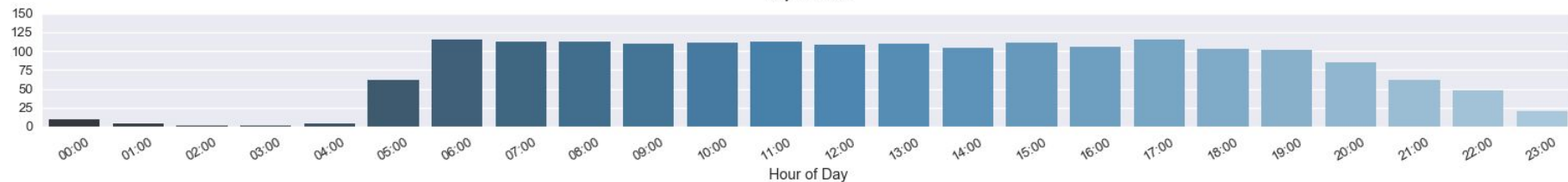
Air Traffic Statistics Explained

- Sample contains all commercial flights in January, August, November, and December of 2016, from the 12 Major Domestic Passenger Airlines.
- The U.S. Department of Transportation defines a major carrier or major airline carrier as a U.S.-based airline that posts more than \$1 billion in revenue during fiscal year, grouped accordingly as “Group III” (“Air Carrier Groupings 2016”, U.S. Bureau of Transportation)
- A flight is counted as “on time” if it operated less than 15 minutes after the scheduled time shown in the carriers’ Computerized Reservation Systems (CRS), with departure and arrival times being calculated from gate to gate, not including taxi or airtime.

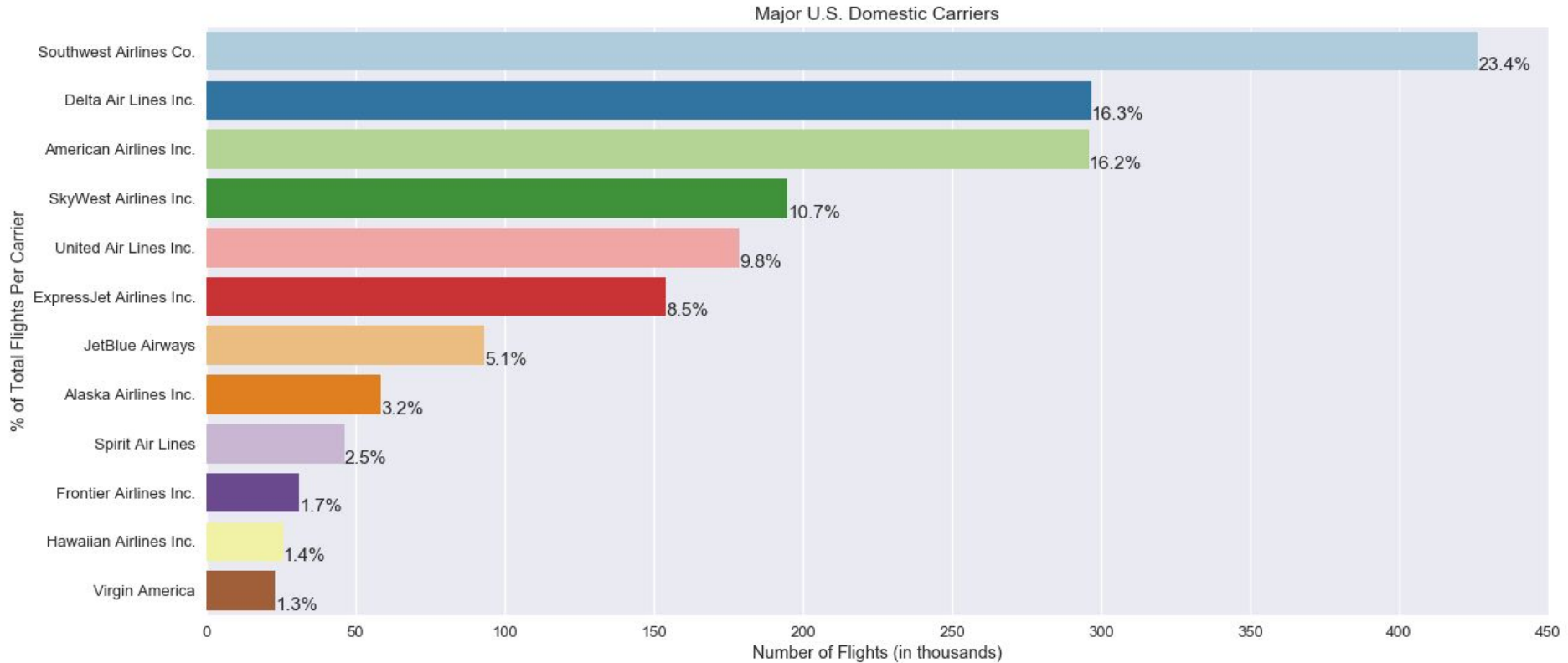


1.1 Frequency by Time

Flight Frequency



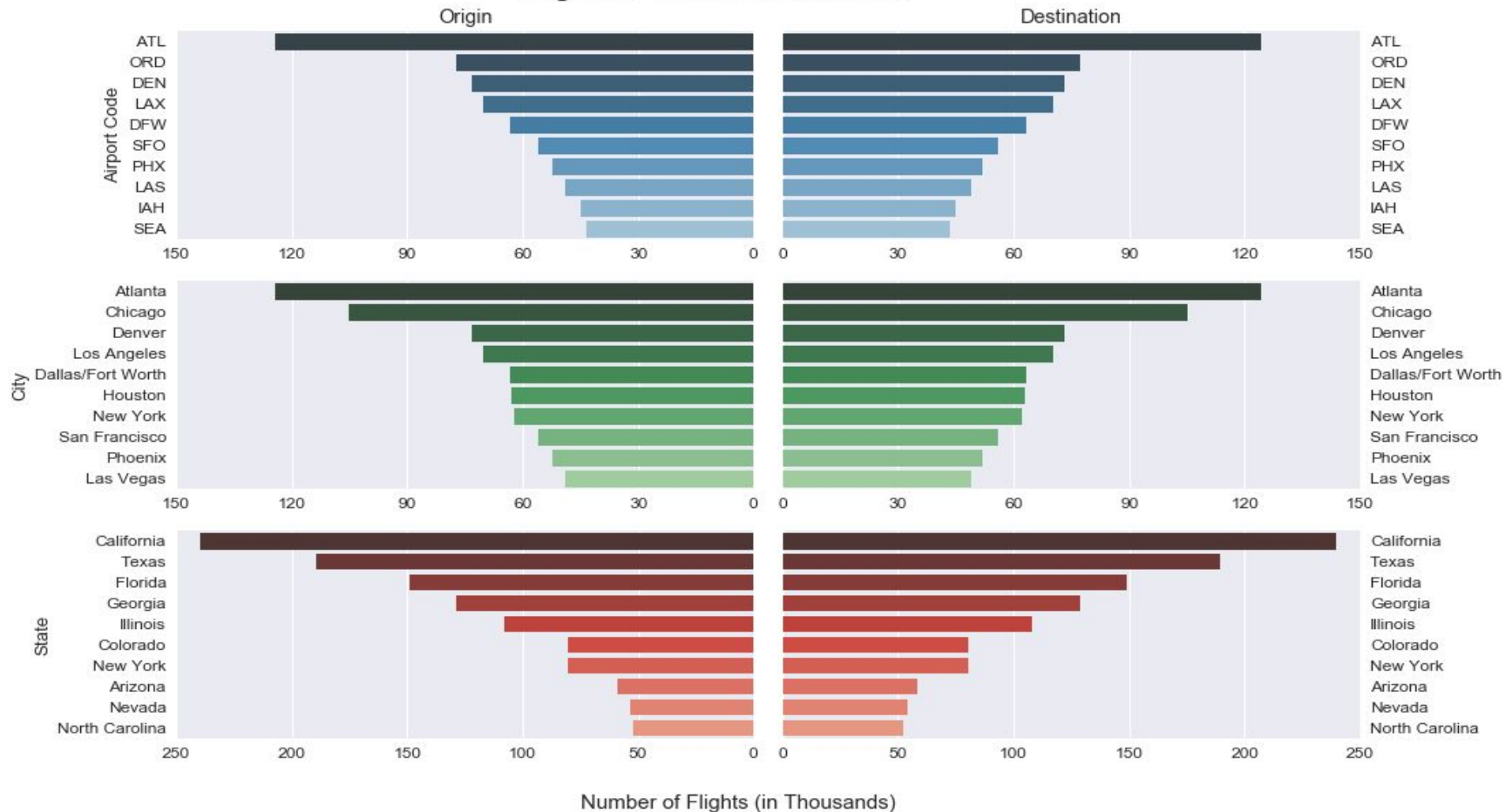
1.2 Frequency by Carrier



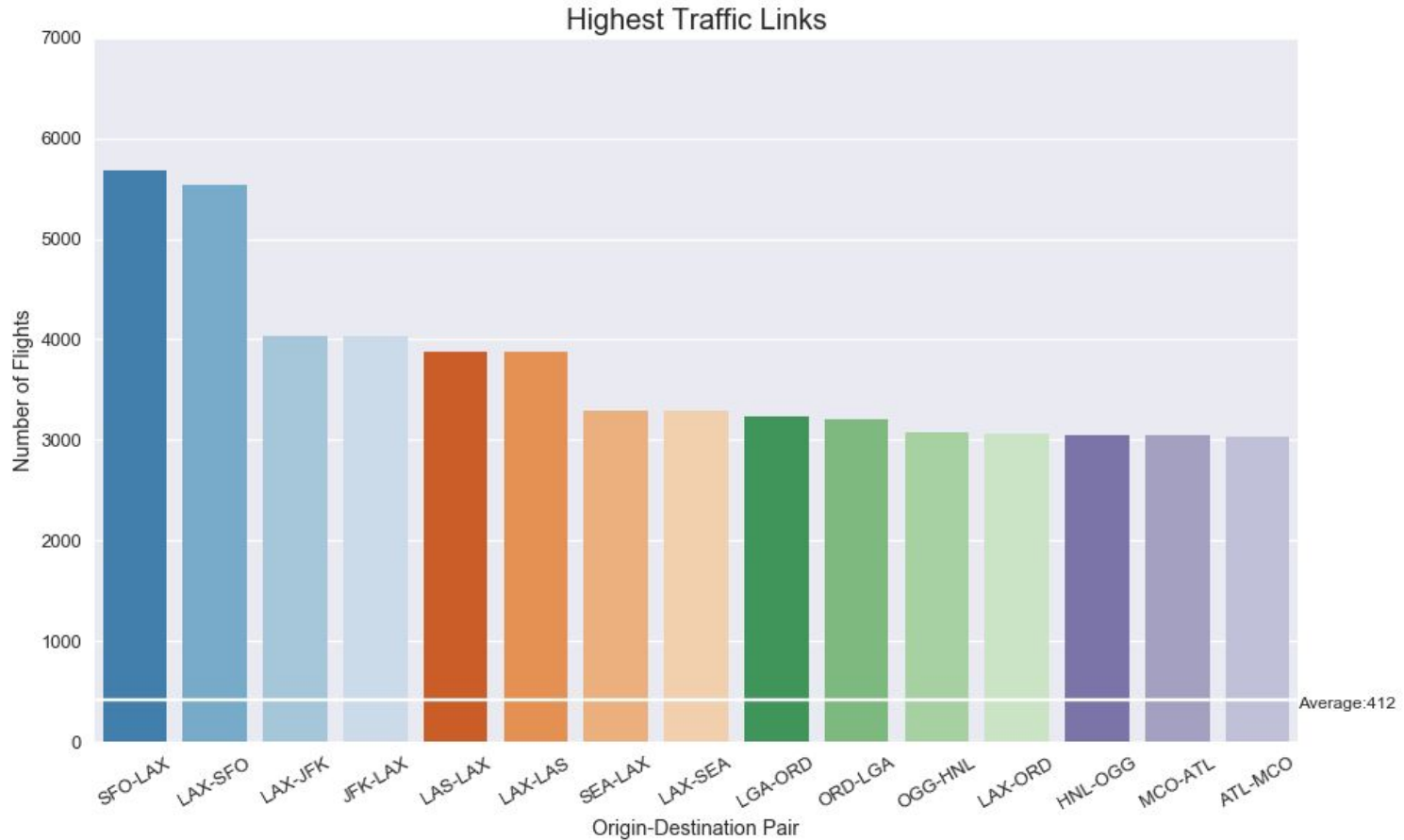
1.3 Frequency by Location

| | Number of Locations Reported |
|--------------------------------|------------------------------|
| Origin Airport | 309 |
| Origin City | 297 |
| Origin State | 52 |
| Destination Airport | 308 |
| Destination City | 297 |
| Destination State | 52 |
| Origin-Destination Pair (Link) | 4431 |

1.3 Highest Traffic Locations

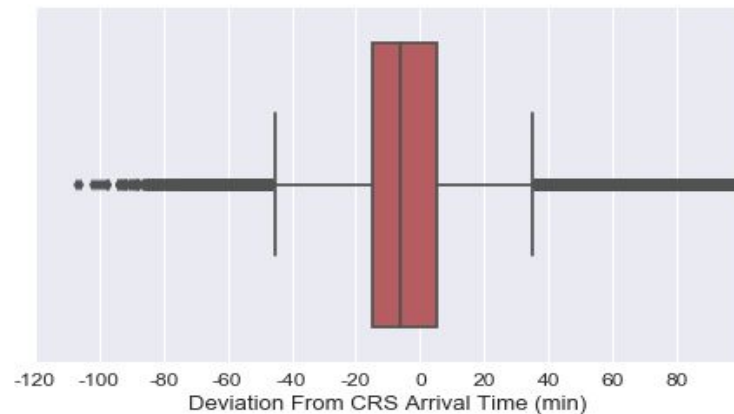
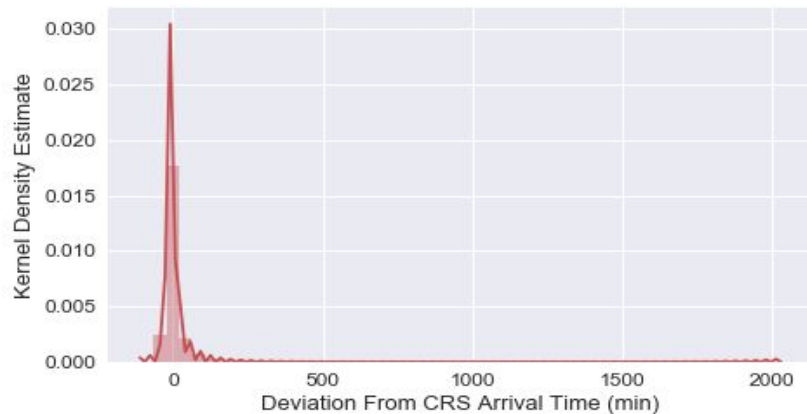
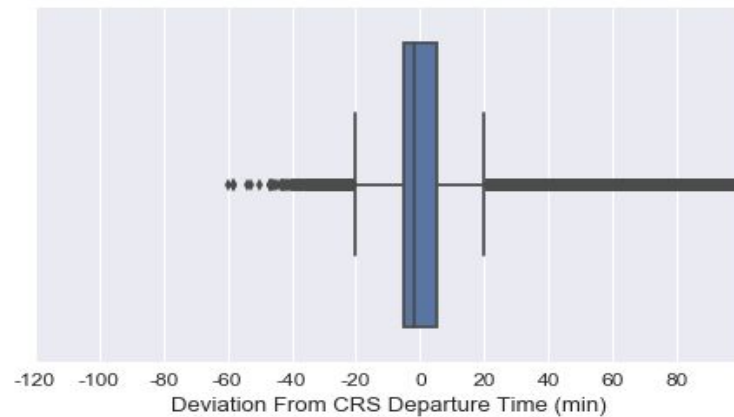
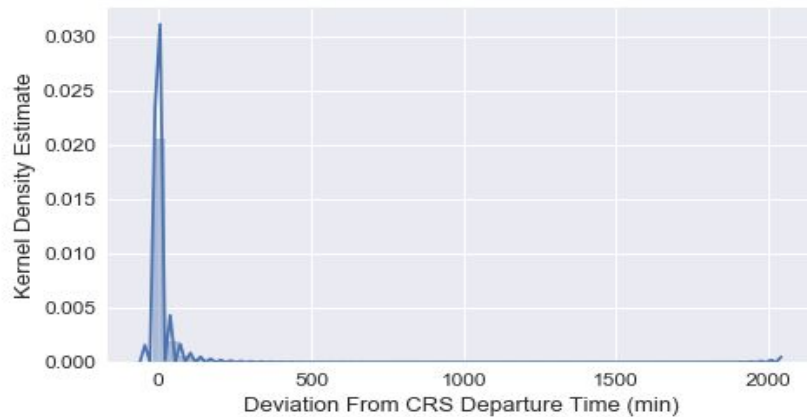


1.3 Highest Traffic Origin-Destination Pairs

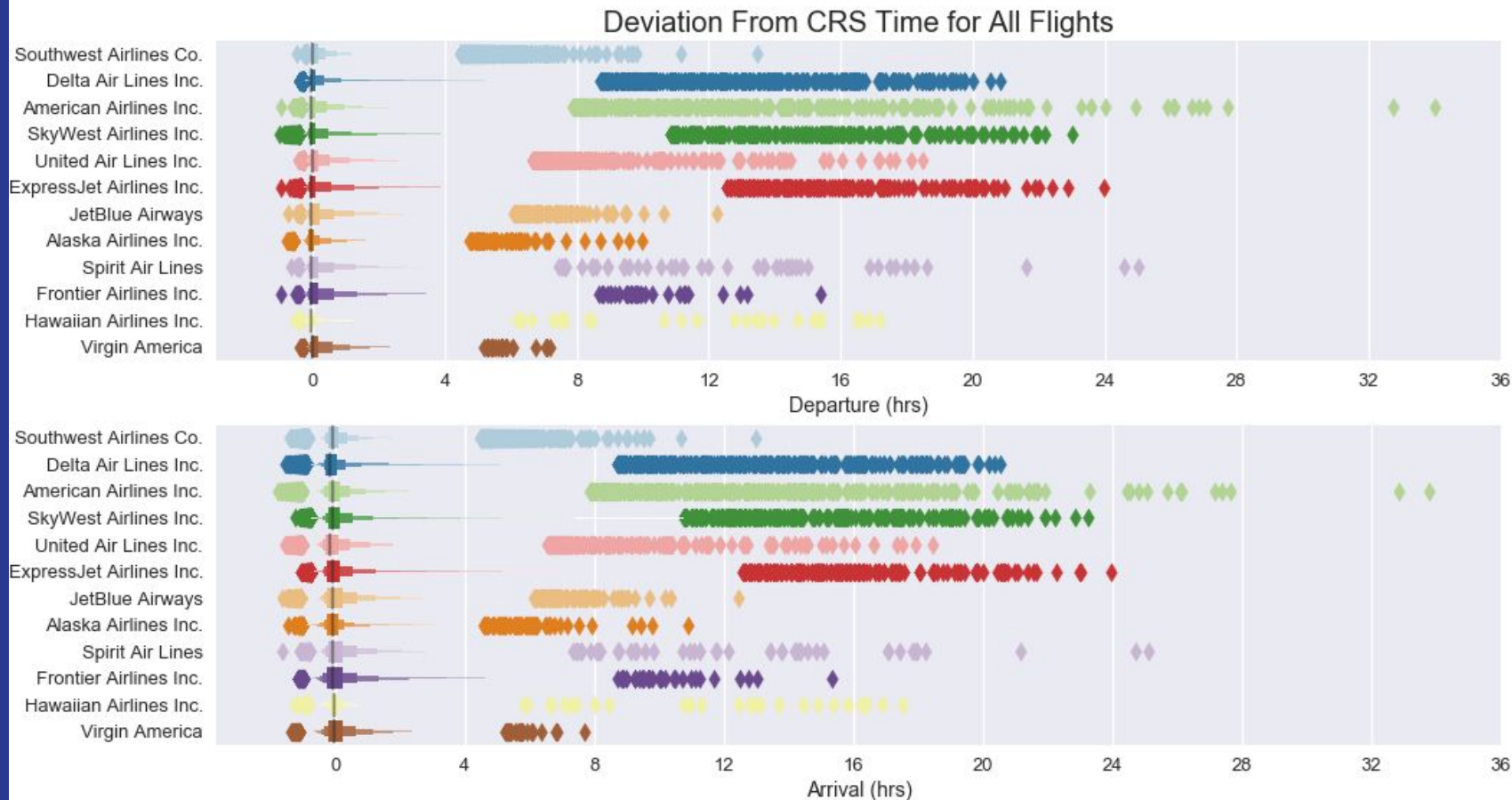


2.1 Overall Delay Distribution

Distribution of Delay Across All Flights

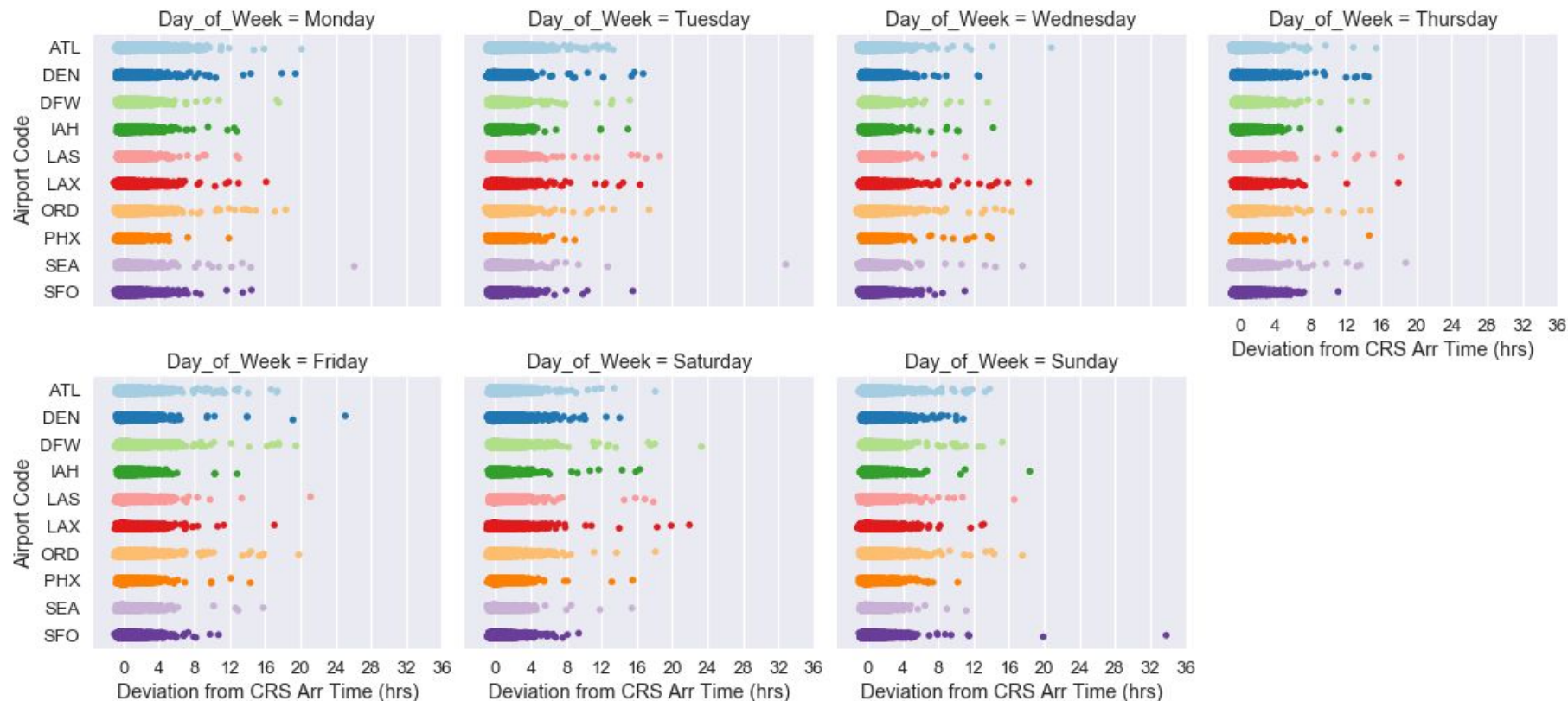


2.2 Distribution of Delay by Carrier



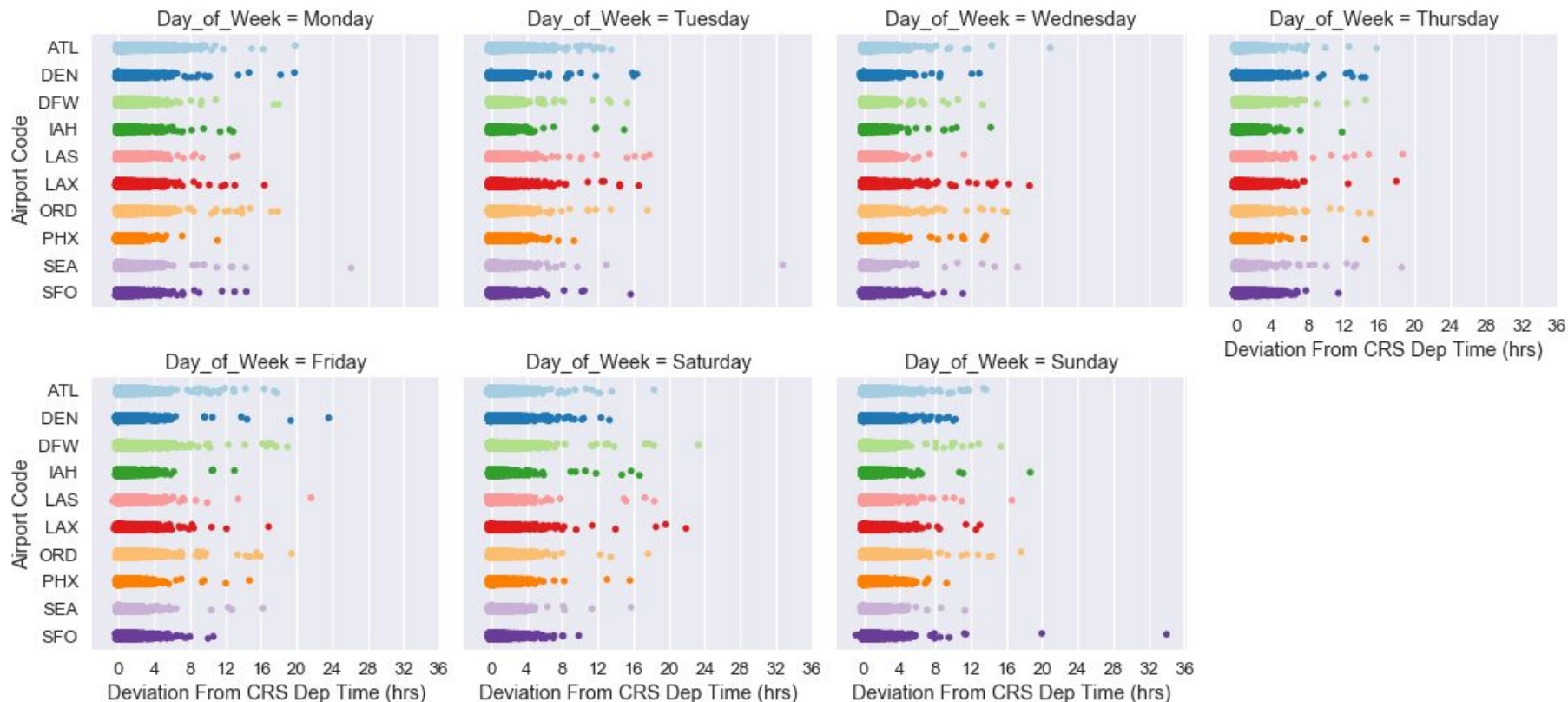
2.3 Distribution of Delay for Highest Traffic Airports

Arrival Distribution For Top 10 Airports

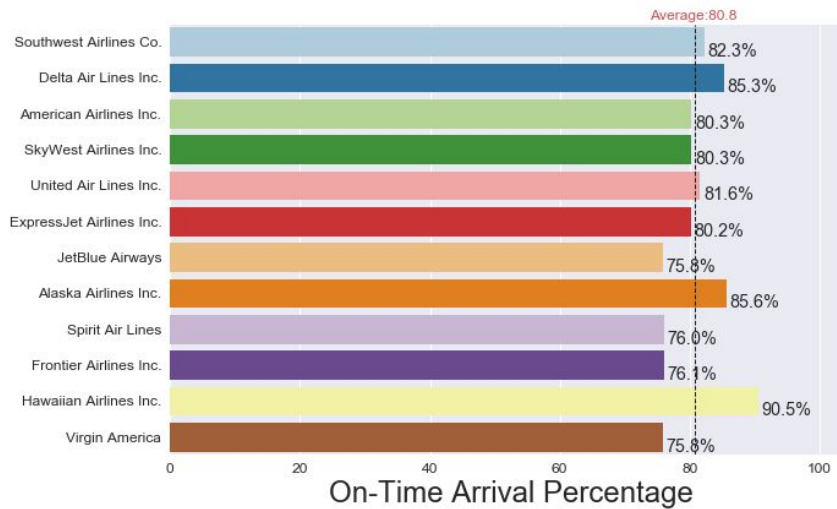
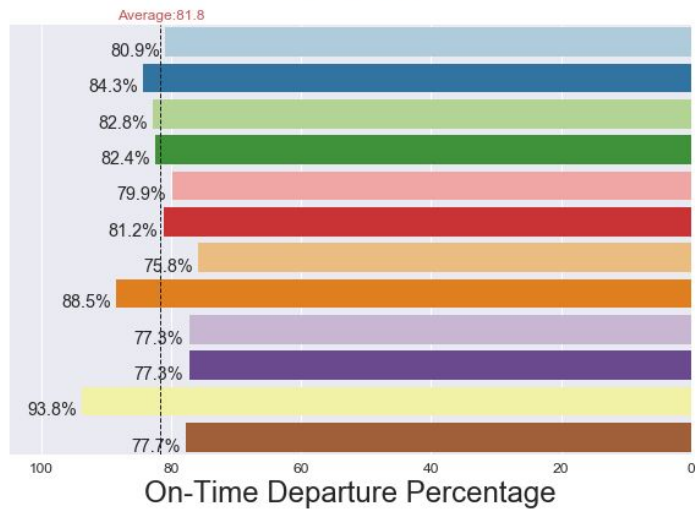
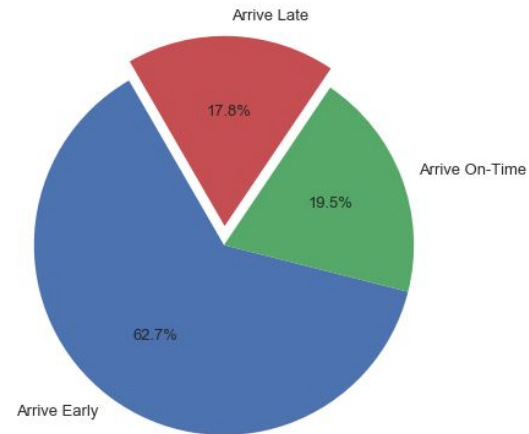
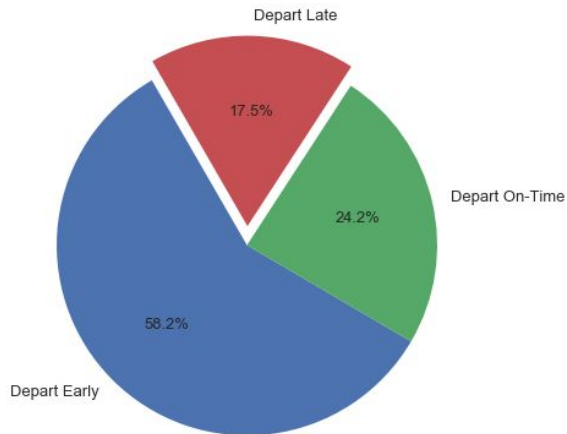


2.3 Distribution of Delay for Highest Traffic Airports

Departure Distribution For Top 10 Airports

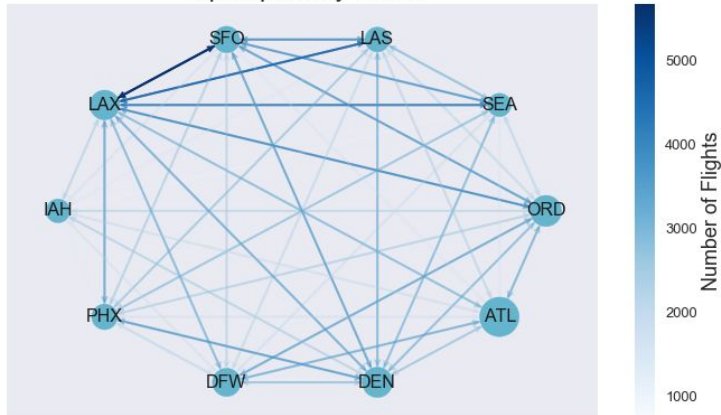


On-Time Performance For All Flights

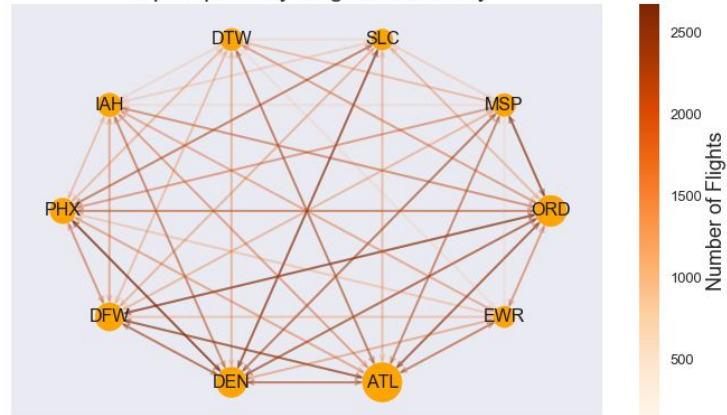


4. Network Visualization

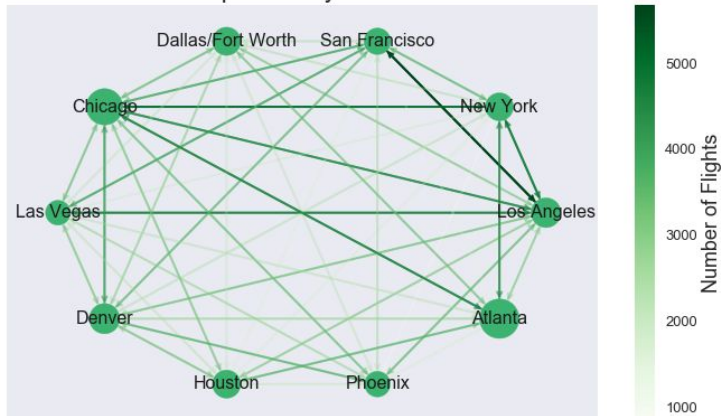
Top Airports by Traffic



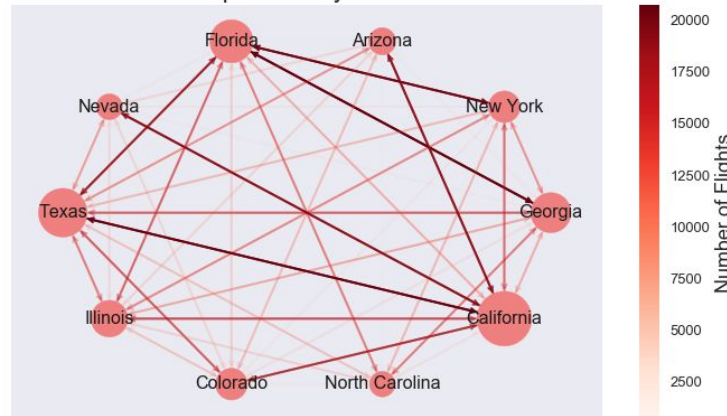
Top Airports by Degree Centrality



Top Cities by Traffic



Top States by Traffic



Inferential Statistics

Initial Data Analysis


1. Test for Normality and CLT in variables of interest
 2. Regression Analysis
 3. Hypothesis Tests
 - T-Test of Arrival and Departure Deviation Means
 - Permutation Test for Pearson's r between Median Arrival and Departure Delays
-

Variables of Interest

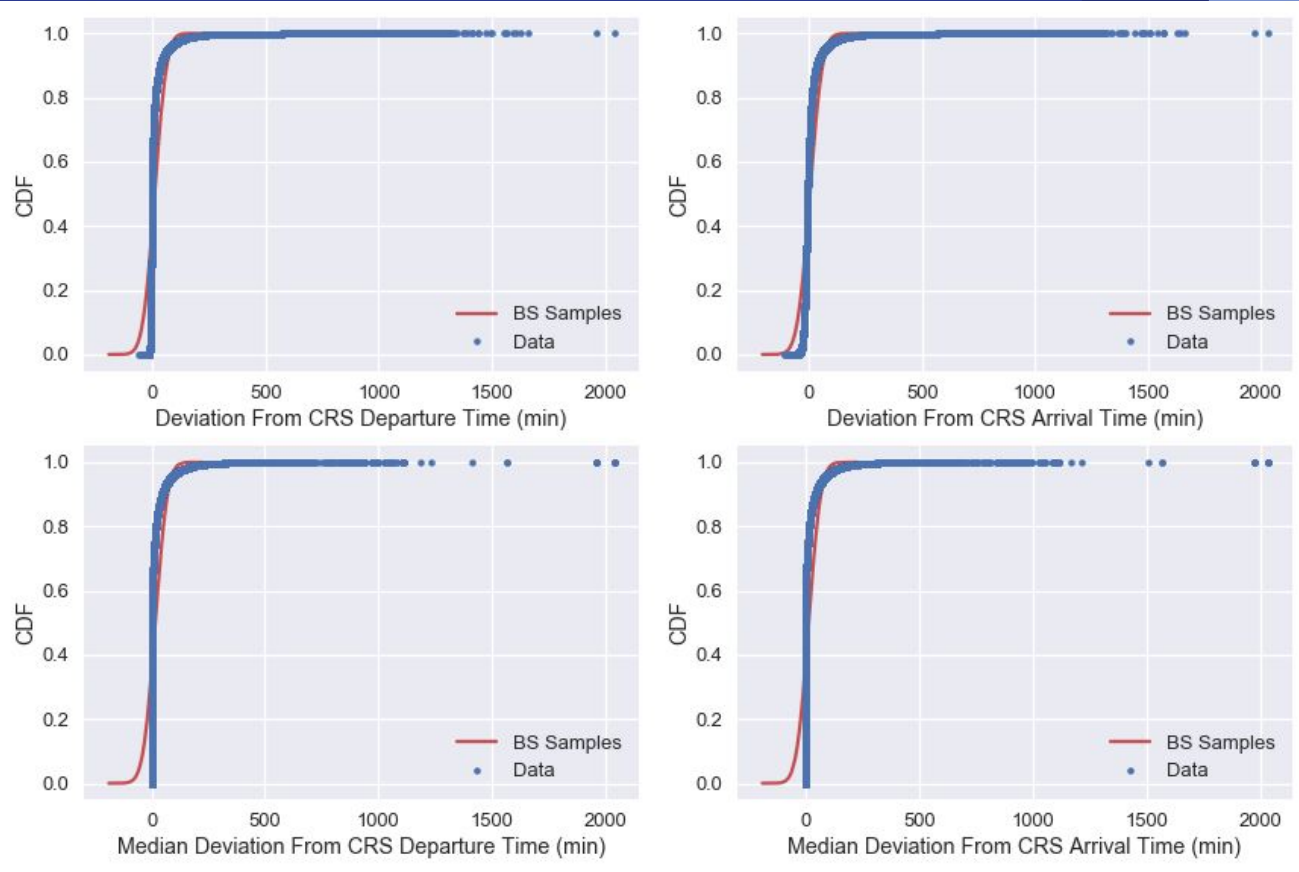
Flights DataFrame:

- dep_deviation: deviation from the actual departure time to the scheduled (CRS) departure time
- arr_deviation: deviation from the actual arrival time to the scheduled (CRS) arrival time

Links_d DataFrame:

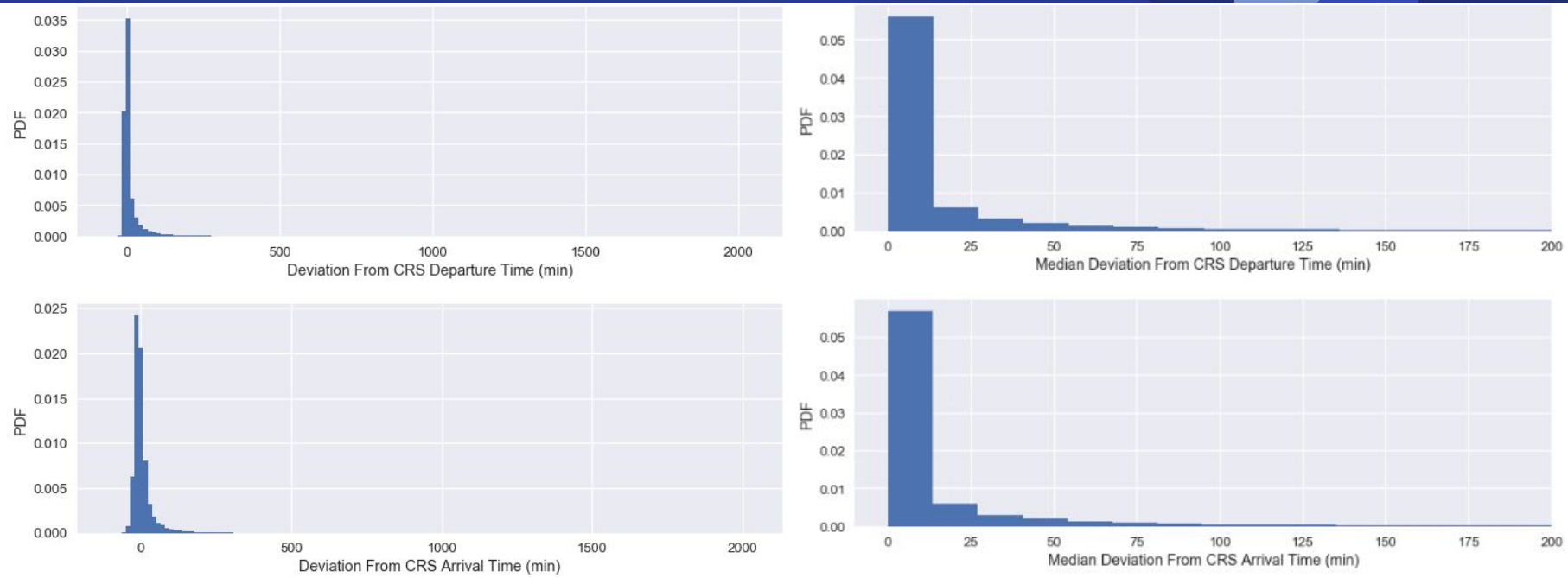
- dep_delay: median deviation of the actual departure time from the scheduled (CRS) departure time, for an origin-destination pair, in an hour of day that had non-zero traffic
 - arr_delay: median deviation of the actual arrival time from the scheduled (CRS) arrival time, for an origin-destination pair, in an hour of day that had non-zero traffic
- 

Test for Normality in Variables of Interest



The Cumulative Density Function of each variable shows they all possess normal distributions

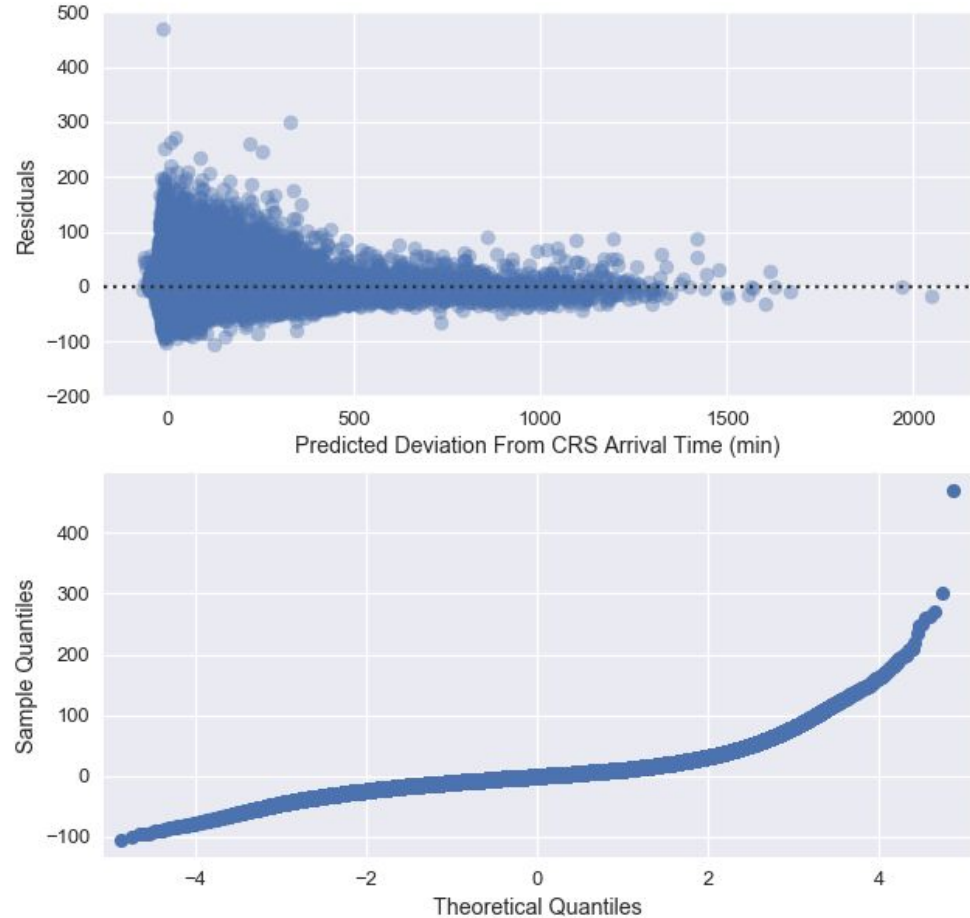
Test for Central Limit Theorem



Central Limit Theorem applies as sample size is very large and the probability density function of each variable shows observations are independent

Regression Analysis

- For the purpose of validating a directed network approach utilizing only departure delay as an indicator of overall (departure and arrival) delay
- The residuals between departure and arrival deviation are in a random pattern, supporting the use of a linear model. The quantile plot shows that the datasets are heavily skewed, and robust methods should be used in model construction to lessen the influence of extreme values.



T-Test of Arrival and Departure Deviation Means

$$H_0 : \mu_d = \mu_a$$

$$H_a : \mu_d \neq \mu_a$$

Margin of Error: 0.045

Difference of Means: 5.549

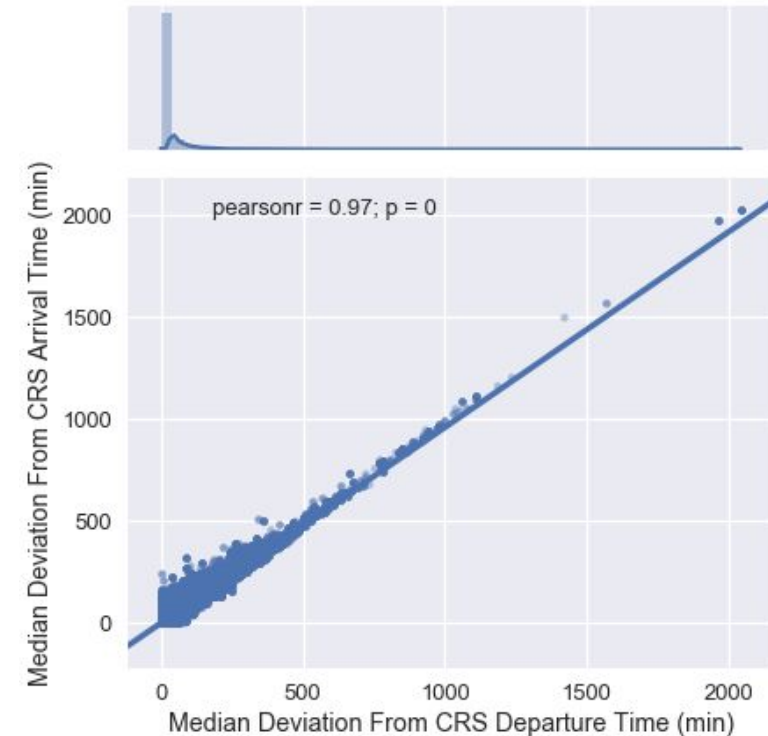
Departure Deviation 95% Confidence Interval: [-11. 112.] min

Arrival Deviation 95% Confidence Interval: [-31. 112.] min

T-Test: tstat = -124.015, P-value = 0.0000000000

With an alpha level of .01 ($\alpha = .01$), the difference between the means of departure and arrival deviation from scheduled (CRS) time was statistically significant, $p < .01$

Permutation Test for Pearson's r between Median Arrival and Departure Delays



H_0 : The correlation between the current median departure delay and median arrival delay for an Origin-Destination pair is not significant

H_a : The correlation between the current median departure delay and median arrival delay for an Origin-Destination pair is significant

With an alpha level of .01 ($\alpha = .01$), the correlation between the median departure delay and the median arrival delay for Origin-Destination pairs is statistically significant, $p < .01$

Model Construction

Binary Departure Delay
Classification

1. Baseline Classifier utilizing Logistic Regression
 2. Resample training data to address target class imbalance, using both under- and over-sampling techniques
 3. Train Logistic Regression and Random Forest Classifiers under both resampling conditions
-

Target Variable Description

Problem Statement:

Predict whether or not the departure delay on an Origin-Destination pair, with a 6-hour prediction horizon, will exceed a 15 min threshold; such that the future delay will fall into one of two classes, 'above threshold' (1) or 'below threshold' (0).

Departure Delay with a 6-hour prediction horizon is defined below:


```
'''write function to create series consisting of delay values at a future time
- loop through each link since time-frames vary per sample i.e. not every link had traffic all 24 hours
- shift series of delay values by -N amount to get values N hours away from the datetime in each row
- forward fill missing values to maintain delay propagation dynamic
'''
def in_nhours(df, col, n_hours, grouping_level):
    future_lst = []
    for i in df.index.get_level_values(grouping_level).unique().sort_values():
        future_lst.append(df.loc[i, col].shift(-n_hours).fillna(method='ffill'))
    future_values_array = pd.concat(future_lst).values
    return future_values_array

#add column of departure delay values 6 hours past the current datetime to links_d dataframe
links_d = links_d.assign(dd_in_6hrs = in_nhours(links_d, 'dep_delay', 6, 'link'))

'''add target column to links_d dataframe
- derived from departure delay in 6hrs in this case
- 1 if delayed (15min past scheduled time per Bureau of Transportation statistics)
- 0 otherwise
'''
links_d['dd_binary'] = (links_d['dd_in_6hrs'] >= 15)*1
```

Baseline - Logistic Regression Classifier

Procedure:

1. Standardize data with RobustScaler from the sklearn.preprocessing package, using the median and IQR to center and scale data in order to account for outliers, which occur significantly in the variables of interest
 2. Split the data 70/30 into training and test sets, stratifying to assure equal incidence of target classes in both sets
 3. Fit Logistic Regression estimator with training set, and get predicted values for both training and test sets, as well as probabilities for test set predictions
 4. Perform 5-fold Stratified Cross-Validation to calculate average prediction accuracy, and get Precision, Recall, and F1-scores for both sets using Classification Report from sklearn.metrics package
 5. Plot applicable performance metrics using the metrics module from scikit-plot
- 

Initial Results

Classification Metrics for Baseline Logistic Regression Model

CV-Accuracy on training data: 0.7721 (+/- 0.0000)

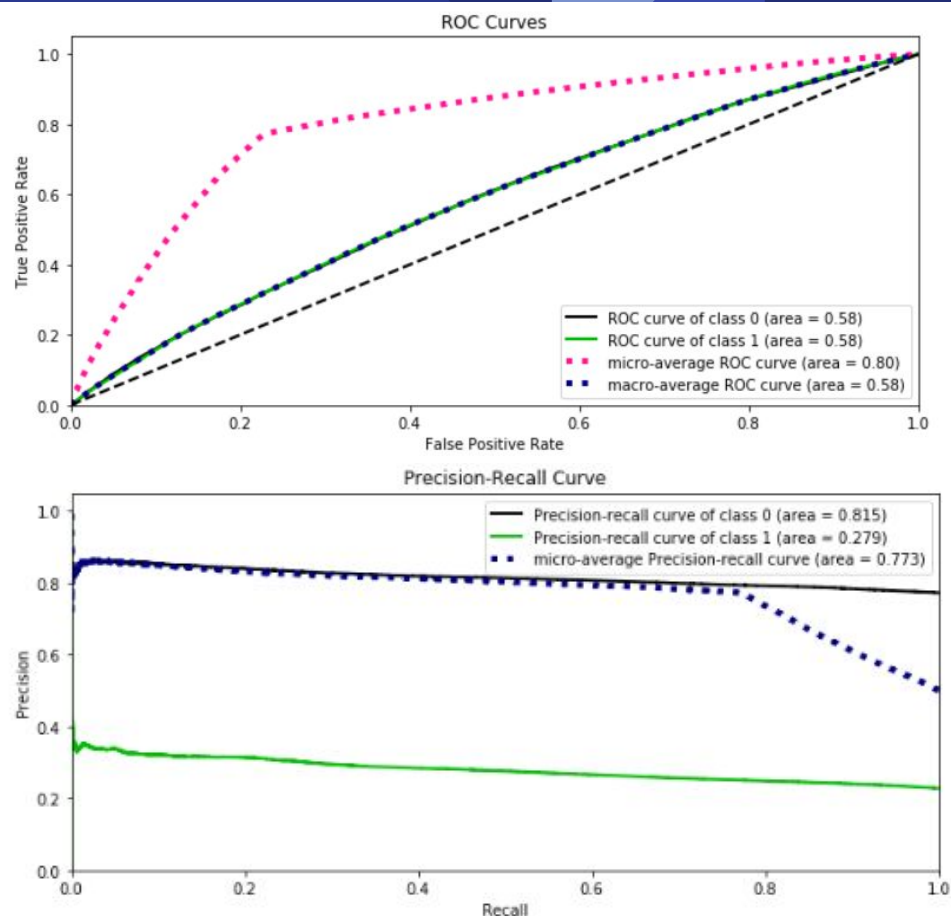
CV-Accuracy on test data: 0.7721 (+/- 0.0000)

[Training Classification Report]

| | precision | recall | f1-score | support |
|-------------|-----------|--------|----------|---------|
| 0 | 0.77 | 1.00 | 0.87 | 477153 |
| 1 | 0.00 | 0.00 | 0.00 | 140817 |
| avg / total | 0.60 | 0.77 | 0.67 | 617970 |

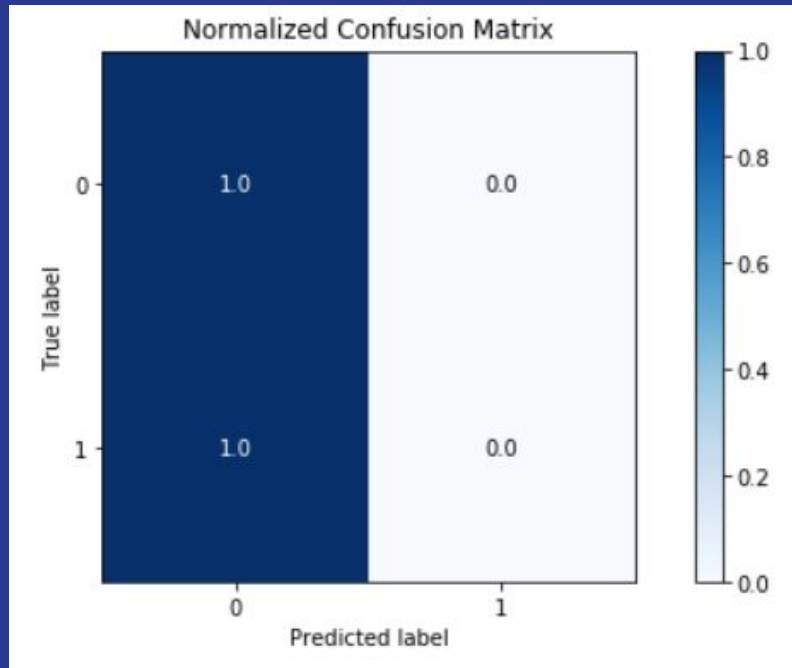
[Test Classification Report]

| | precision | recall | f1-score | support |
|-------------|-----------|--------|----------|---------|
| 0 | 0.77 | 1.00 | 0.87 | 204495 |
| 1 | 0.00 | 0.00 | 0.00 | 60350 |
| avg / total | 0.60 | 0.77 | 0.67 | 264845 |



Initial Results Cont'd

- Due to the imbalanced distribution of the delayed class , 22.7% of dataframe being used for model construction, the baseline classifier performed very poorly in predicting true positives, opting to classify all predictions as not delayed in order to maximize accuracy,.
- Mean accuracy (0.7721 for test set CV)), was deemed to be an inappropriate evaluation metric in this case, with the individual Precision, Recall, and F1-scores for each class being more indicative of model performance.



Moving Forward

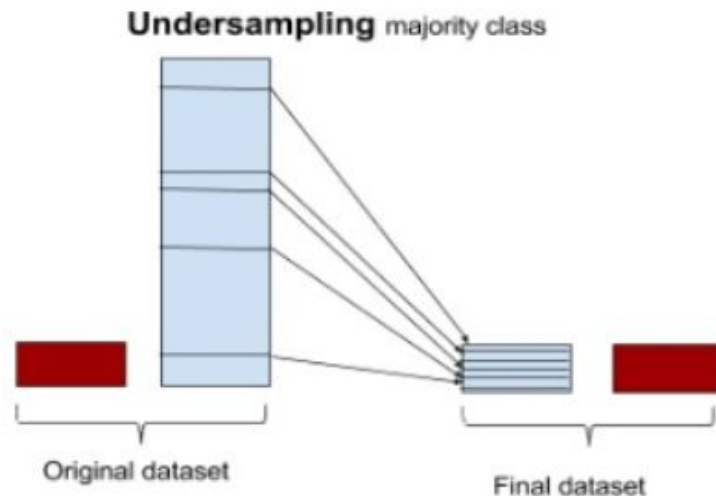
Resample to Counter Class Imbalance

1. Utilize Under- and Over- sampling techniques on the training dataset
2. Train each method on RandomForest and Logistic Regression classifiers
3. Test Classifiers under both conditions and perform cross-validation with the test set containing the natural distribution of classes
4. Assess model performance with Precision, Recall, F-1, and AUC scores. Particularly the Area under the ROC curve statistic, as it is equal to the probability that a random positive example will be ranked above a random negative example, and the Precision-Recall curve as a visual metric.



Condition 1: Random Under Sampling

- Using RandomUnderSampler from imblearn.under_sampling package
- Under-samples the majority class in the training dataset by randomly picking samples without replacement



Original target shape: Counter({0: 477153, 1: 140817})

Resampled target shape: Counter({0: 140817, 1: 140817})

Condition 1 Results: Logistic Regression Classifier

Classification Metrics on Undersampled Logistic Regression Model

Accuracy on training data: 0.5540 (+/- 0.0050)

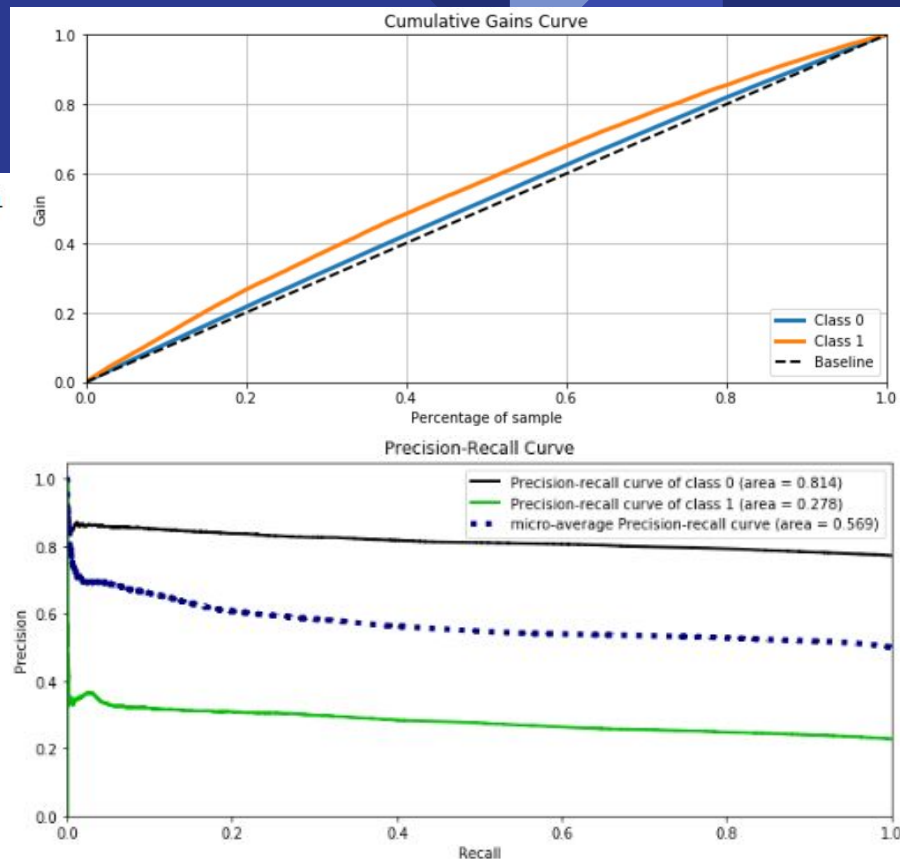
Accuracy on test data: 0.7721 (+/- 0.0000)

[Training Classification Report]

| | precision | recall | f1-score | support |
|-------------|-----------|--------|----------|---------|
| 0 | 0.56 | 0.54 | 0.55 | 140817 |
| 1 | 0.56 | 0.58 | 0.57 | 140817 |
| avg / total | 0.56 | 0.56 | 0.56 | 281634 |

[Test Classification Report]

| | precision | recall | f1-score | support |
|-------------|-----------|--------|----------|---------|
| 0 | 0.81 | 0.53 | 0.64 | 204495 |
| 1 | 0.27 | 0.57 | 0.36 | 60350 |
| avg / total | 0.69 | 0.54 | 0.58 | 264845 |



Condition 1 Results: Random Forest Classifier

Tuned Model Parameters: {'max_features': 'log2', 'n_estimators': 100}
Best GSCV score on Training Set: 0.7398

Classification Metrics for Undersampled Random Forest Classifier

CV-Accuracy on training data: 0.7529 (+/- 0.0028)

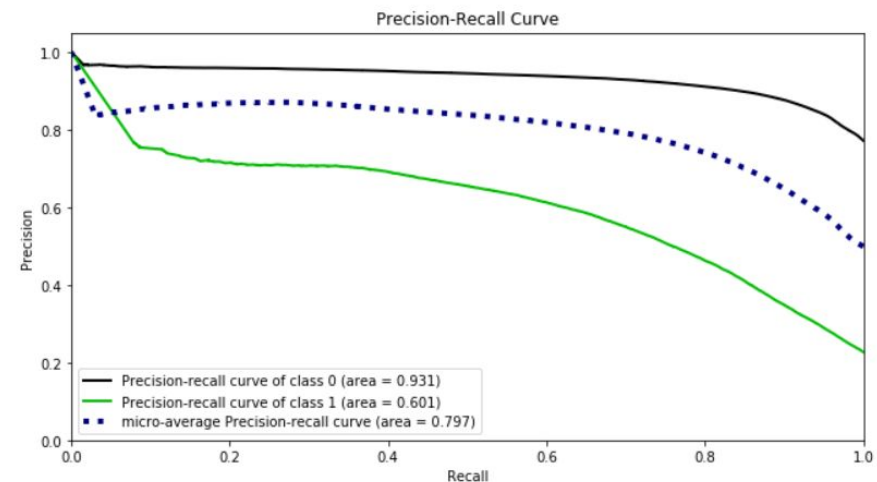
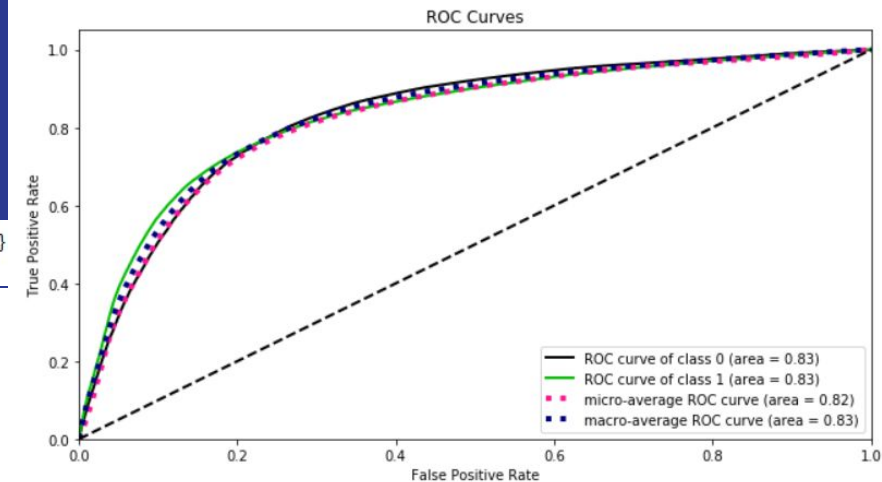
CV-Accuracy on test data: 0.8093 (+/- 0.0025)

[Training Classification Report]

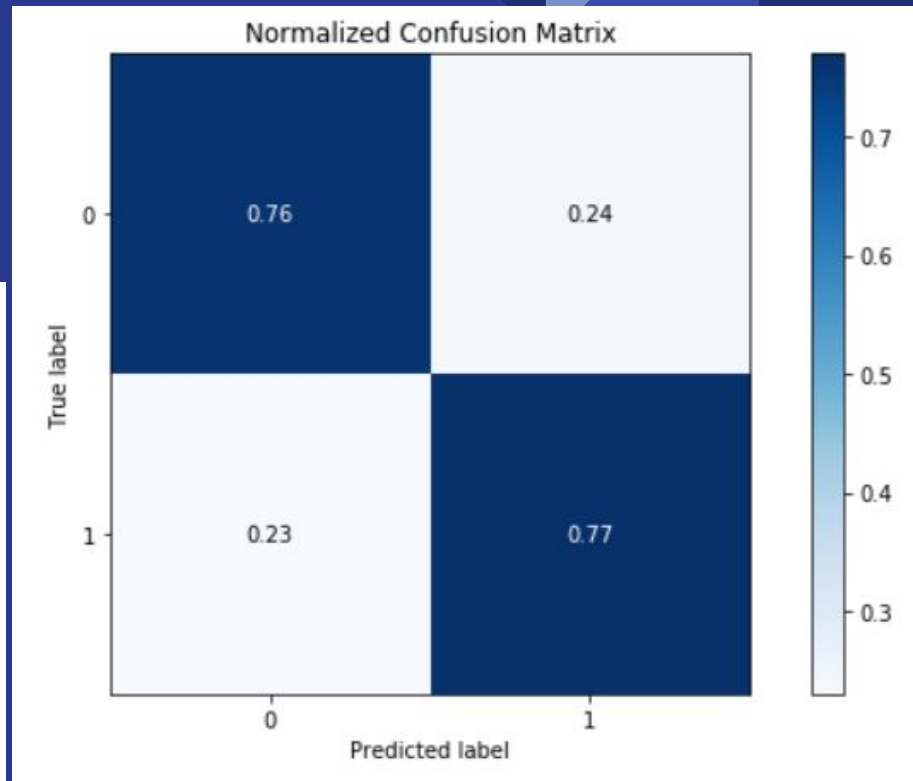
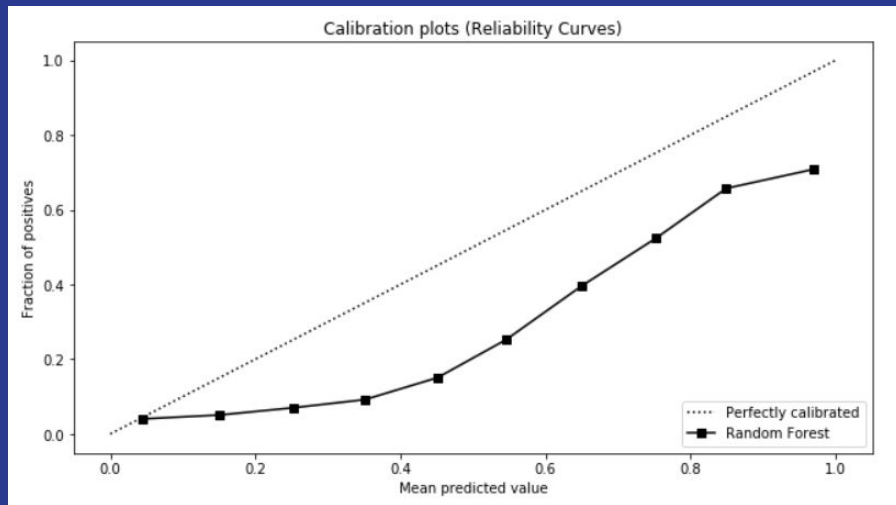
| | precision | recall | f1-score | support |
|-------------|-----------|--------|----------|---------|
| 0 | 0.94 | 0.93 | 0.94 | 140817 |
| 1 | 0.93 | 0.94 | 0.94 | 140817 |
| avg / total | 0.94 | 0.94 | 0.94 | 281634 |

[Test Classification Report]

| | precision | recall | f1-score | support |
|-------------|-----------|--------|----------|---------|
| 0 | 0.92 | 0.76 | 0.83 | 204495 |
| 1 | 0.49 | 0.77 | 0.60 | 60350 |
| avg / total | 0.82 | 0.76 | 0.78 | 264845 |

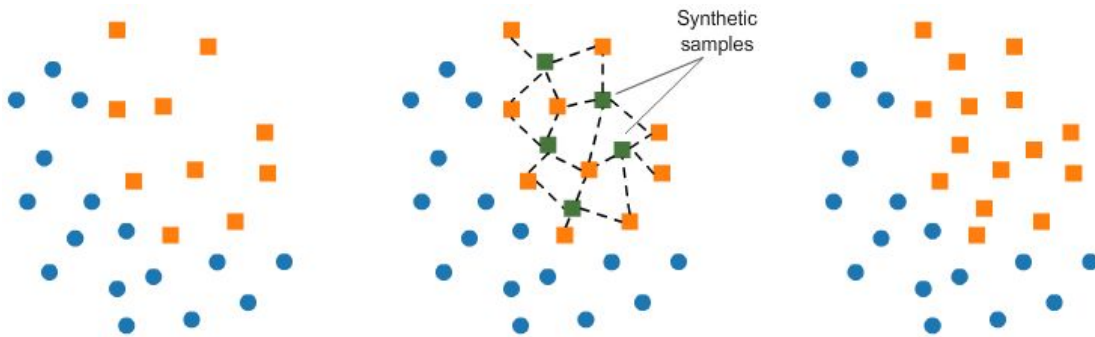


Condition 1 Results: Random Forest Classifier



Condition 2: Synthetic Minority Over-sampling Technique

- Using SMOTE from `imblearn.over_sampling` package
- Over-samples the minority class in the training dataset by creating synthetic samples, as opposed to creating copies.
- The algorithm selects k-number of similar instances (using a distance measure) and perturbs a new instance one attribute at a time by a random amount within the difference to its neighbors.



```
Original target shape: Counter({0: 477153, 1: 140817})  
Resampled target shape: Counter({0: 477153, 1: 477153})
```

Condition 2 Results: Logistic Regression Classifier

Classification Metrics on Oversampled Logistic Regression Model

Accuracy on training data: 0.5570 (+/- 0.0012)

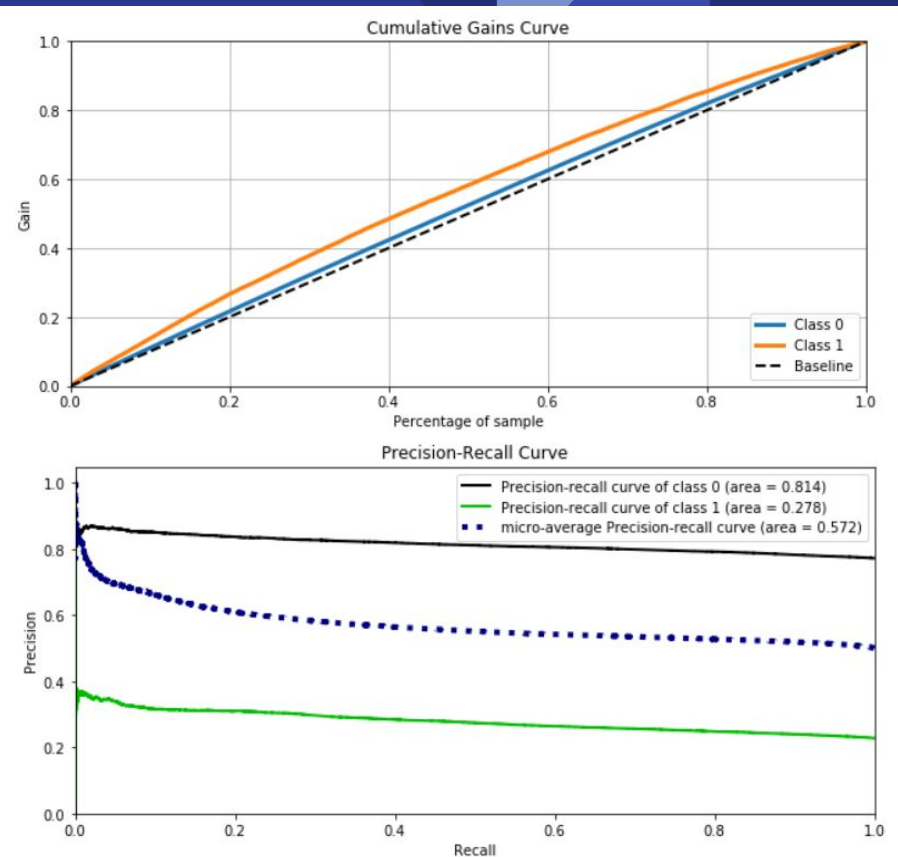
Accuracy on test data: 0.7721 (+/- 0.0000)

[Training Classification Report]

| | precision | recall | f1-score | support |
|-------------|-----------|--------|----------|---------|
| 0 | 0.56 | 0.54 | 0.55 | 477153 |
| 1 | 0.56 | 0.58 | 0.57 | 477153 |
| avg / total | 0.56 | 0.56 | 0.56 | 954306 |

[Test Classification Report]

| | precision | recall | f1-score | support |
|-------------|-----------|--------|----------|---------|
| 0 | 0.81 | 0.54 | 0.65 | 204495 |
| 1 | 0.27 | 0.57 | 0.36 | 60350 |
| avg / total | 0.69 | 0.55 | 0.58 | 264845 |



Condition 2 Results: Random Forest Classifier

Tuned Model Parameters: {'max_features': 'log2', 'n_estimators': 100}

Best GSCV score on Training Set: 0.8963

Classification Metrics for Oversampled Random Forest Classifier

CV-Accuracy on training data: 0.9032 (+/- 0.1080)

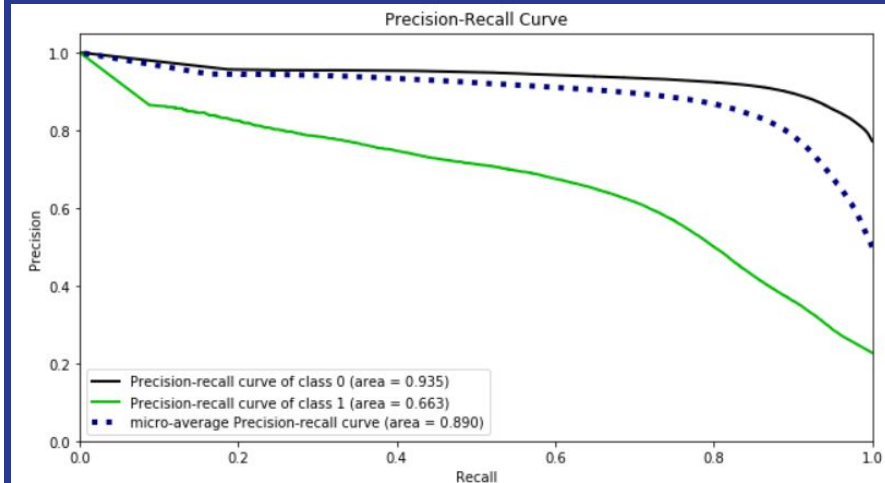
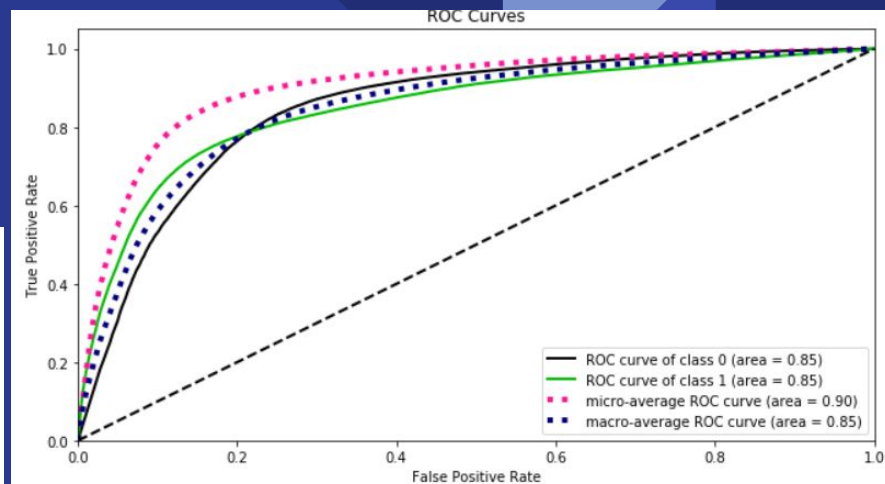
CV-Accuracy on test data: 0.8092 (+/- 0.0019)

[Training Classification Report]

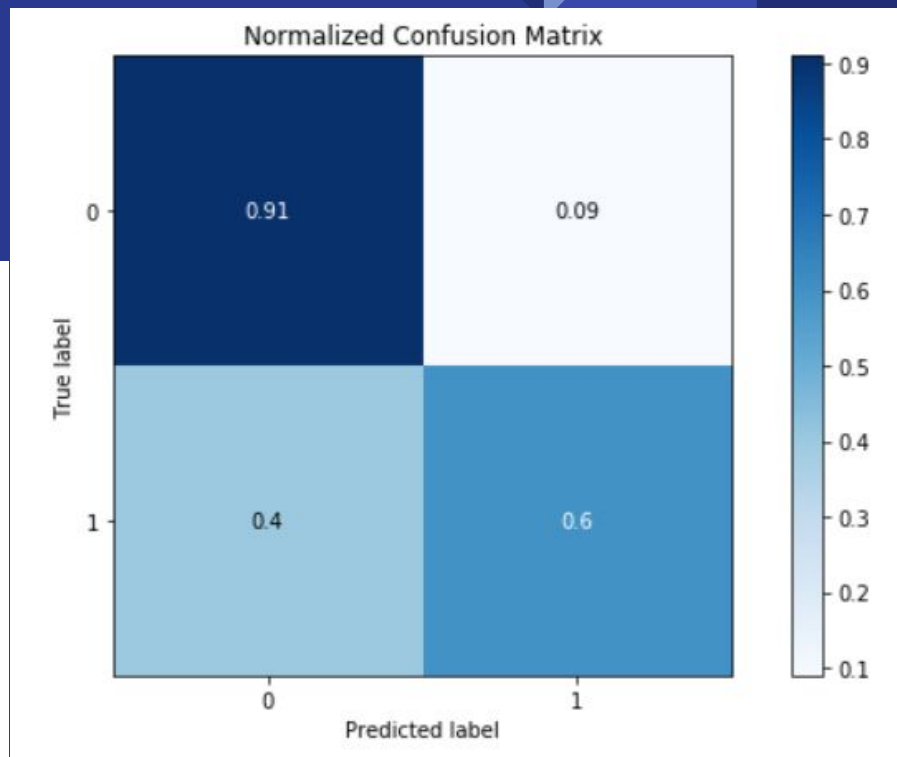
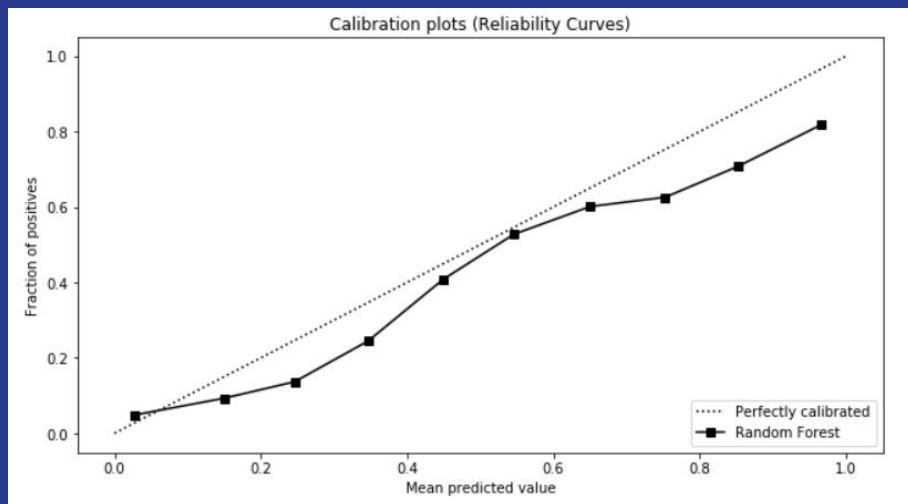
| | precision | recall | f1-score | support |
|-------------|-----------|--------|----------|---------|
| 0 | 0.96 | 0.95 | 0.95 | 477153 |
| 1 | 0.95 | 0.96 | 0.95 | 477153 |
| avg / total | 0.95 | 0.95 | 0.95 | 954306 |

[Test Classification Report]

| | precision | recall | f1-score | support |
|-------------|-----------|--------|----------|---------|
| 0 | 0.89 | 0.91 | 0.90 | 204495 |
| 1 | 0.67 | 0.60 | 0.64 | 60350 |
| avg / total | 0.84 | 0.84 | 0.84 | 264845 |



Condition 2 Results: Random Forest Classifier



Conclusion

- Overall, of the four estimators trained, the logistic regression models performed the worst, proving unreliable for this particular problem, although they might potentially be improved if weights were added to the classes. Therefore, the ensemble approach appears to more appropriate in this case.
- Utilizing the AUC score as the primary comparative statistic for model performance, the Random Forest Classifier trained on data oversampled using SMOTE (Synthetic Minority Oversampling Technique) performed best, with an AUC of 0.85, a Precision score for the delay class of 0.67, and an F1-score of 0.64, and would be my recommendation to a potential client.