Predicting Flight Departure Delays in an Air-Traffic Network

Capstone Project 1
Miguel Montano

I. Background and Problem Statement

The inherently complex nature of air transportation, with constrained airspace and airport resources, thousands of aircraft and overburdened air-traffic controllers, and unforeseen weather disruptions results in the incidence of delays being an inevitability. Nearly 40% of delays were due to the delayed arrival of the incoming aircraft, reflecting the high levels of interdependence in the delay dynamics (Gopalkrishnan et al. 2017). Delays result in the need for additional gates and ground personnel, and impose costs on airline customers (including shippers in the form of lost productivity, and additional wages). Comprehensive studies showed $40 billion in annual costs in 2015, and in 2016, the average cost of aircraft block (taxi plus airborne) time for U.S. passenger airlines was estimated to be $62.55 per minute ("Per-Minute Cost of Delays to US Airlines", airlines.org). A tremendous amount of capital is spent annually on mitigating the effects of delay, but with an ability to anticipate the likelihood of a delay, further steps can be taken to remedy auxiliary costs. I propose an approach to mitigating this problem, by constructing a model to 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).

II. Potential Clients

There are three types of clients that may benefit from such an approach, flight data distributors, air traffic controllers, and airport ground crews. Flight data distributors consist of companies distributing use of the Aircraft Situation Display to Industry (ASDI) from the US Dept. of Transportation's Volpe Transportation Center, which provides real-time position and flight plans in U.S. airspace. Air traffic controllers range from the Air Traffic Control System Command Center (ATCSCC) which receives Airport Arrival Rate (AAR) data from air traffic facilities, institutes Ground Delay Programs, and runs compressions to alleviate congested networks; to Air Route Traffic Control Centers (ARTCC) looking to make better predictions of AAR's and plan efficient holding patterns; and Traffic Management Personnel seeking to file flight plans as early as possible to get the earliest Expect Departure Clearance Time (EDCT) in order to avoid getting pushed into the next time block. Finally, ground crews are the implementers of Ground Delay Programs at domestic airports, and being on the proverbial frontlines of flight delays, would greatly benefit from a head start, alleviating the costs that result from inefficient tarmac operations during congested network states.

III. Dataset

Data has been acquired from an online posting on Kaggle containing US domestic flight delays in the month of January, August, November and December of 2016. Data is free and has no contingencies upon its usage, and contains extensive descriptive data on all domestic flights within the time range, including their arrival/departure locations, time intervals spent on various stages of the arrival/departure process, and arrival/departure deviations from their scheduled times, which provides the incidence of delay.

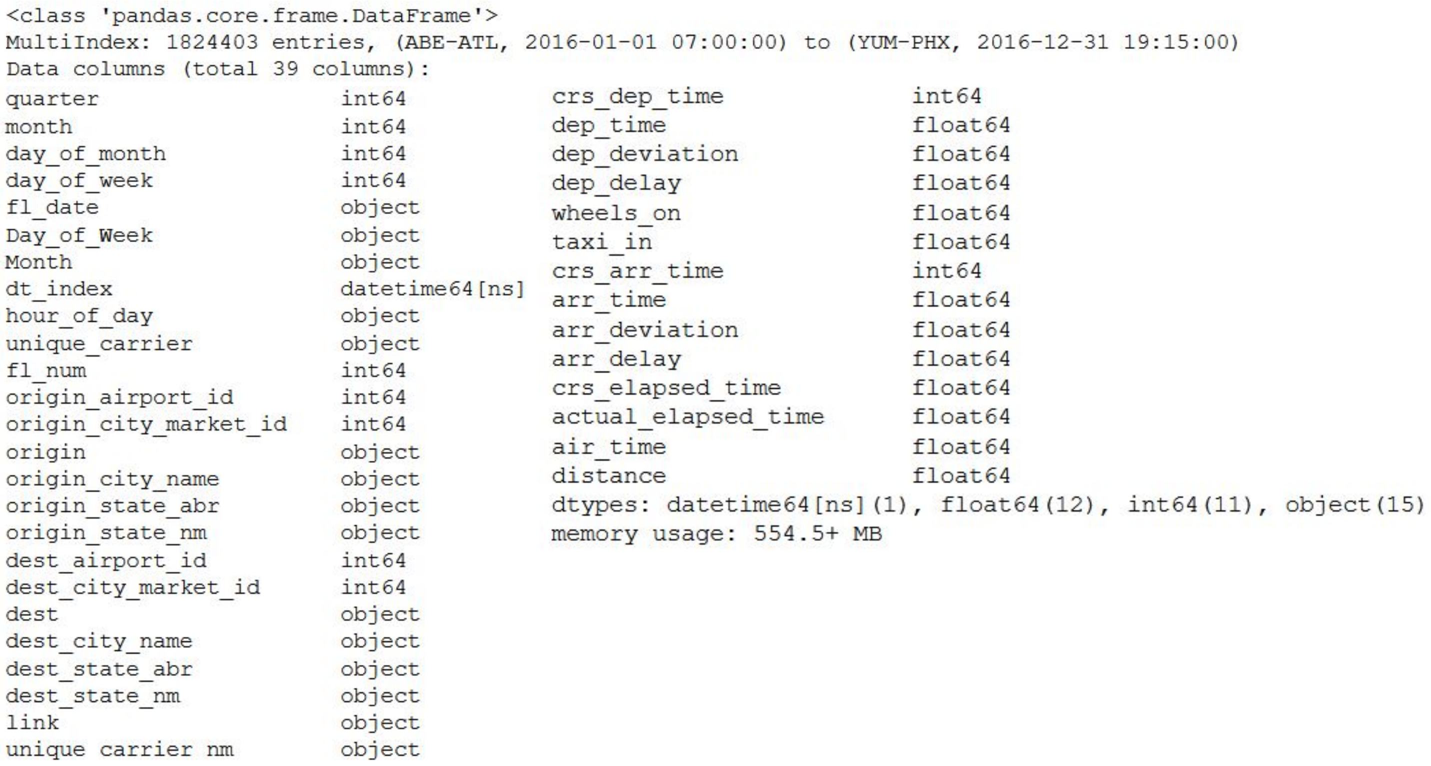
IV. Data Wrangling

1. Initialization and pre-processing

Original flight data was in a CSV format and imported into a pandas DataFrame, consisting of 1,856,051 entries, and 43 columns. With initial surface cleaning of the data consisting of steps to improve readability, such as making text lowercase, and dropping columns that consisted of over fifty percent missing values. After viewing descriptive statistics for the remaining columns with missing values it was determined that they were very pertinent to the problem at hand (they dealt with flight delay times directly), and since they possessed high variability, imputation of means would not be an appropriate approach. Thus the rows with missing values were dropped, as they comprised of only 1.7% of total rows.

2. Compartmentalize, Evaluate, and Join Flights

The remaining columns were separated into three dataframes based on the component of flight information they pertained to: flight date (date_info), flight identification (id_info), and metrics (flight_metrics). Air traffic statistic definitions were gathered via the Bureau of Transportation statistics website, such as a major carrier or major airline carrier being defined as a U.S.-based airline that posts more than $1 billion in revenue during the fiscal year; and 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. For flight date, numerical identifiers were used to create columns of names, and a datetime index was added to assist in the creation of an hour of day column, as well as to assist in the creation of final dataframe later. For flight identification, redundant numerical identifiers such as sequence ids were dropped, unique carrier names were added, and an origin-destination column named links was added to facilitate the creation of a mock air traffic network. For flight metrics, a column pertaining to the number of flight per row was dropped (since all rows correlated to one flight), and deviation and delay columns for arrival and delay were respectively renamed to avoid confusion in later analysis. The three content-specific dataframes were joined to create the final Flights dataframe, consisting of 1,824,403 entries and 39 features; 9 of which are temporal, 16 that are categorical (6 origin, 6 destination, link, and unique carrier name), and 14 that are metric (timestamps, distance, actual vs elapsed CRS times).



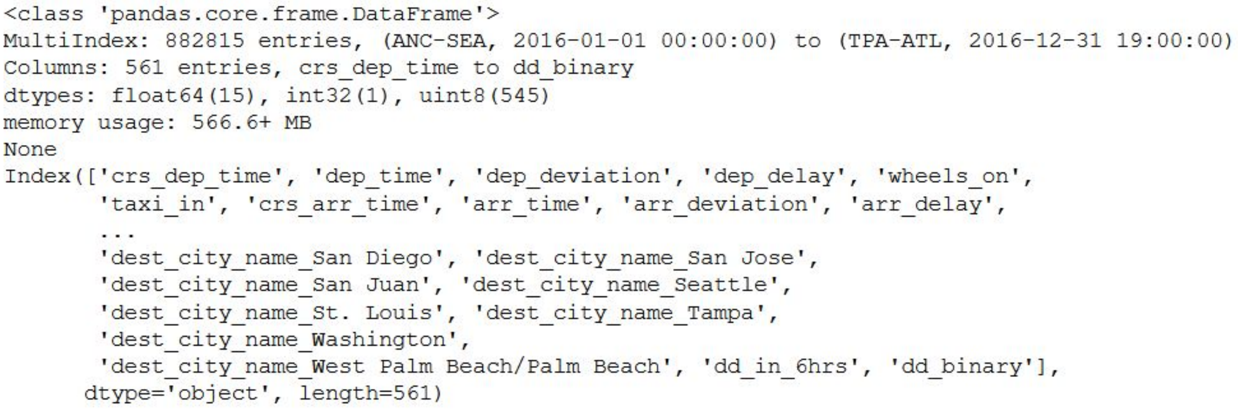
```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 1824403 entries, (ABE-ATL, 2016-01-01 07:00:00) to (YUM-PHX, 2016-12-31 19:15:00)
Data columns (total 39 columns):
quarter                  int64           crs_dep_time              int64
month                    int64           dep_time                  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          dtypes: datetime64[ns](1), float64(12), int64(11), object(15)
origin_state_abr         object          memory usage: 554.5+ MB
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
```

3. Create NetworkX DiGraph from components

Using the networkx.from_pandas_edgelist package, a directed graph was made from the Flights dataframe, with the edges consisting of the distance between nodes, the CRS elapsed time, and a weight calculated from the number of total flights between nodes. Node attributes were added to the NetworkX DiGraph object from two dictionaries, one consisting of degree, degree centrality, and betweenness centrality values per node; and the other consisting of the node's airport code, and respective city and state names.

4. Create final mock air traffic network dataframe, links_d

A mock air traffic network was created by grouping all flights into origin-destination pairs (interchangeably also called links), and resampling the top 300 links by hour, then calculating an aggregate median for each flight metric per link per hour during the sample period. The top 300 links were chosen because they had a high frequency of daily traffic, and a median versus a mean calculation was utilized in order to get a better picture of the spread in the presence of prominent outliers that might otherwise skew observations. Finally, categorical features were dummified, and a binary target class was created by attributing a value of 1 if the link was in a state of delay 6 hours from the current time ('above threshold'), or 0 ('below threshold') otherwise.

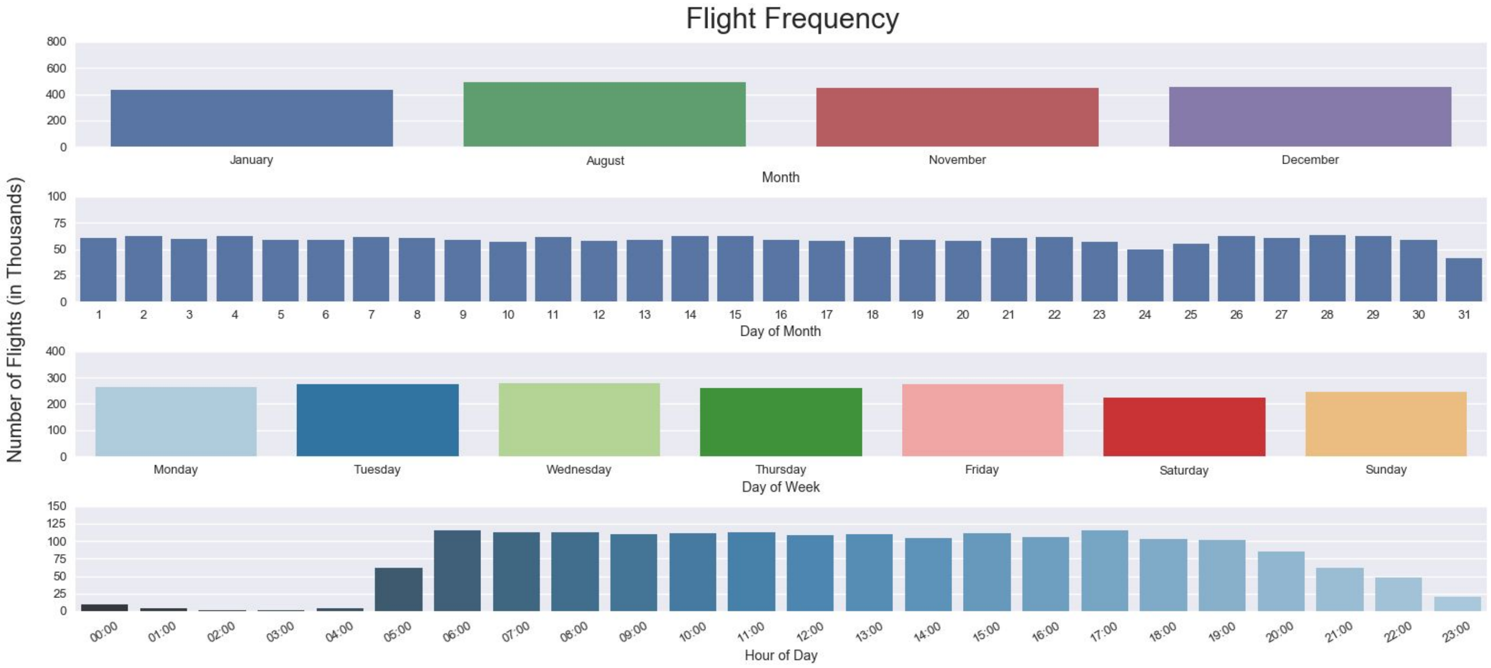


```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 882815 entries, (ANC-SEA, 2016-01-01 00:00:00) to (TPA-ATL, 2016-12-31 19:00:00)
Columns: 561 entries, crs_dep_time to dd_binary
dtypes: float64(15), int32(1), uint8(545)
memory usage: 566.6+ MB
None
Index(['crs_dep_time', 'dep_time', 'dep_deviation', '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)
```

V. Storytelling

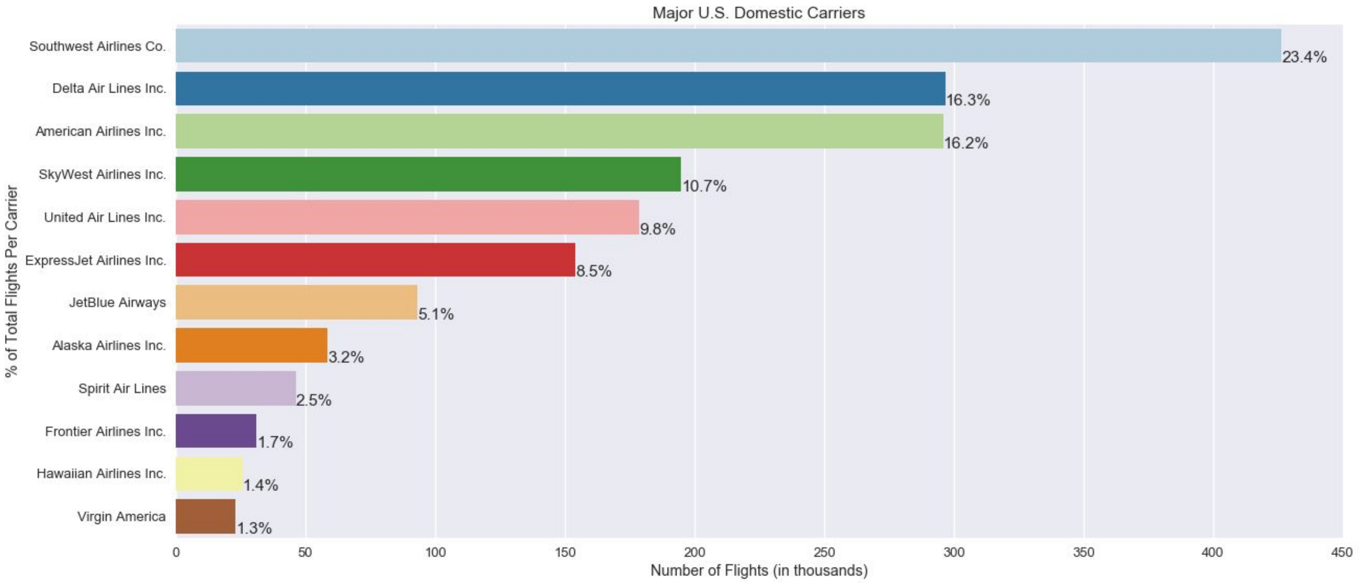
In visualizing the data plots were created fitting into four groups, those displaying flight frequency, those displaying the distribution of delay across different categories, on-time performance for all flights, and a visualization of NetworkX Directed Graph object by top node attributes.

1.1 Frequency by Time



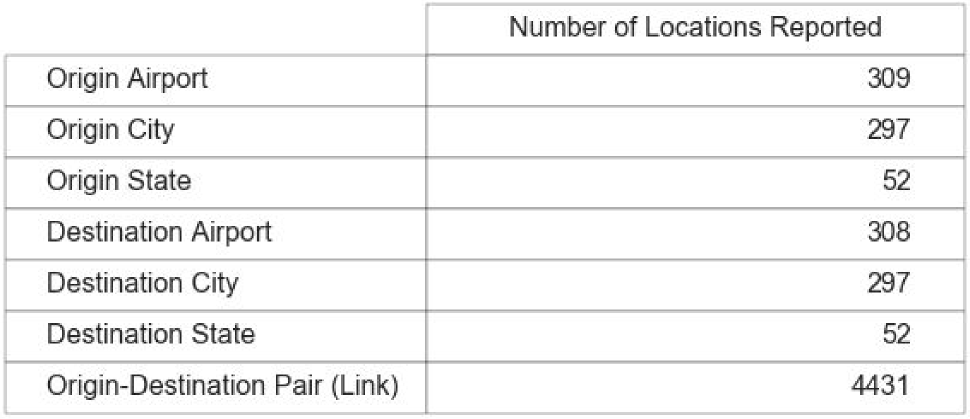
This collection of plots shows that flight frequency doesn’t vary much by temporal category, with the exception of hour of day, as it appears very few flights occur between midnight and 4 am.

1.2 Frequency by Carrier



Recall, 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. This bar graph shows that the top three carriers (Southwest, Delta, and American Airlines) comprise of over 50% of all domestic commercial airline traffic during the sample period.

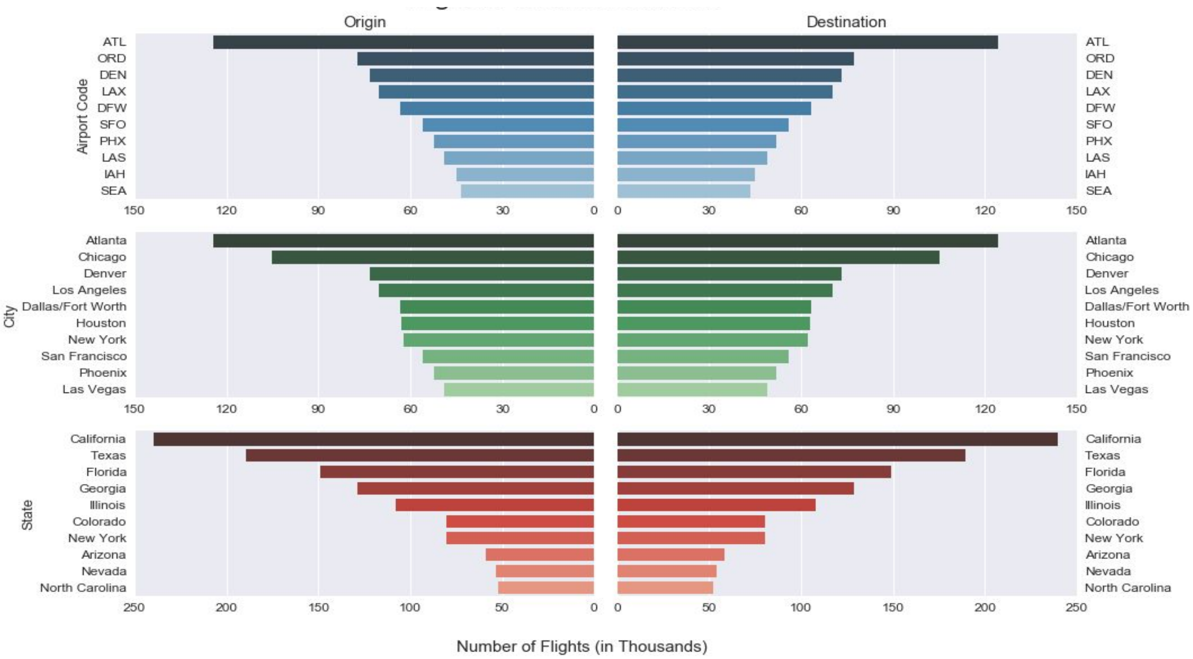
1.3 Frequency by Location



	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

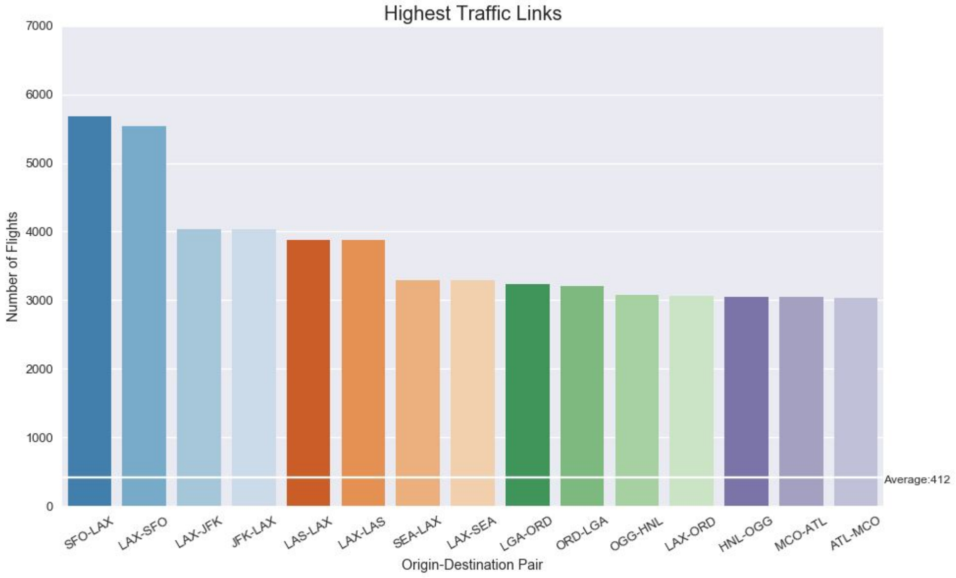
Highlights from this table are that there are 4431 unique origin-destination pairs, and one airport that only had an outgoing flight, and no incoming flights. That outlier was New Castle Airport in Delaware, which is not a commonly used major commercial airport, as the state of Delaware is small and direct flights are typically directed to Philadelphia International Airport instead.

1.3 Highest Traffic Locations



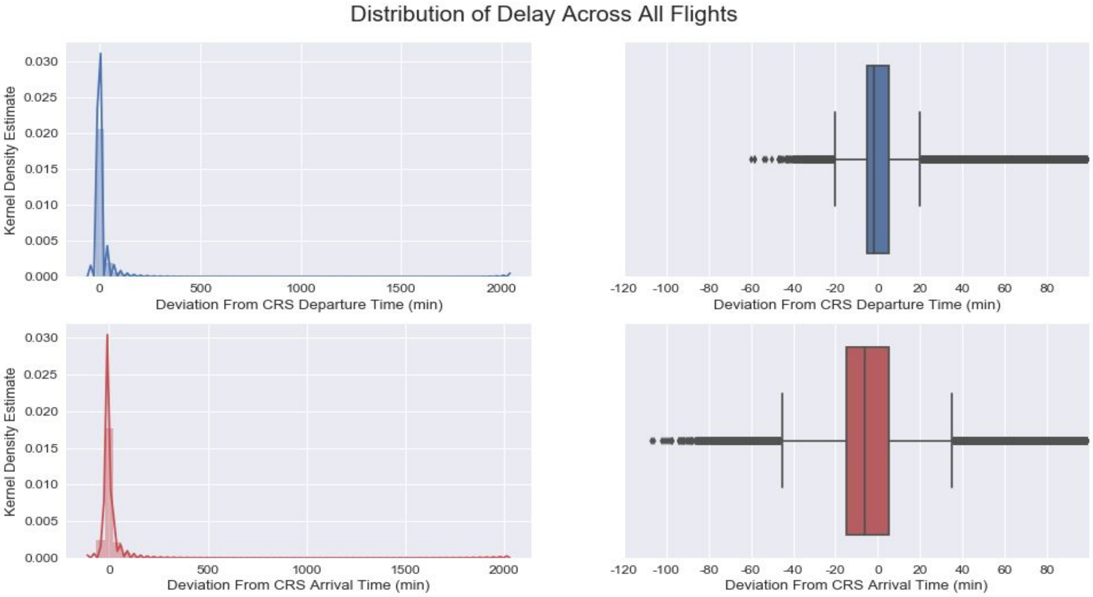
The highest traffic origins and destinations were identical in all categories as all flights were counted in each instance, as the origin in one and the destination in another. Notable takeaways being that even among the ten highest traffic locations, the disparity between the top location and the next closest ones was considerable.

1.3 Highest Traffic Origin-Destination Pairs



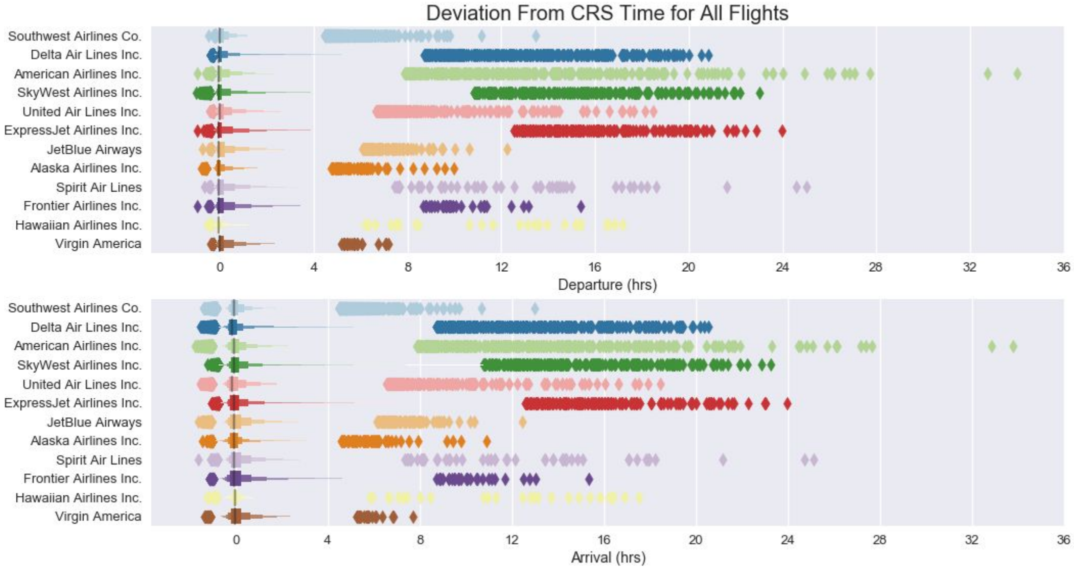
This bar graph shows that the most frequented origin-destination pairs had traffic significantly greater than the average pair, with the highest traffic link being over ten times greater.

2.1 Overall Delay Distribution



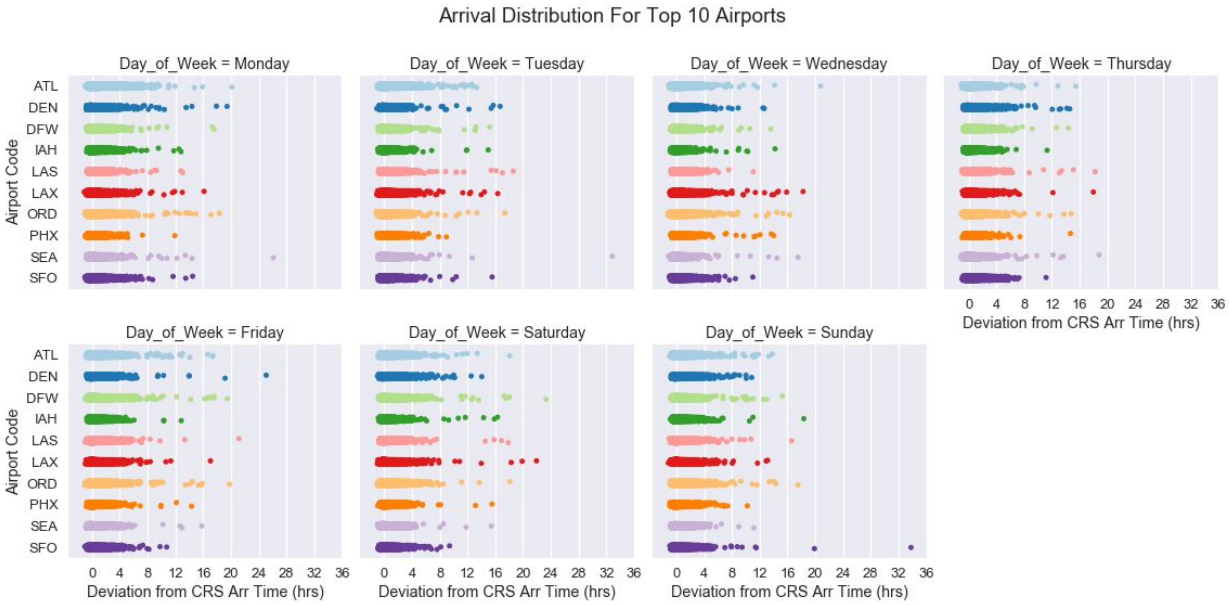
Kernel Density Estimates and Boxplots of the deviation from CRS arrival and departure time for all flights showed that the majority of flight delays were below 20 min, and significant outliers occurred in both distributions. A larger IQR for the deviation from CRS arrival time showed greater variability for arrival delays than departure delays, with the median for both distributions being below zero.

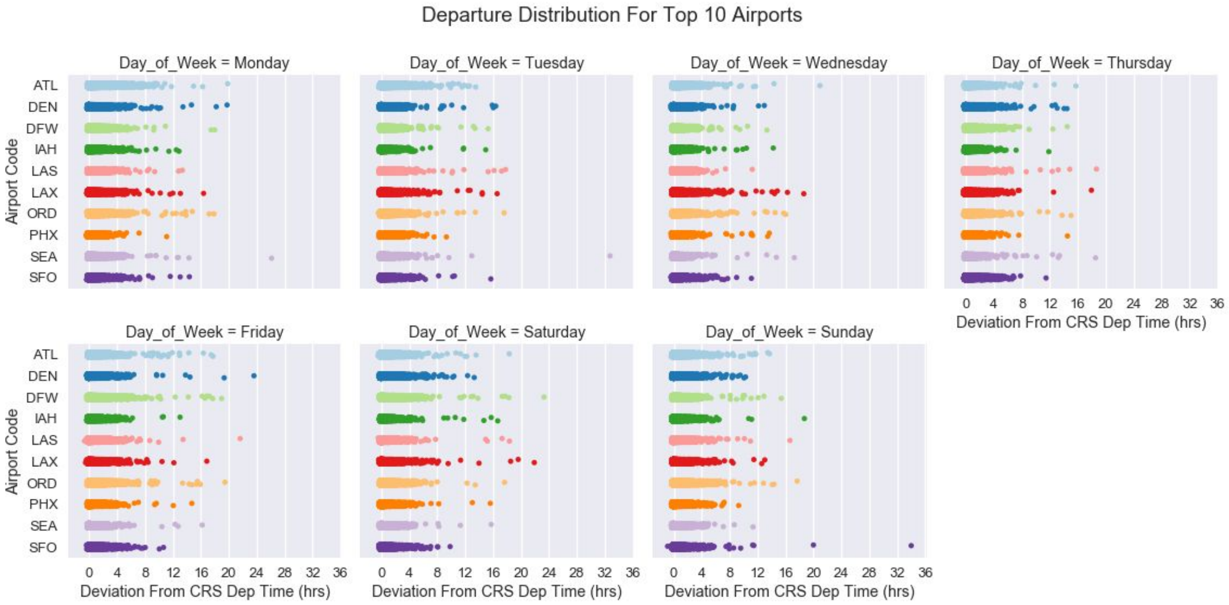
2.2 Distribution of Delay by Carrier



An enhanced Boxplot, also known as a “letter value” plot, of the distribution of arrival and departure delay for all flights by unique carrier showed they all had instances of significant delay, with Virgin America notably being the only carrier not to have a flight delayed greater than 8 hours.

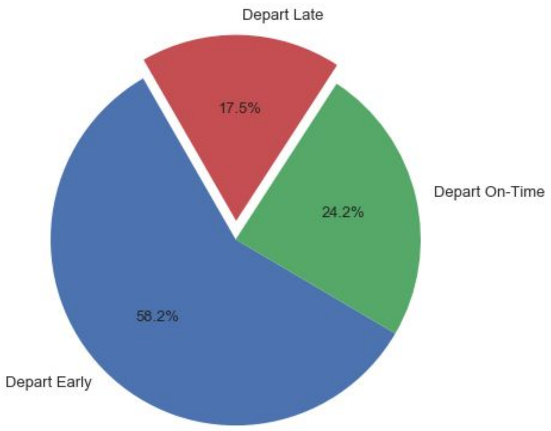
2.3 Distribution of Delay for Highest Traffic Airports

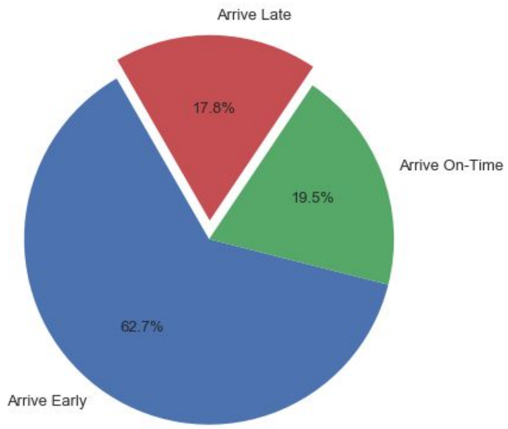


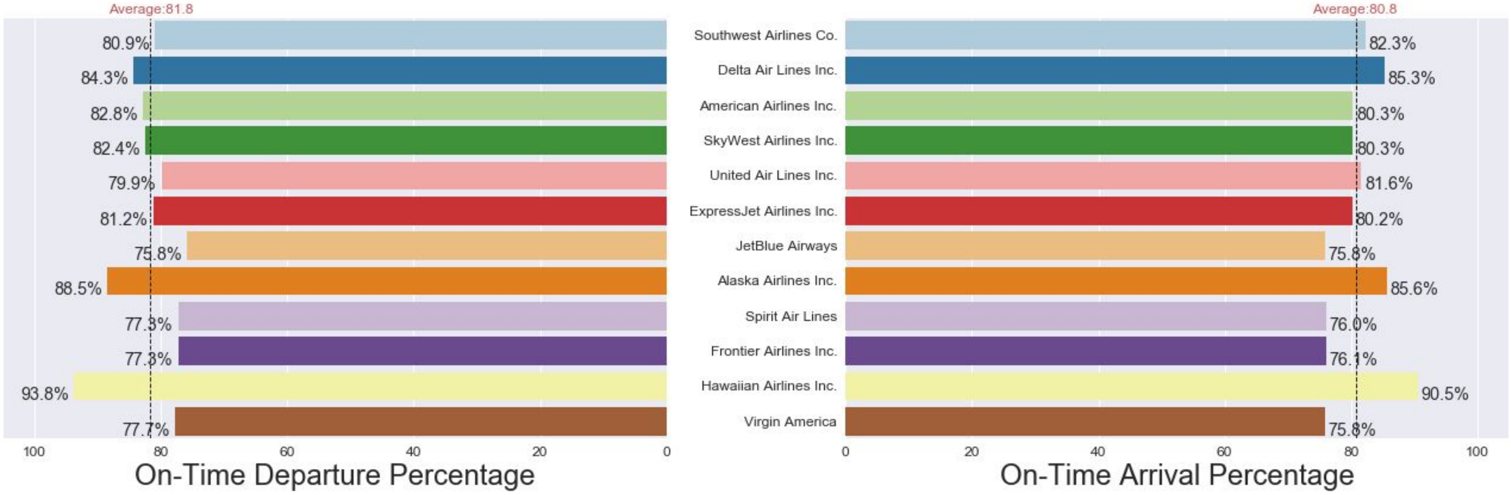


Scatterplots of the deviation from the scheduled arrival and departure times for the top ten airports showed that delay did not vary greatly by day of week, although the most extremely delayed flights (20+ hours past CRS time) occurred primarily on weekends.

3. On-Time Performance For All Flights



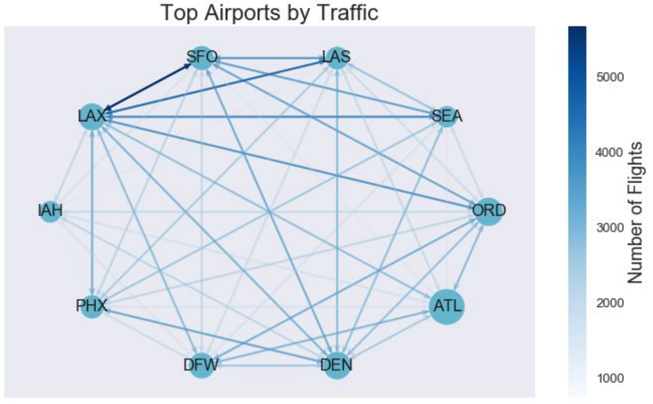




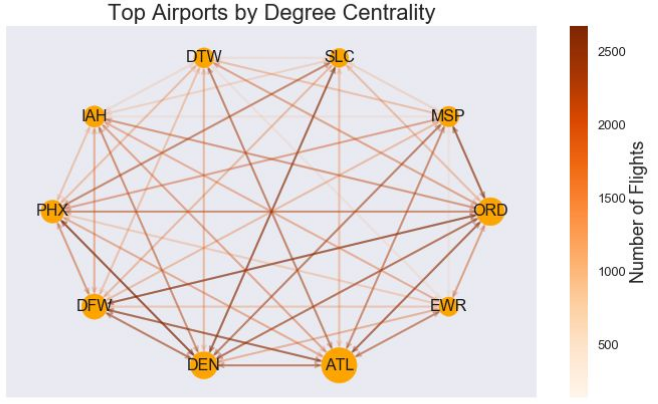
On-time performance is a commonly used metric in measuring the effectiveness of the system, with a flight being counted as "on time" if it operated less than 15 minutes after the scheduled time shown in the carriers' Computerized Reservation Systems (CRS), with departure and arrival times being calculated from gate to gate, not including taxi or airtime. Airlines are typically considered to perform well when their on-time performance reaches 90%, and during the current sample period only Hawaiian Airlines accomplishes this feat; and is one of the two carriers in the bottom half of the major carriers list to surpass the average on-time percentage for arrival and delay (the other being Alaska Airlines).

4. Network Visualization

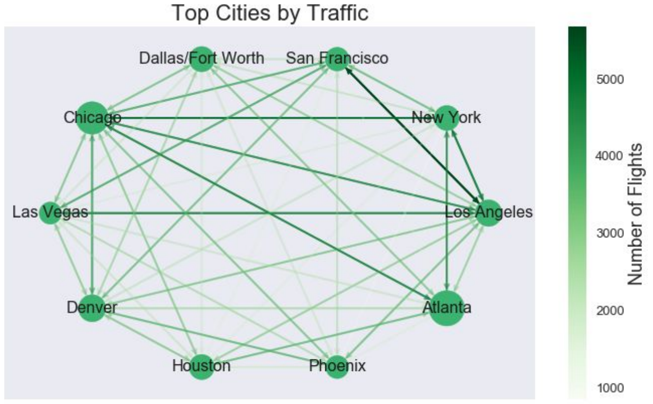
Top Airports by Traffic



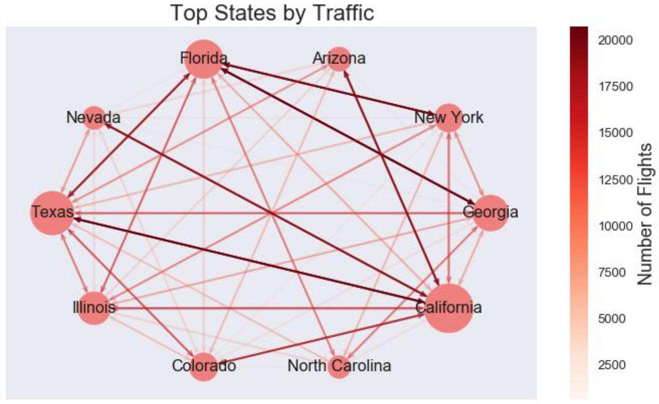
Top Airports by Degree Centrality



Top Cities by Traffic



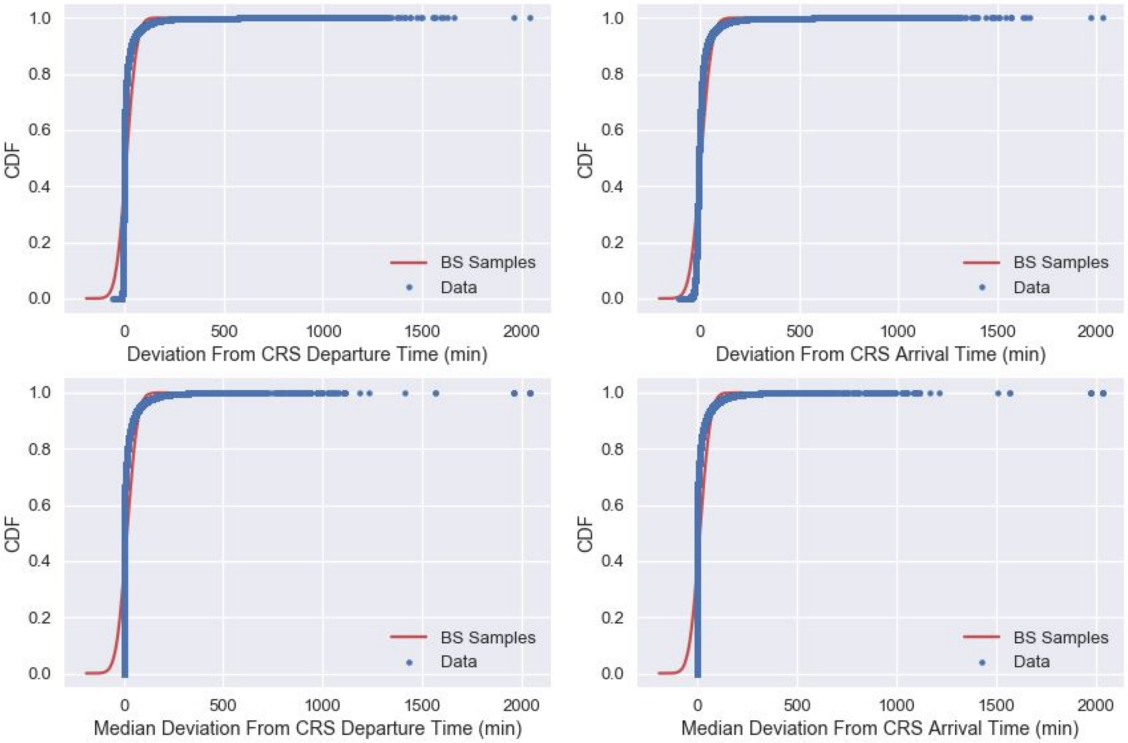
Top States by Traffic



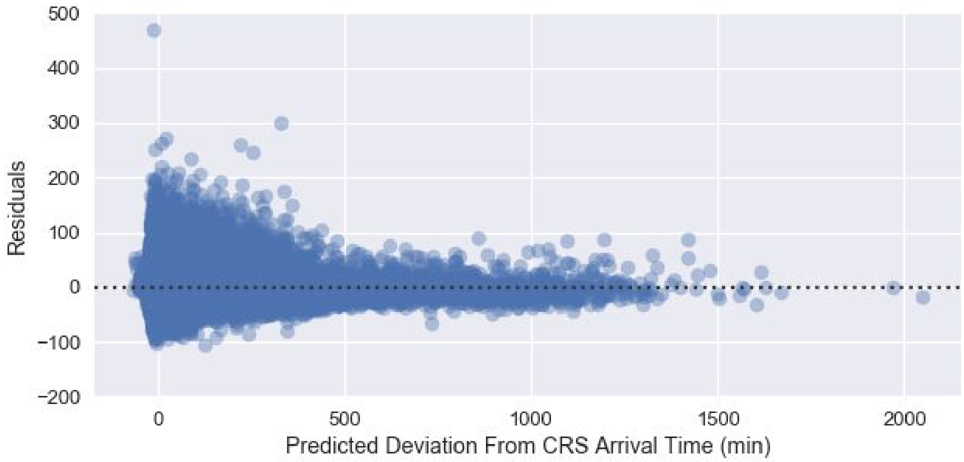
Subgraphs of the NetworkX DiGraph created from the flights dataframe were plotted to show various relationships between the top nodes for each attribute, with the saturation of the edge color correlating to the number of flights between the two nodes and the size of the nodes correlating to their ranking. Notable takeaways are the strong relationship between the San Francisco and Los Angeles airports, and the importance of the Atlanta airport in the overall network; as well as the fact that the most connected (high degree centrality) nodes differed from the top nodes by traffic, probably due to their key geographic locations.

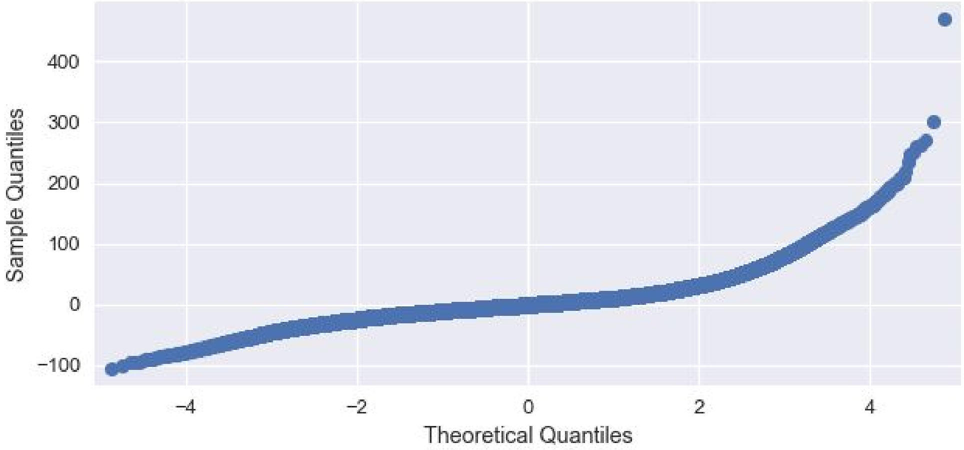
VI. Inferential Statistics

After cleaning the data and screening the features and their relationships through visualization, certain variables were deemed to be of interest as a they pertain directly to the problem at hand, the deviation from CRS arrival and departure times for all flights, and the corresponding median deviation from CRS times per origin-destination pair in the mock network dataframe Links_d. These variables are of direct consequence as it is the propagation dynamic that is integral in using current delay times to predict a future delay status. The central limit theorem was deemed applicable in this case as the sample size is very large and the probability density function of each variable showed observations are independent. Furthermore, a cumulative density function was plotted for each of them explicitly showing they possessed normal distributions.



A brief regression analysis was also conducted for the purpose of validating a directed network approach utilizing only departure delay as an indicator of overall (departure and arrival) delay. From which it was concluded that in each categorical date value, deviation from CRS departure and deviation from CRS arrival time shared a strong positive relationship, and a plot of the residuals vs predicted values displayed a random arrangement further supporting the use of a linear model. Furthermore, a quantile plot showed that the datasets were heavily skewed, and it was deemed that robust methods should be utilized in model construction to lessen the influence of extreme values.



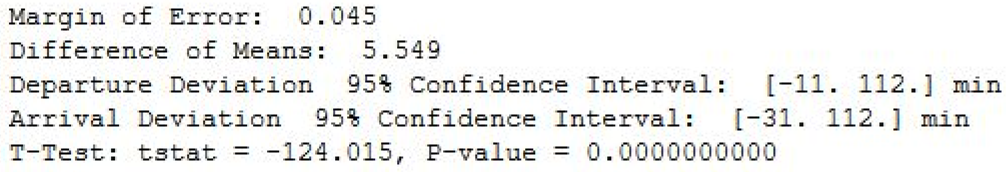


Additional hypotheses were tested to minimize the risk of incorrect or misleading results, a T-Test was conducted to verify that the difference between arrival and departure deviation means was significant for all flights; and a Permutation test for Pearson's r between the median deviation from CRS time for departure and arrival in the mock network to cement further analysis built upon the two having a deterministic relationship.

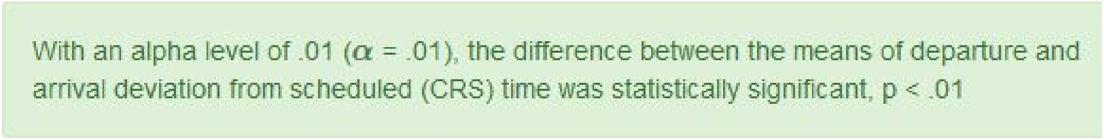
1. **T-Test of Arrival and Departure Deviation Means**

$$\mathrm{H}_{\mathrm{o}} : \mu_d = \mu_a$$

$$\mathrm{H}_{\mathrm{a}} : \mu_d \neq \mu_a$$



```
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
```

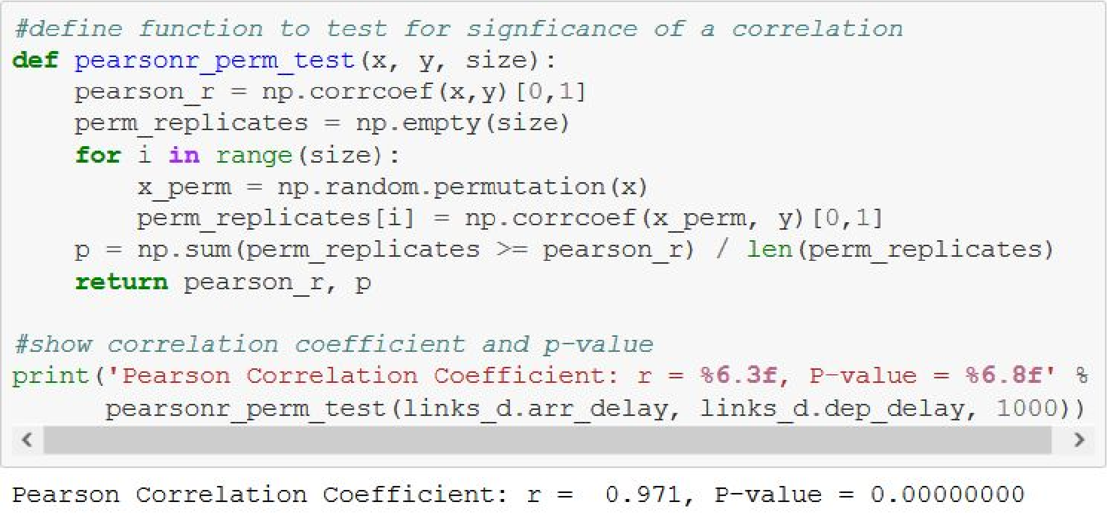


With an alpha level of .01 (α = .01), the difference between the means of departure and arrival deviation from scheduled (CRS) time was statistically significant, p < .01

2. Permutation Test for Pearson’s r between Median Arrival and Departure Delay

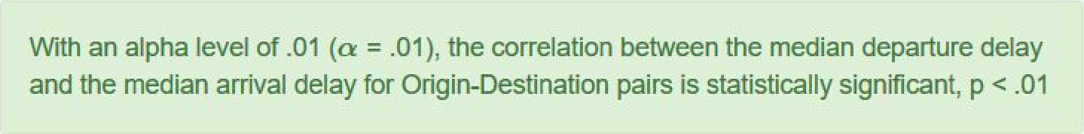
H$_o$: *The correlation between the current median departure delay and median arrival delay for an origin-destination pair is not significant*

H$_a$: *The correlation between the current median departure delay and median arrival delay for an given origin-destination is significant*

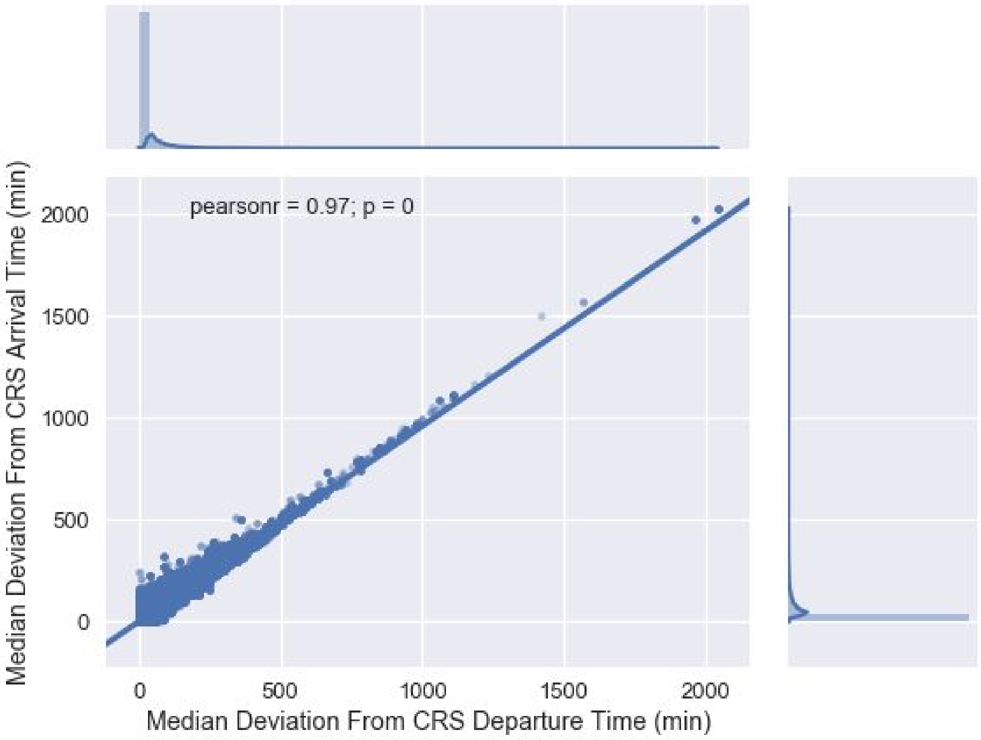


```
#define function to test for signficance of a correlation
def pearsonr_perm_test(x, y, size):
    pearson_r = np.corrcoef(x,y)[0,1]
    perm_replicates = np.empty(size)
    for i in range(size):
        x_perm = np.random.permutation(x)
        perm_replicates[i] = np.corrcoef(x_perm, y)[0,1]
    p = np.sum(perm_replicates >= pearson_r) / len(perm_replicates)
    return pearson_r, p

#show correlation coefficient and p-value
print('Pearson Correlation Coefficient: r = %6.3f, P-value = %6.8f' %
      pearsonr_perm_test(links_d.arr_delay, links_d.dep_delay, 1000))
```

```
Pearson Correlation Coefficient: r =  0.971, P-value = 0.00000000
```



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



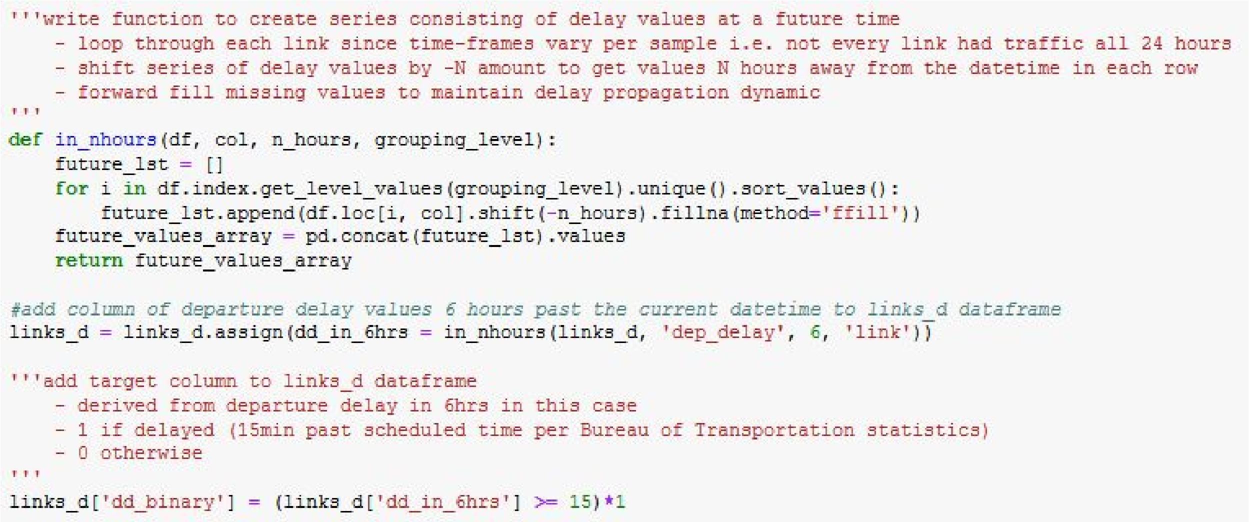
VII. Model Construction

Problem Statement Recap:

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).

1. Target Variable Description

A 15 min threshold was selected to coincide with the Bureau of Transportation statistics definition of flight delay, and a 6 hour prediction horizon was chosen because earlier analysis had shown that the vast majority of flights that are classified as such are typically delayed for less than an hour, so a 6 hour window would be enough to provide practical value for those seeking to anticipate delays and ameliorate subsequent issues.



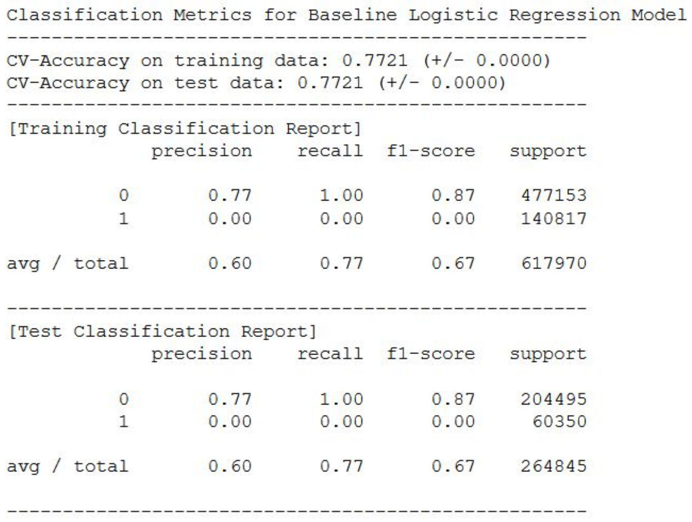
```
'''write function to create series consisting of delay values at a future time
    - loop through each link since time-frames vary per sample i.e. not every link had traffic all 24 hours
    - shift series of delay values by -N amount to get values N hours away from the datetime in each row
    - 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
```

2. Baseline - Logistic Regression Classifier

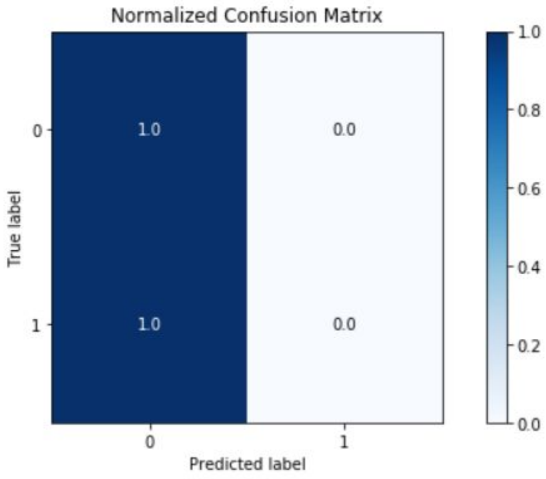
Procedure:

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

2. Baseline - 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

----------------------------------------------------
```



Due to the imbalanced distribution of the delayed class in the target variable, 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 (0.772 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.

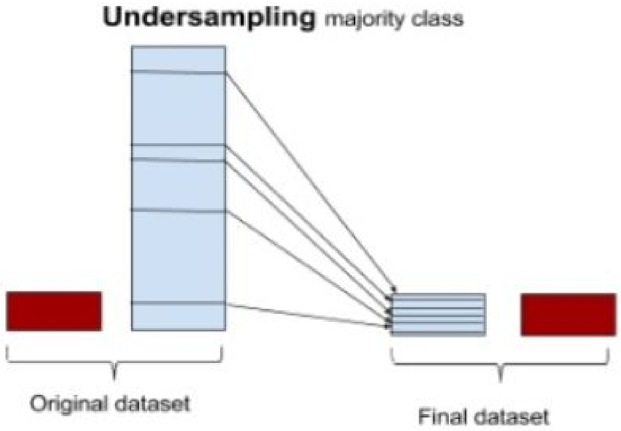
Moving Forward - Resample to counter class imbalance

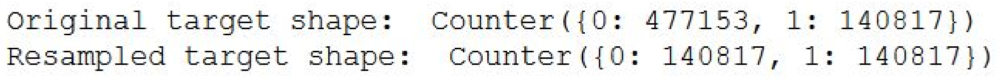
Procedure:

1. Utilize Under and Over-sampling techniques on the training dataset
2. Train each method on Random Forest 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.

3. Condition I - Random Under Sampling

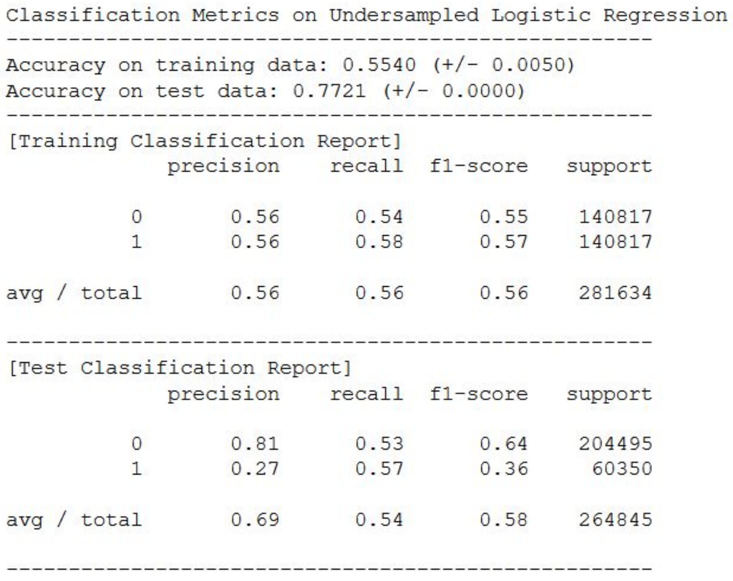
Using RandomUnderSampler from imblearn.under_sampling package, which under samples the majority class in the training dataset by randomly picking samples without replacement.



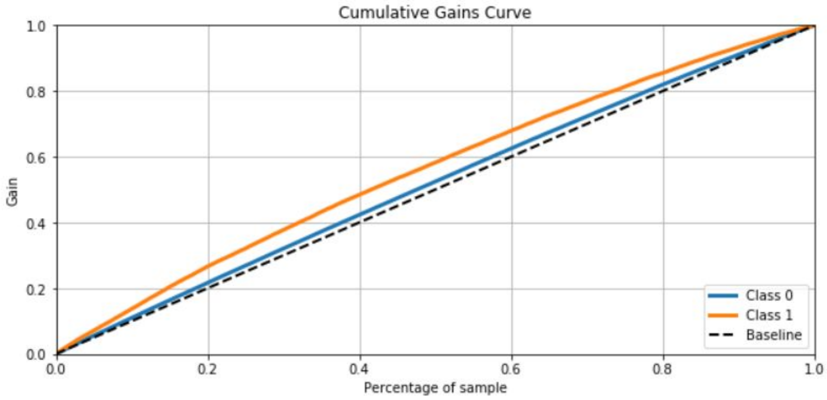


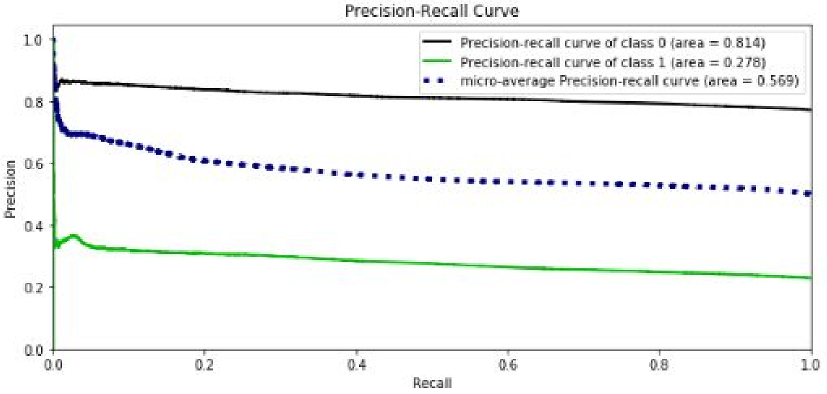
```
Original target shape:  Counter({0: 477153, 1: 140817})
Resampled target shape:  Counter({0: 140817, 1: 140817})
```

3. Condition I - Logistic Regression Classifier

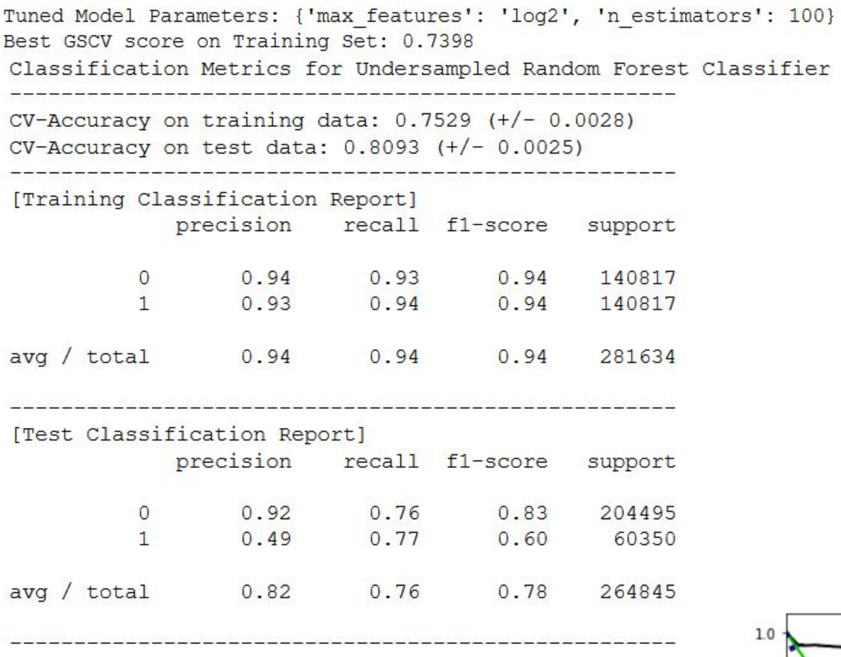


```
Classification Metrics on Undersampled Logistic Regression
----------------------------------------------------
Accuracy on training data: 0.5540 (+/- 0.0050)
Accuracy on test data: 0.7721 (+/- 0.0000)
----------------------------------------------------
[Training Classification Report]
             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

----------------------------------------------------
```

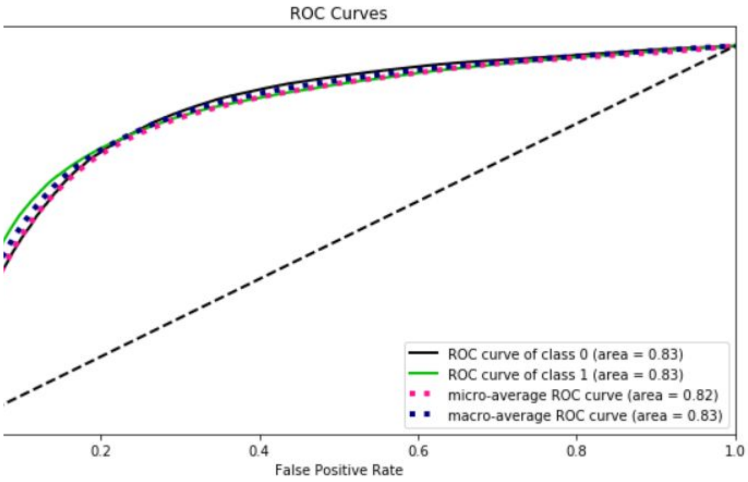


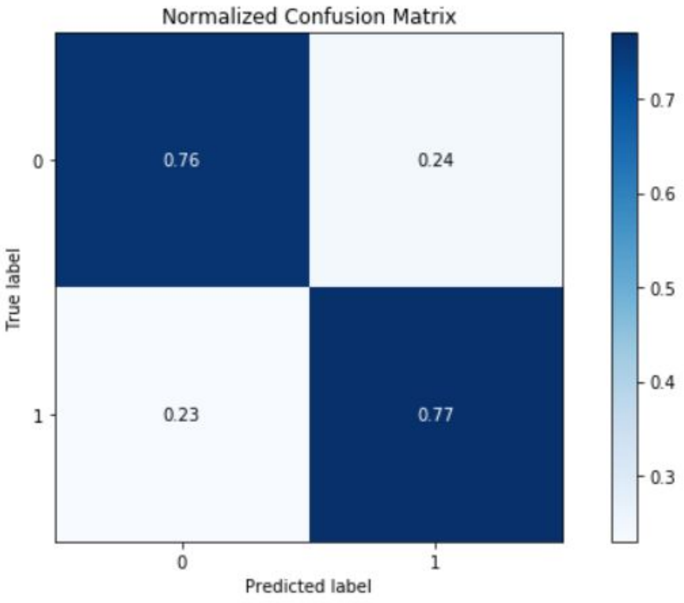


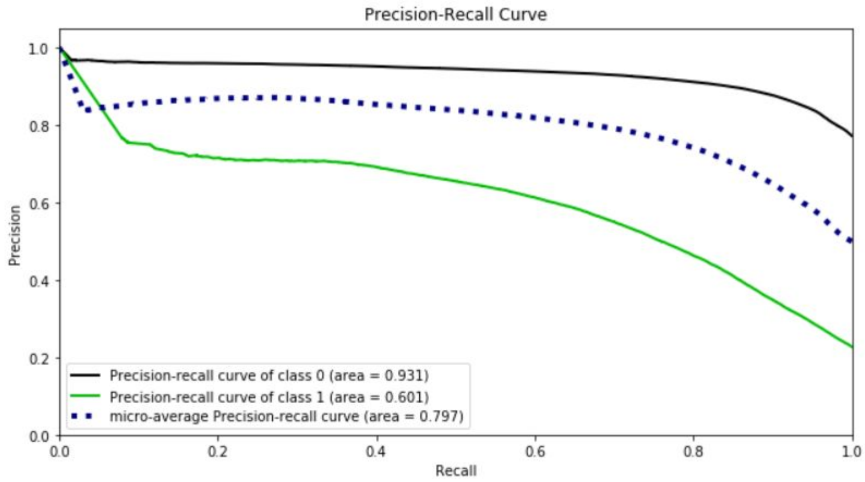
3. Condition I - Random Forest Classifier

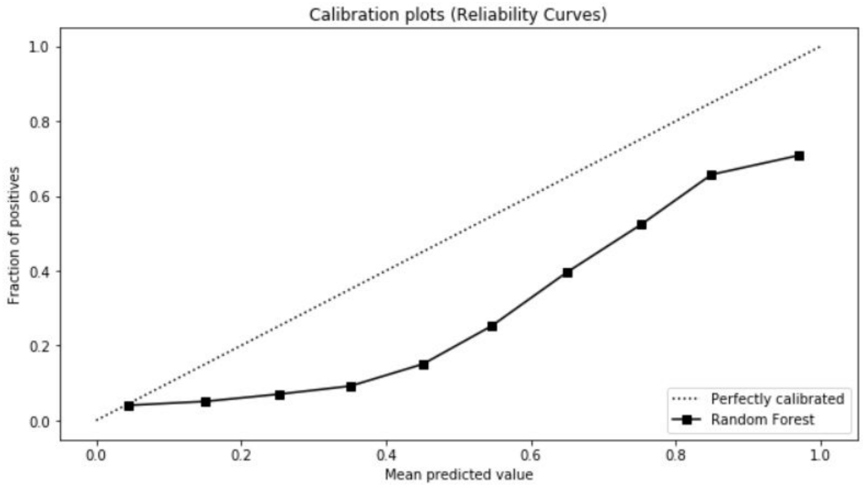


```
Tuned Model Parameters: {'max_features': 'log2', 'n_estimators': 100}
Best GSCV score on Training Set: 0.7398
 Classification Metrics for Undersampled Random Forest Classifier
 ---------------------------------------------------
 CV-Accuracy on training data: 0.7529 (+/- 0.0028)
 CV-Accuracy on test 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

 ---------------------------------------------------
```



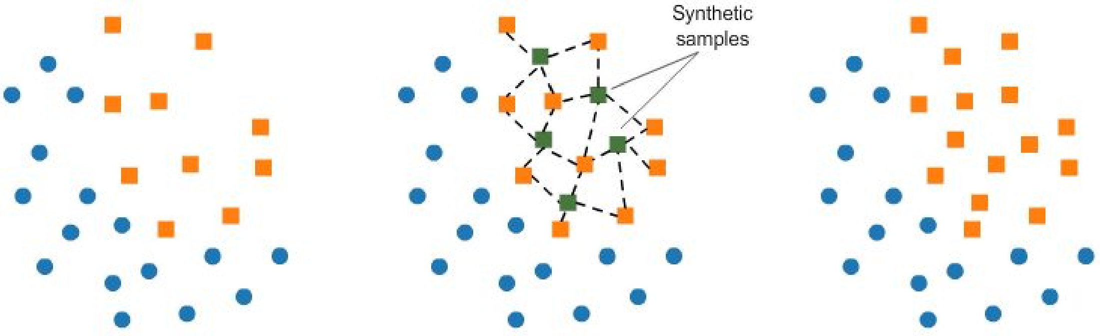


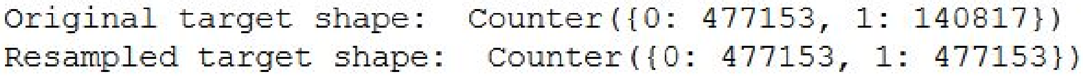




4. Condition II - Synthetic Minority Over-sampling Technique

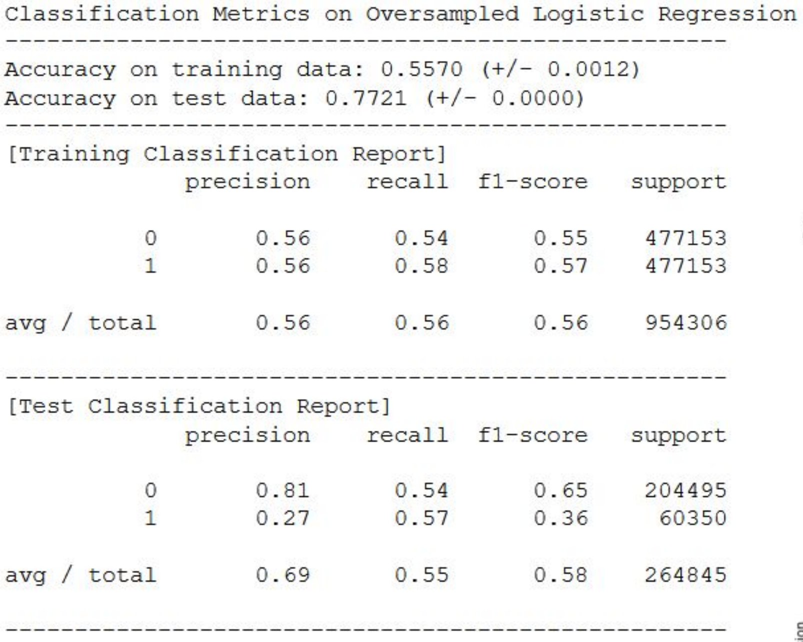
Using SMOTE from imblearn.over_sampling package, which oversamples the minority class in the training dataset by creating synthetic samples, as opposed to creating copies. The algorithm selects k-number of similar instances (using a distance measure), and perturbs a new instance one attribute at a time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



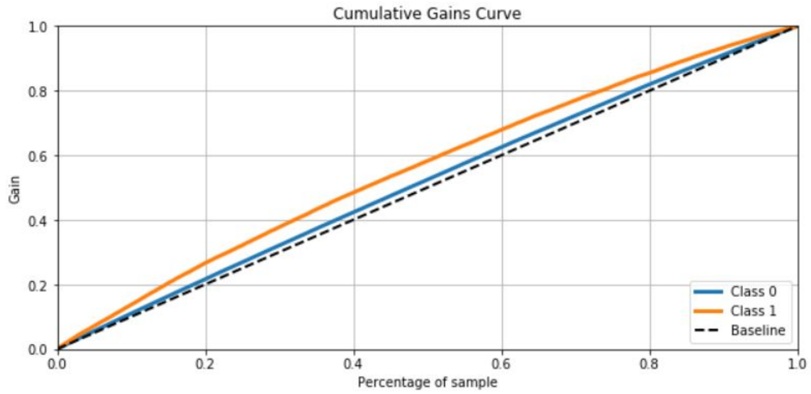


```
Original target shape:  Counter({0: 477153, 1: 140817})
Resampled target shape:  Counter({0: 477153, 1: 477153})
```

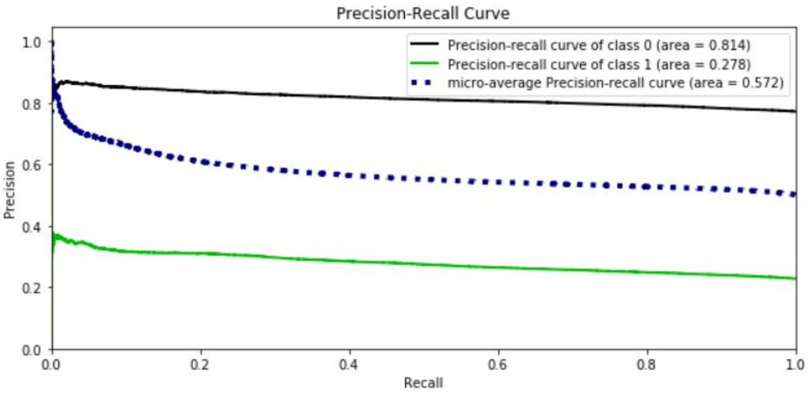
4. Condition II - Logistic Regression Classifier



```
Classification Metrics on Oversampled Logistic Regression
-----------------------------------------------------
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

-----------------------------------------------------
```

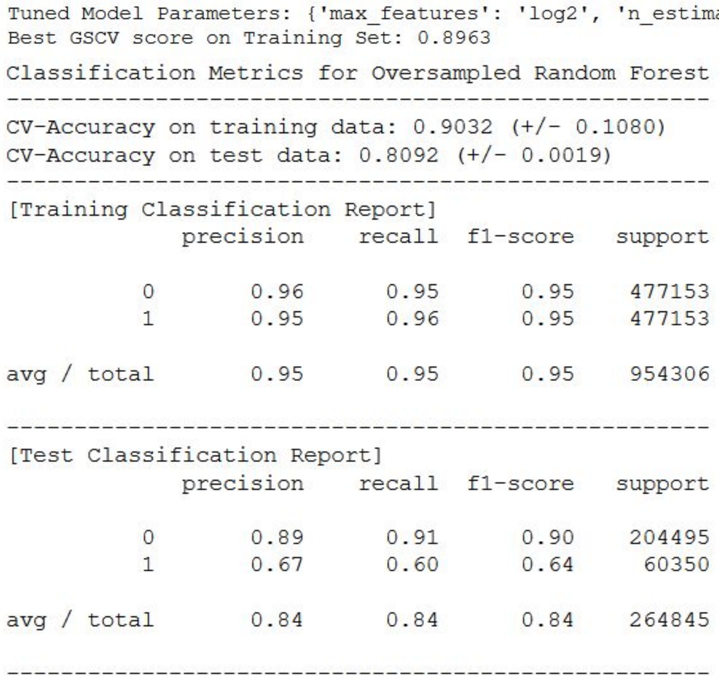


Cumulative Gains Curve

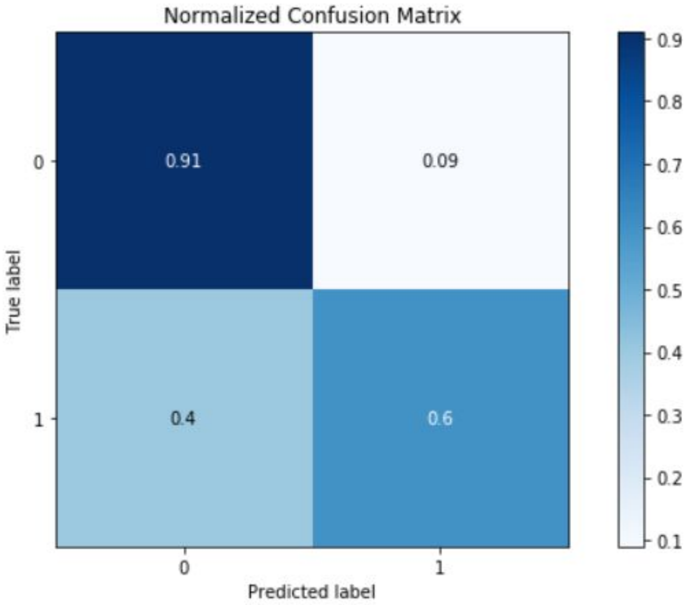


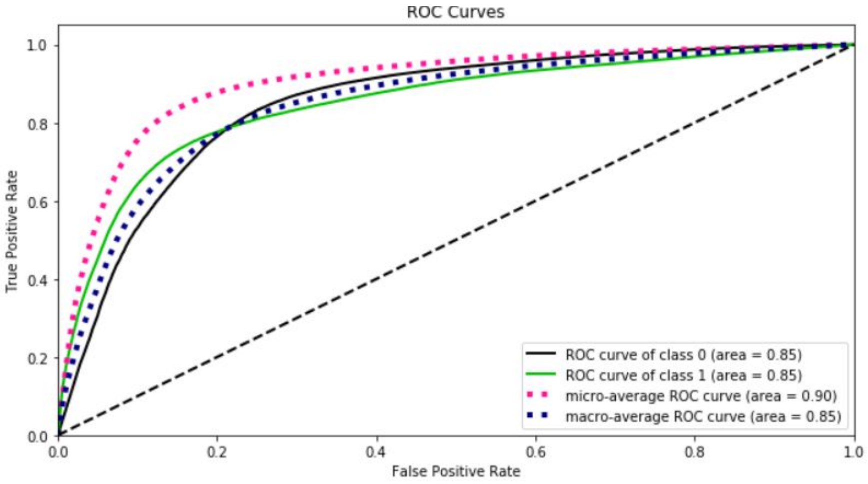
Precision-Recall Curve

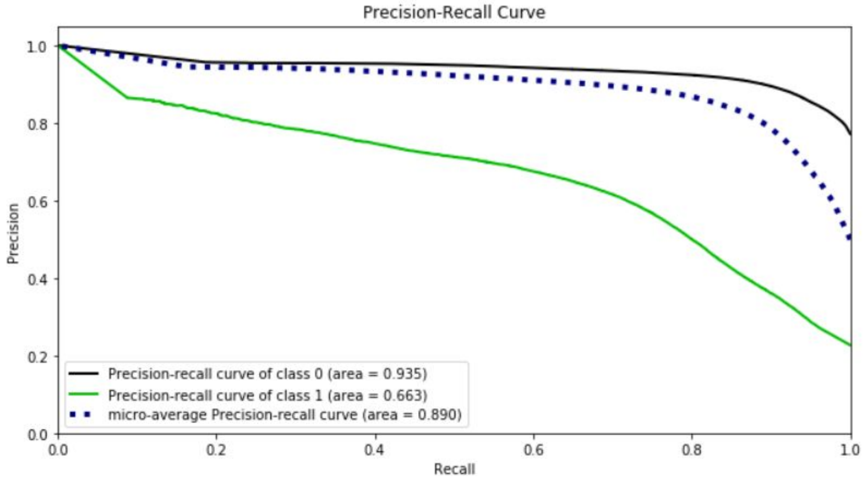
4. Condition II - Random Forest Classifier

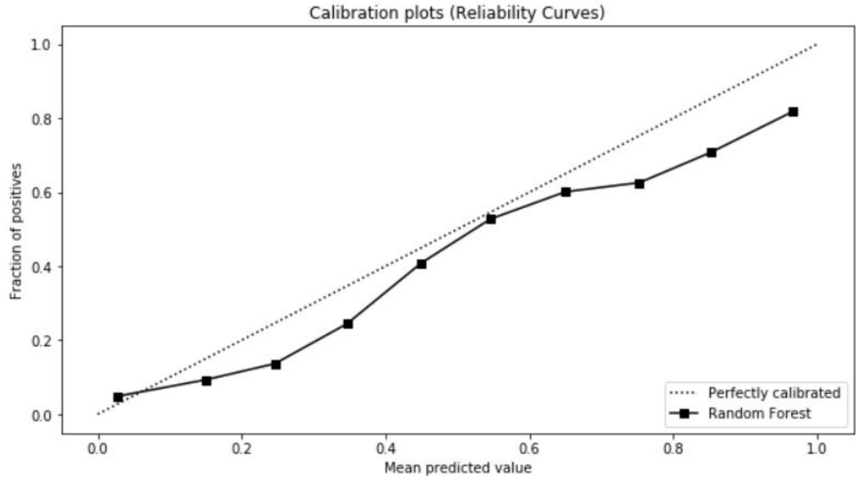


```
Tuned Model Parameters: {'max_features': 'log2', 'n_estimators': 100}
Best GSCV score on Training Set: 0.8963
Classification Metrics for Oversampled Random Forest Classifier
------------------------------------------------------
CV-Accuracy on training data: 0.9032 (+/- 0.1080)
CV-Accuracy on test 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

------------------------------------------------------
```









5. Results

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. The Random Forest Classifier trained on data using the Random Under Sampling technique had a higher Recall score for the delay class (0.77), but considerably lower Precision, with a score of 0.49. Meaning it had a lower false positive rate, but of the additional results being returned, most had incorrect labels in comparison to the training labels, yielding less utility.

VIII. Conclusions and Client Recommendations

Of the four estimators trained, the logistic regression models performed the worst, proving unreliable for this particular problem, although they have potential for improvement if weights were added to the classes. Conversely, the ensemble approach appears to more appropriate for this problem, and of the two Random Forest Classifiers, the one trained on the SMOTE dataset yielded the most useful results, and would be my recommendation to a potential client seeking to predict a delay state on any particular origin-destination pairs. Implementation of this model would be possible for any of the potential clients delineated earlier, as all would be recipients of ASDI (real-time position and flight plans in U.S. airspace) data, but for the purpose of providing a larger perspective, uses of this model can be split into three categories grouped by practitioner:

(1) For data distributors:

The predictive component provided by this model can be manipulated and re-trained into different classifier packages by utilizing alternative prediction horizons (besides 6 hours in the future), and thresholds for delay (15 min was used in this instance), allowing users to have a customized view into future delay states.

(2) For air-traffic controllers:

By having a concise prediction of delay states on origin-destination pairs available, which requires minimal calculation on the part of Traffic Management Personnel in comparison to current practices, flight plans and Ground Delay Programs can be crafted with a forward-looking approach, reducing Airport Arrival Rates and preventable propagated delays.

(3) For airport ground crews:

Proverbially the frontlines of the delay problem when it comes to air-traffic networks, the simple yet useful output of this model has the potential to yield the earliest meaningful results for this group, as even a few hours of preparation can result in substantial gain in terms of avoiding wasted manpower, and establishing relief efforts in congested runways.