Machine Learning Engineer Nano-degree
Capstone Project
Predicting arrival delays for flights in the US.

Project Overview

Airline travel is increasingly becoming a primary way of travel for people over medium and long distances. The increased air travel and congested plane slots at the airport result in sometimes significant delays. This is particularly true during the high travel season, not to mention weather related delays. Is it possible to discern a pattern in airline arrival delays and build a model that can learn and predict delays based on historical data?

In this project we attempt to forecast and attribute flight delays in the domestic US market. In general, flights may be delayed for various reasons. Flights may get diverted or canceled due adverse weather, may get delayed due to aircraft maintenance issues, or labor unrest or just plain congestion in the air or at the airports. In many instances planes may depart from their origin later than the publicized departure time but airlines usually build in a guard band in the schedule and a late departure may not pre-ordain a late arrival. Also delayed departures can be made up during the actual flight time by flying the aircraft faster (burning more fuel etc.) There are many parts in this puzzle and we will try to pinpoint the main factors that can explain the later arrivals. We will then build a model to train and see if it can accurately predict on a held out set. This is the Machine Learning part. Arguably some airlines are better at managing delays than others and we will take a look at the data to see which airlines manage delays better and which routes in particular are delay prone. This is the Data Analysis part.

Getting the data

In this project, we will evaluate the performance and predictive power of various models that have been trained and tested on data collected by the Bureau of Transportation Statistics (BTS.) A model trained on this data that is seen as a *good fit* could then be used to make certain predictions about a flight delays. All the data that we have taken can be found at the following web address Government website.
The guide for each of the fields is as below:

Field	Description
`FlightDate`	Flight Date (yyyymmdd)
`UniqueCarrier`	Unique Carrier Code. When the same code has been used by multiple carriers, a numeric suffix is used for earlier users, for example, PA, PA(1), PA(2). Use this field for analysis across a range of years.
`FlightNum`	Flight Number
`OriginAirportID`	Origin Airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused.

OriginCityMarketID	Origin Airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market.
OriginCityName	Origin Airport, City Name
DestAirportID	Destination Airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused.
DestCityMarketID	Destination Airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market.
DestCityName	Destination Airport, City Name
DepTime	Actual Departure Time (local time: hhmm)
DepDelay	Difference in minutes between scheduled and actual departure time. Early departures show negative numbers.
ArrTime	Actual Arrival Time (local time: hhmm)
ArrDelay	Difference in minutes between scheduled and actual arrival time. Early arrivals show negative numbers.
AirTime	Flight Time, in Minutes
Flights	Number of Flights
Distance	Distance between airports (miles)
CarrierDelay	Carrier Delay, in Minutes
WeatherDelay	Weather Delay, in Minutes
NASDelay	National Air System Delay, in Minutes
SecurityDelay	Security Delay, in Minutes
LateAircraftDelay	Late Aircraft Delay, in Minutes

Table 1: Description of the keys in our Dataset

A quick check on the dates of the data shows that these data are for the month of January/ 2016. We will analyze the data in more details later to make sure it is not dominated by outliers or disproportionately scaled features.

Description of the data

```
'This is the information about the data types in our flight data that
we read in (un) pre-processed'

FL_DATE                    object
UNIQUE_CARRIER             object
```

```
FL_NUM                    int64
ORIGIN_AIRPORT_ID         int64
ORIGIN_CITY_MARKET_ID     int64
ORIGIN_CITY_NAME         object
DEST_AIRPORT_ID           int64
DEST_CITY_MARKET_ID       int64
DEST_CITY_NAME           object
DEP_TIME                float64
DEP_DELAY               float64
ARR_TIME                float64
ARR_DELAY               float64
AIR_TIME                float64
FLIGHTS                 float64
DISTANCE                float64
CARRIER_DELAY           float64
WEATHER_DELAY           float64
NAS_DELAY               float64
SECURITY_DELAY          float64
LATE_AIRCRAFT_DELAY     float64
Unnamed: 21             float64
dtype: object
```

Table 2: Types of data in our dataset.

Problem Statement

A cursory investigation about the Flight delay data shows that we need to separate the dataset into **features** and the **target variable**. The **target variable**, `'ARR_DELAY'`, will be the variable we seek to predict. All the columns other than the target variable end up as the **features**. Some of the features give us quantitative information and some give us qualitative information about each data point. These are stored in `feature_set`.

Metrics

We carry out two kinds of analysis on our data. The first analysis is regression and the second analysis is classification. For the regression we evaluate the goodness of fit using the regression score which is a built in method of the regression model. The regression score according to the Scikit learn website is a R^2 metric. The definition of the metric from the website is as follows:

*"The coefficient R^2 is defined as (1 - u/v), where u is the regression sum of squares ((y_true - y_pred) ** 2).sum() and v is the residual sum of squares ((y_true - y_true.mean()) ** 2).sum(). Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0."*

For the classification we valuate the goodness of classification using the F-Score from the Scikit learn website which is defined as :

```
F1 = 2 * (precision * recall) / (precision + recall)
```

For the Grid search Cross Validation for parameter tuning we get use the r2_score from the Scikit learn website which is defined as:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

then the variability of the data set can be measured using three sums of squares formulas:

- The total sum of squares (proportional to the variance of the data):
$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$
- The regression sum of squares, also called the explained sum of squares:
$$SS_{\text{reg}} = \sum_i (f_i - \bar{y})^2,$$
- The sum of squares of residuals, also called the residual sum of squares:
$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

The most general definition of the coefficient of determination is

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}.$$

where y = true values

f = predicted values

e = residuals

Why did we choose these metrics for our evaluation?

R^2 : This metric measures the goodness of fit/predictive power of the model. The metric is hard to understand because the values can range from -ve to 1. The metric is fairly well defined/ understood at distinct values like 0 or 1. However over most of its range it behaves non-linearly and hence is very hard to understand. This is the default score method of the DecisionTreeRegressor and the RandomForestRegressor and hence we used it. Another metric to use is RMSE, which is more intuitive and easier to understand and which we added after getting feedback on the project. RMSE makes it easier to compare models that have similar or close R^2 scores.

F-Score : This metric is a "complete" metric that can be used to evaluate any classifier model. Accuracy is an inadequate metric for a dataset with imbalanced classes with a predominantly negative class. In this case a model can incorrectly predict all the important data points (defined as the few where we need it to be correct) and still be very accurate because it predicted all the negative cases (which are not important) correctly. F-Score incorporates both precision and recall to get around the limitations of accuracy. In our classification case the output labels are unbalanced. The 'early' and 'late' are the predominant class labels and the flights that are more than 60 minutes late ('Very late') are only 5% of the class.

Data Exploration

Next we carry out some data exploration. For detailed analysis please look @ the iPython notebook.

```
'Descriptive Statistics for Arr/Dep delays'
```

	ARR_DELAY	DEP_DELAY	DISTANCE	AIR_TIME
count	433298.000000	433298.000000	433298.000000	433298.000000
mean	1.533654	7.755997	843.752214	116.395818
std	39.096842	36.730931	609.423956	72.981191
min	-79.000000	-47.000000	31.000000	8.000000
25%	-15.000000	-5.000000	391.000000	62.000000
50%	-7.000000	-2.000000	679.000000	98.000000
75%	5.000000	5.000000	1089.000000	148.000000
max	1659.000000	1663.000000	4983.000000	698.000000

Table 3: Table for the descriptive statistics for Arr/Dep Delays.

Our Dataset has the following number of data points:

```
Flight Delay dataset has 445827 data points with 22 variables each.
```

Exploratory Visualization

We find out the longest and shortest flights.

Longest:
http://weekendblitz.com/top-7-longest-domestic-us-flights

Shortest:
Google Maps (google maps doesn't show a flight exists. What does google know? ;)

'Display the Longest distance flight in our Database'

FL_DATE	UNIQUE_CARRIER	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN_CITY_MARKET_ID	ORIGIN_CITY_NAME	DEST_AIRPORT_ID	DEST_CITY_MARKET_ID	DEST_CITY_NAME	DEP_TIME	DEP_DELAY	ARR_TIME	ARR_DELAY	AIR_TIME	FLIGHTS	DISTANCE	CARRIER_DELAY	WEATHER_DELAY	NAS_DELAY	SECURITY_DELAY	LATE_AIRCRAFT_DELAY
1/23/16	HA	50	12173	32134	Honolulu, HI	12478	31703	New York, NY	2024	244	1125	270	551	1	4983	26	244	0	0	0

Table 4: Table for the longest (Distance) non-stop Domestic flight operated in all 50 states

Longest non-stop flight operated in all 50 states:

Winner: New York (JFK) to Honolulu (HNL) at **4,983 miles** in 11 hours and 40 minutes. Operated by Hawaiian Airlines.

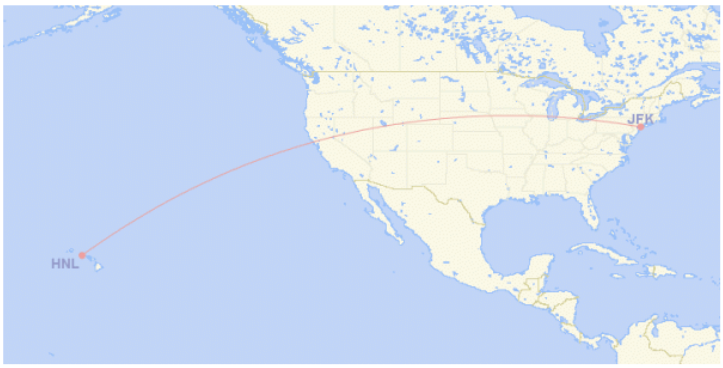

Figure 1: Longest (Distance) non-stop Domestic flight operated in all 50 states

'Display the Shortest distance flight in our Database'

FL_DATE	UNIQUE_CARRIER	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN_CITY_MARKET_ID	ORIGIN_CITY_NAME	DEST_AIRPORT_ID	DEST_CITY_MARKET_ID	DEST_CITY_NAME	DEP_TIME	DEP_DELAY	ARR_TIME	ARR_DELAY	AIR_TIME	FLIGHTS	DISTANCE	CARRIER_DELAY	WEATHER_DELAY	NAS_DELAY	SECURITY_DELAY	LATE_AIRCRAFT_DELAY
1/24/16	AS	65	15841	35841	Wrangell, AK	14256	34256	Petersburg, AK	1057	-9	1140	10	15	1	31	NaN	NaN	NaN	NaN	NaN

Table 5: Table for the shortest (Distance) non-stop Domestic flight operated in all 50 states

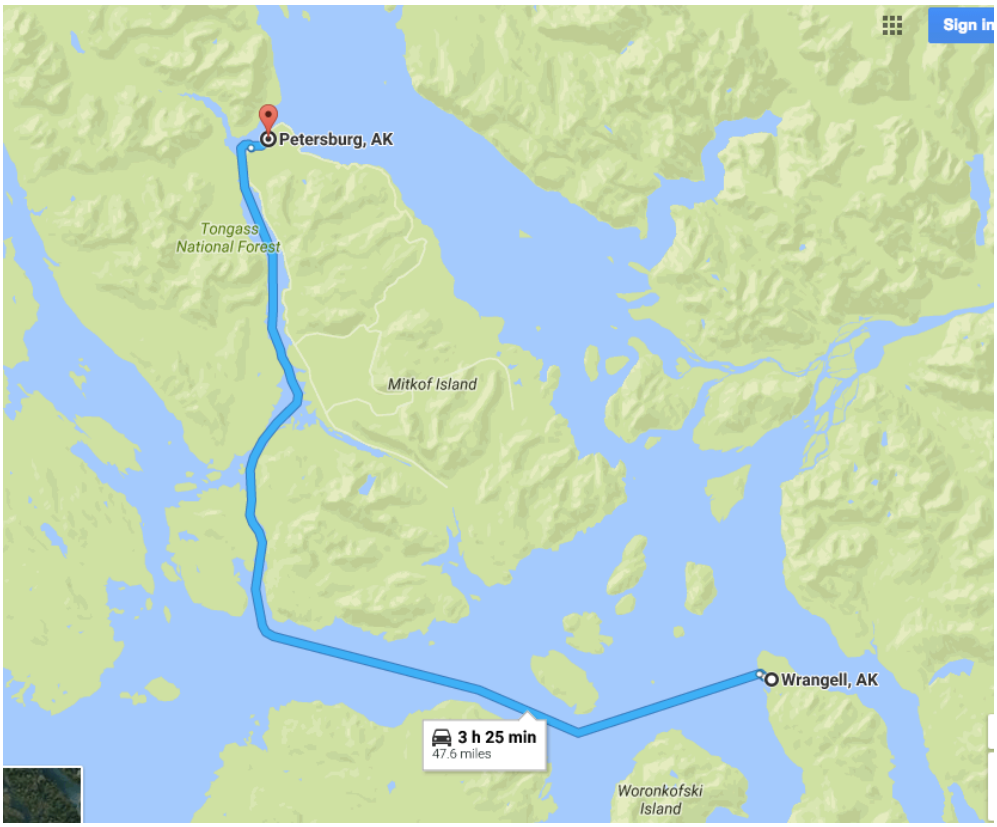

Figure 2: Shortest (Distance) non-stop Domestic flight operated in all 50 states

'Display the Longest flying time flight in our Database'

FL_DATE	UNIQUE_CARRIER	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN_CITY_MARKET_ID	ORIGIN_CITY_NAME	DEST_AIRPORT_ID	DEST_CITY_MARKET_ID	DEST_CITY_NAME	DEP_TIME	DEP_DELAY	ARR_TIME	ARR_DELAY	AIR_TIME	FLIGHTS	DISTANCE	CARRIER_DELAY	WEATHER_DELAY	NAS_DELAY	SECURITY_DELAY	LATE_AIRCRAFT_DELAY
1/12/16	HA	51	12478	31703	New York, NY	12173	32134	Honolulu, HI	955	-5	1655	30	698	1	4983	30	0	0	0	0

Table 6: Table for the longest (Time) non-stop Domestic flight operated in all 50 states

Longest non-stop flight operated in all 50 states:

Winner: New York (JFK) to Honolulu (HNL) at **4,983 miles** in 11 hours and 40 minutes. Operated by Hawaiian Airlines.

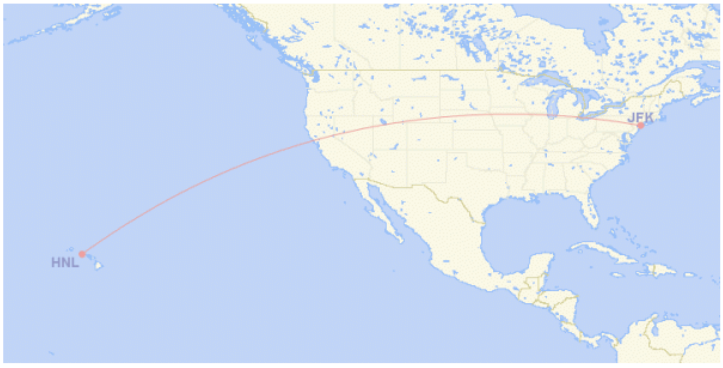

Figure 3: Longest (Time) non-stop Domestic flight operated in all 50 states

'Display the Shortest flying time flight in our Database'

	FL_DATE	UNIQUE_CARRIER	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN_CITY_MARKET_ID	ORIGIN_CITY_NAME	DEST_AIRPORT_ID	DEST_CITY_MARKET_ID	DEST_CITY_NAME	DEP_TIME	DEP_DELAY	ARR_TIME	ARR_DELAY	AIR_TIME	FLIGHTS	DISTANCE	CARRIER_DELAY	WEATHER_DELAY	NAS_DELAY	SECURITY_DELAY	LATE_AIRCRAFT_DELAY
	1/7/16	AS	65	15841	35841	Wrangell, AK	14256	34256	Petersburg, AK	1046	-20	1106	-21	8	1	31	NaN	NaN	NaN	NaN	NaN

Table 7: Table for the shortest (Time) non-stop Domestic flight operated in all 50 states

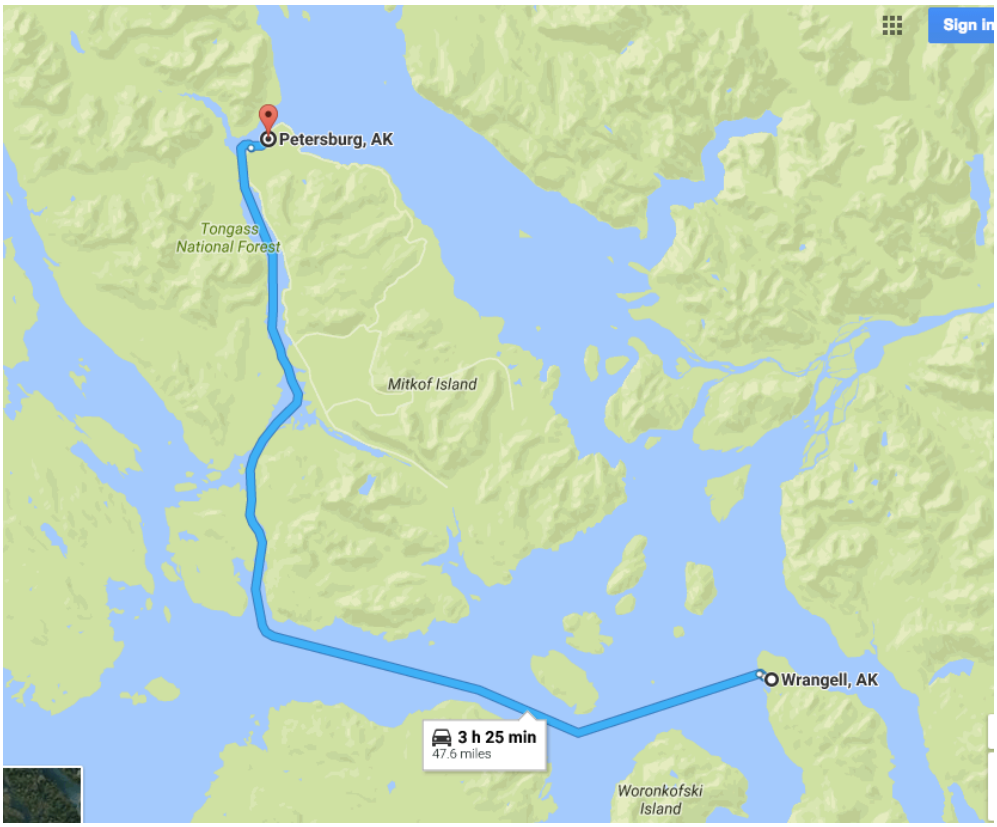

Figure 4: Shortest (Time) non-stop Domestic flight operated in all 50 states

The data we have gathered is rich and lets see if we can try to glean some insights from it. Out of curiosity we will answer the following questions:

1) For the flights experiencing largest arrival delays related to weather, what was the weather like at these airports/cities on the day of flight?

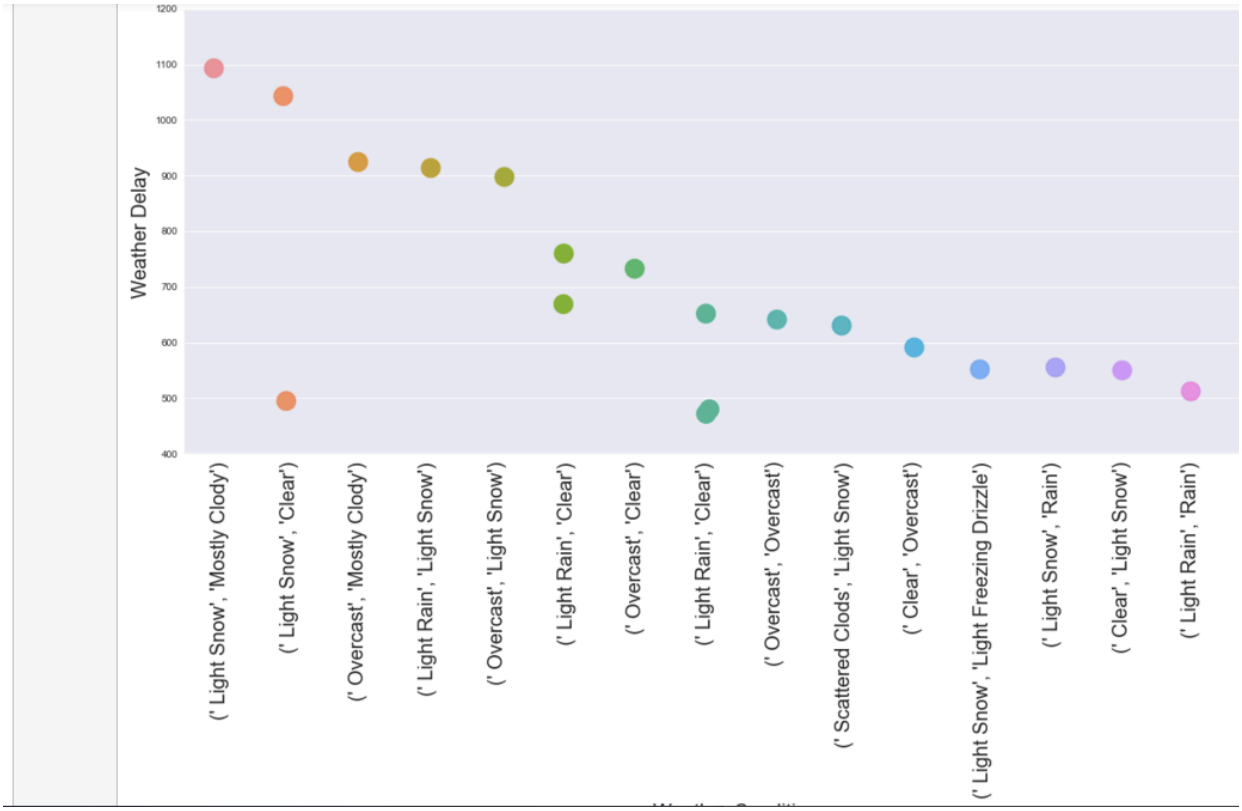

Figure 5: Weather conditions causing the largest Weather related delays

There is no discernible weather pattern that stands out which resulted in the delay. Clearly in the month of January we expect snow to be there on the ground and sometimes overcast skies. This seems to be the persistent theme for the delays in the system.

2) Which airports experienced these large weather related delays?

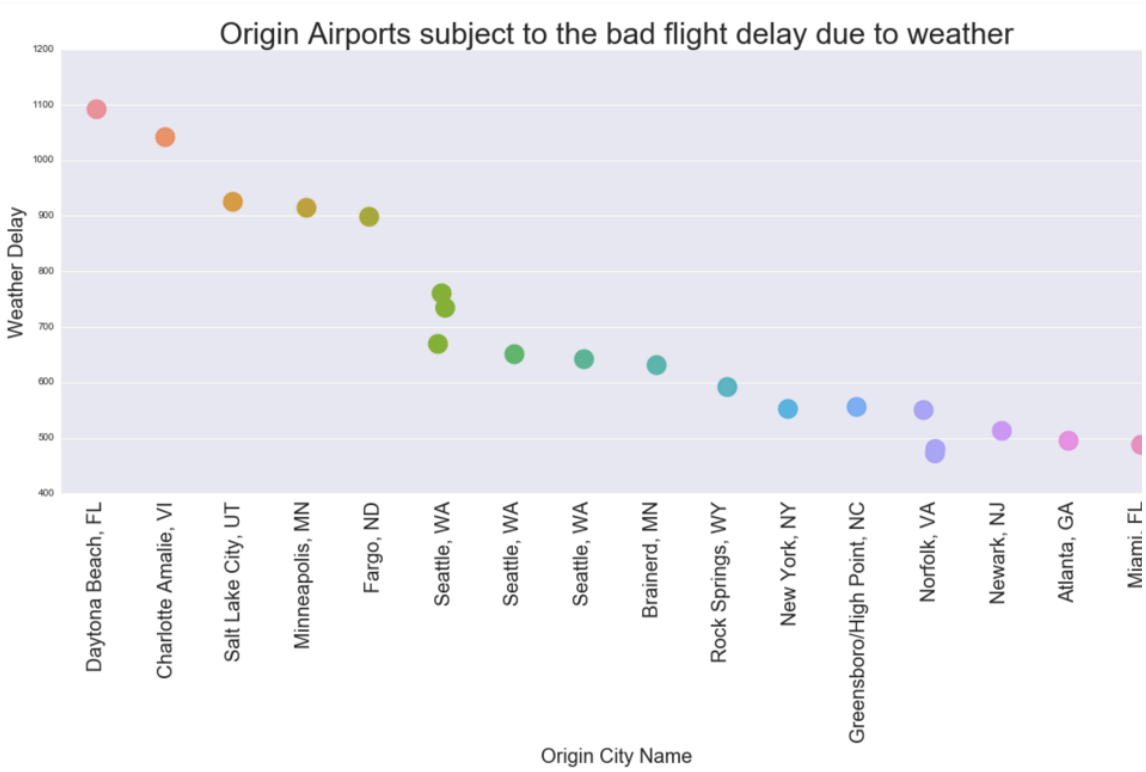

Figure 6: Origin cities experiencing the largest Weather related delays

Who would have thought Daytona Beach, FL would be the airport for the largest delay? It is very surprising. Further the previous figure shows that this delay was not related to any hurricane :) We must note that delays in the system are cascading since there are quick turnovers of the planes to get high utilization per seat. Sometimes airlines may use airports as temporary storage/parking till delays in other parts of the system are worked out. Whatever the case it is not clear from these figures the cause of these delays, since no single reason stands out that caused the large delays.

3) Finally we will plot out the airports which had the most flights for the month of January.

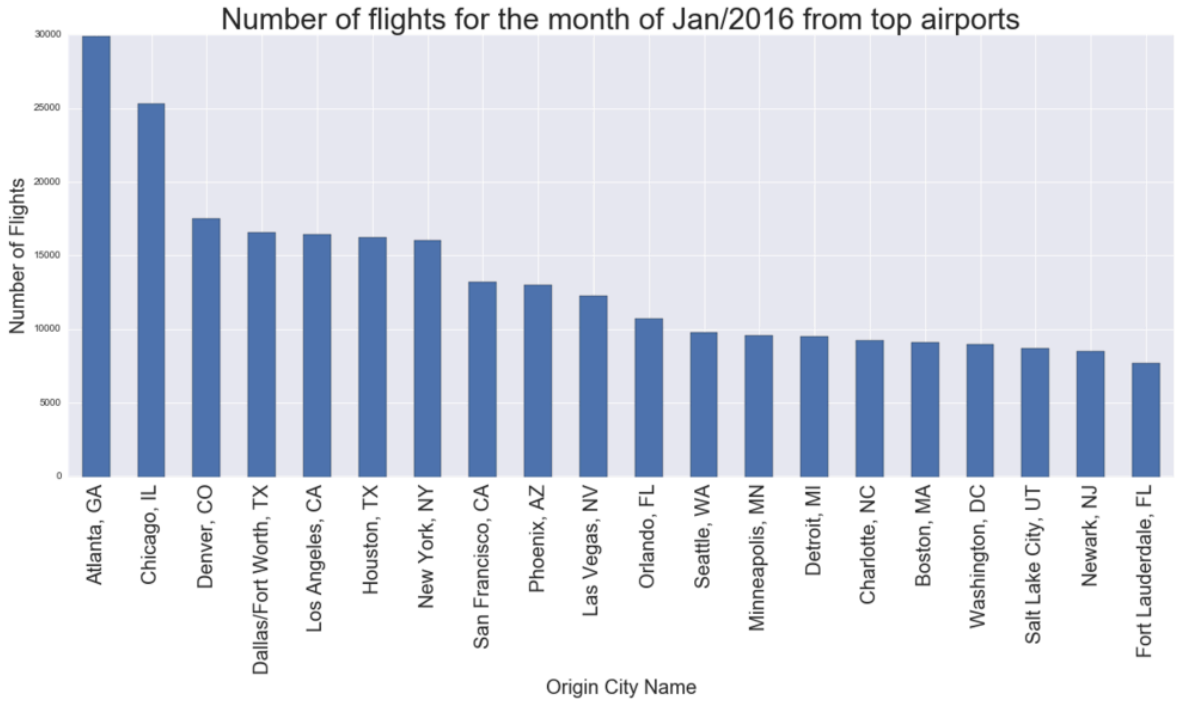

Figure 7: Number of flights departing from each top city for month of Jan/2016

Atlanta (Delta), Chicago (United) , Denver (United) and Dallas (American) seem to be the top busiest airports.

4) What airlines had the most number of flights in this month? What % of these flights were delayed by more than 180 minutes?

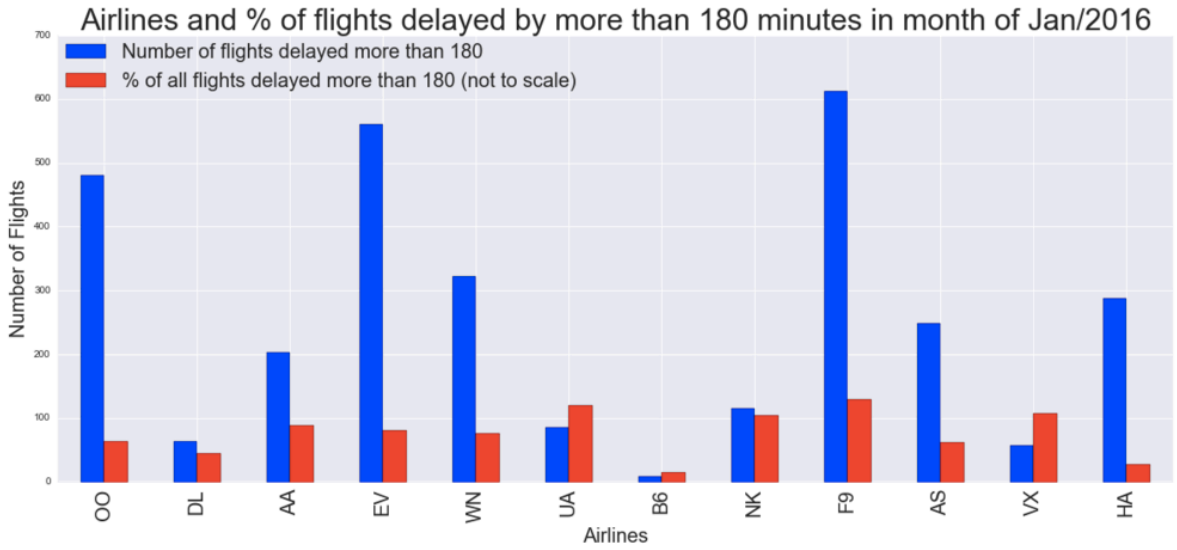

Figure 8: Number of flights and % of flights delayed by more than 180 mins plotted by airline

Frontier (F9) airlines seems to be the worst airline both in terms of the number of delays worse than 180 minutes (~ 4 hours) and the percentage of flights delayed more than 4 hours.

5) Which city pairs experienced the most delayed flights beyond 180 minutes?

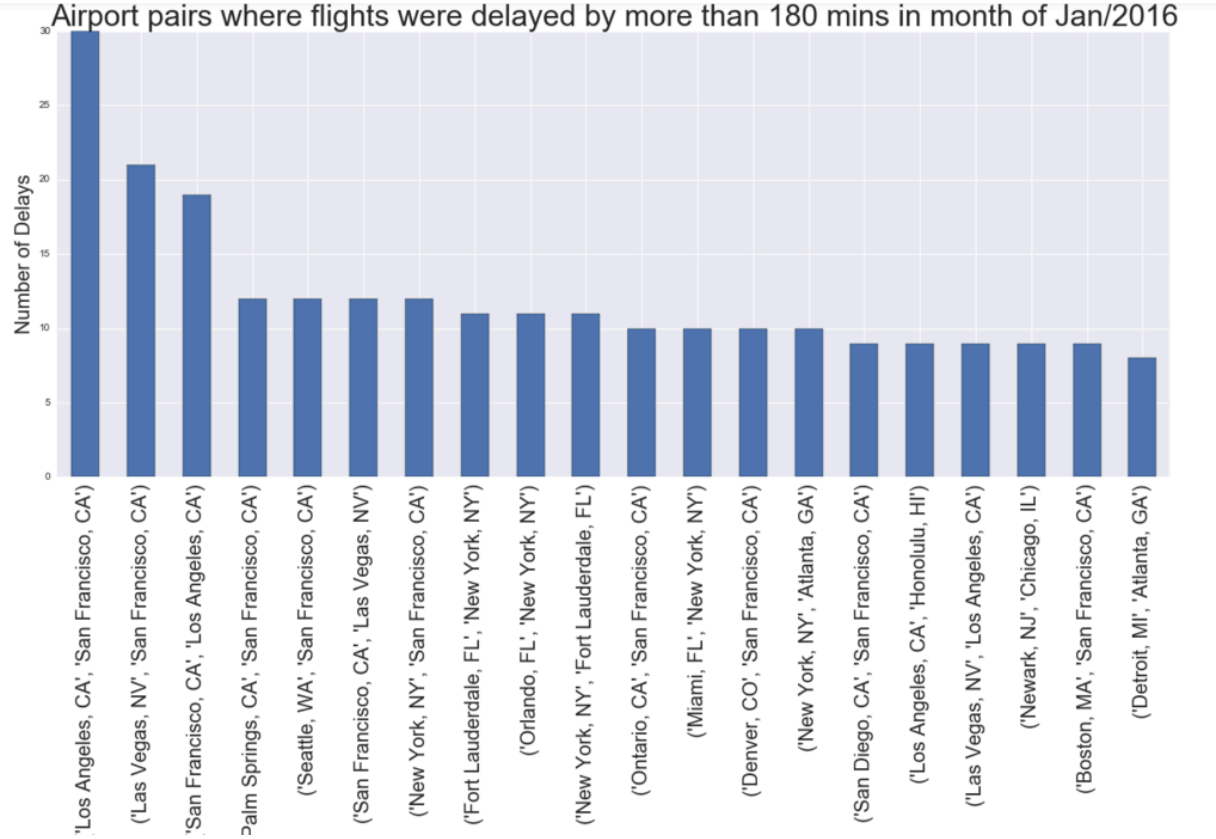

Figure 9: Top City pairs experiencing delays more than 180 mins

Most of the top delays seem to be between cities on the west coast. This is surprising because the winter should be very mild on the West Coast compared to the Midwest or North East. The data are full of surprises!

6) which date and day of the week are the delays predominant?

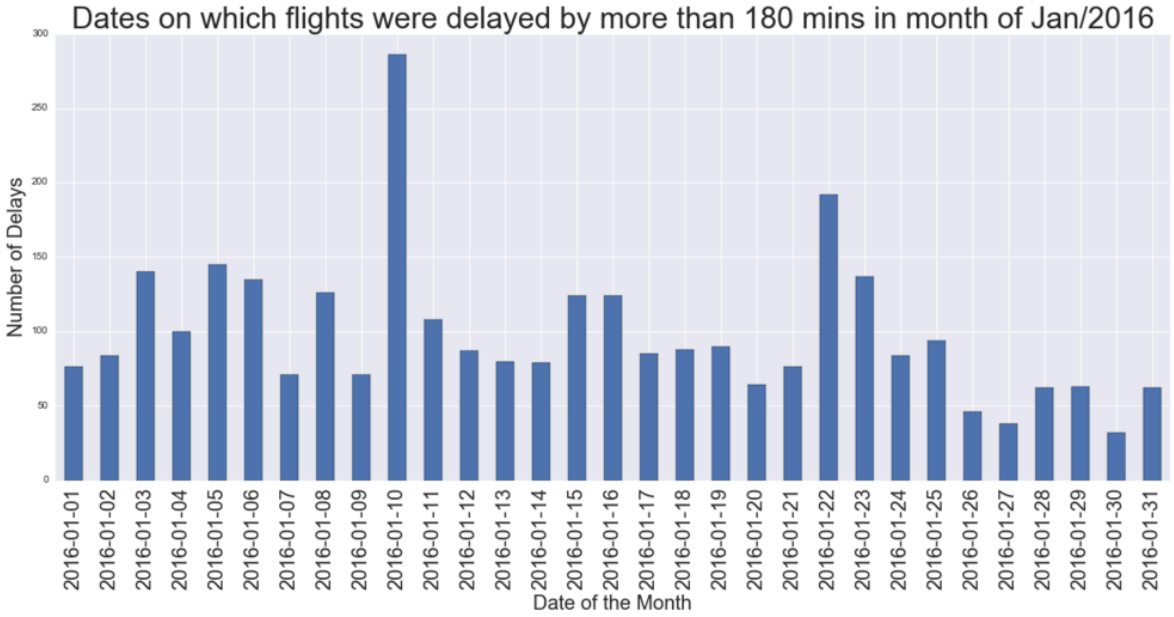

Figure 10: Dates in Jan/2016 experiencing delays more than 180 mins

There is a clear weekly pattern discernible in this data. Ignoring the large spikes on the 10th and the 22nd we see a wave that peaks every 7 days or so. It would be instructive to plot out the delays as a function of the day of the week.

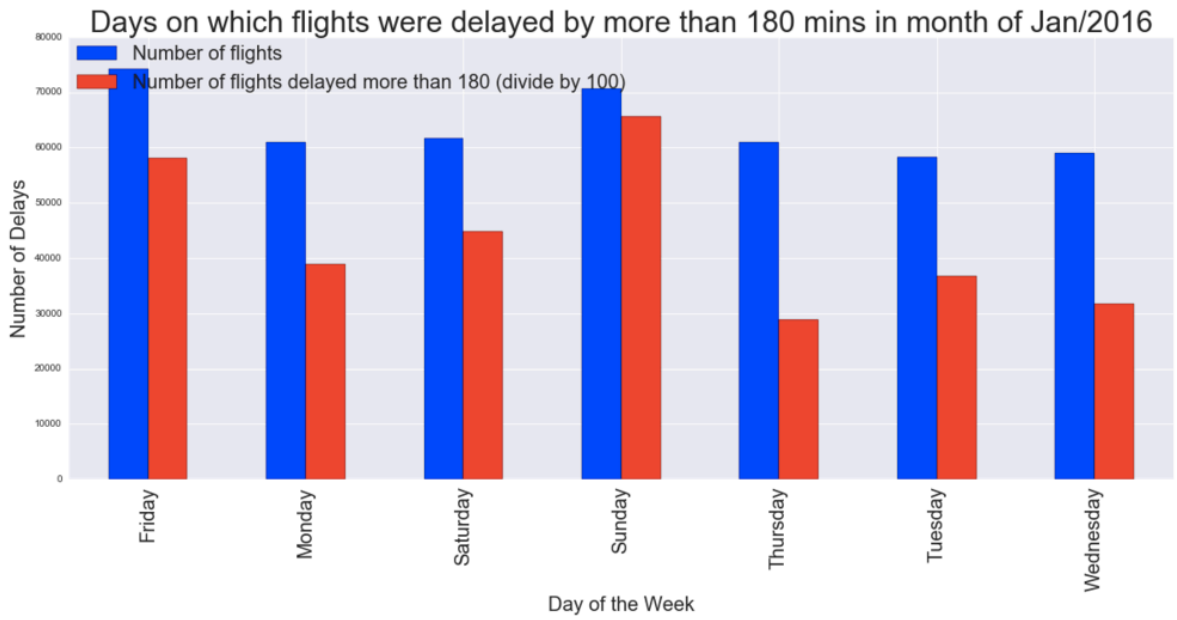

Figure 11: Days of the week in Jan/2016 experiencing delays more than 180 mins

We clearly see that there are spikes on Friday and Sunday for flights to be delayed by more than 180 minutes. This is not surprising. There are in general more flights on Fridays and Sundays. It is tempting to create another feature (DAY_OF_THE_WEEK) for the model to train on. There is significant information about variances for the flights for each specific day that this needs to be captured in the model. On the other hand the information in FL_DATE does not capture any information that we can use to identify a trend. For instance the large spike in flights on the 10th of January carried no information for a model to learn on (unless we had data that showed similar spikes in February and March etc, which then would be meaningful for a model to learn.) We thus drop the FL_DATE field in our analysis and change it to DAY_OF_THE_WEEK.

Note that our data does not have any weather related information for the days of the flights (much less for the days of the significant flight delays). In order to correlate our flight delay data with the weather we will have to query historical data from http://www.wunderground.com.

Also note that since querying the website is very expensive we try to limit the data we query and hence we restrict it to only flights that were delayed more than 180 minutes and this delay was mostly weather related. In order to save time we have already saved data once queried as a csv file and this file is read by default. The behavior can be controlled by a variable (callApi) by changing its value from False to True.

Algorithms and Techniques

We use multiple regressors and classifiers in our analysis to work on our data

Regressors:
regressor1 = DecisionTreeRegressor(random_state=0)
regressor2 = RandomForestRegressor(random_state=0, n_estimators=20)

Classifiers:
clf_A = DecisionTreeClassifier(random_state=0)
clf_B = KNeighborsClassifier()
clf_C = GaussianNB()
clf_D = RandomForestClassifier(random_state=0, n_estimators=20)

In the paragraphs we give an explanation of how each regressor/classifier works. The difference between a classifier and regressor for a model is what quantity the model maximizes at each node. For a classifier it is information gain and for a regressor it is reduction in variance. Also a regressor assumes piecewise linear relationship at each of its leaf node and if the prediction input falls between two training features then prediction is treated as piecewise linear function and interpolated value is calculated from the predictions of the two closest features.

DecisionTree : Decision tree models can be used in most classification tasks and are very versatile and work on most discontinuous datasets. In fact this is exactly how humans think in making a decision and hence they are intuitive and simple. The trees themselves are built using the Classification and Regression Tree (CART) algorithm, where at each node a "best" feature is selected that best explains the output data. Decision trees are prone to over-fitting and if not constrained (either by depth size or information gain) they can produce complicated classifiers that may be hard to generalize. Decision trees are built at each level based on the information gain (i.e reduction in entropy or uncertainty). The information gain at the root node is the highest and reduces the deeper we build our tree (this is true because at each node we choose the "best" attribute.) We should stop building the tree at any node if the information gain doesn't meet an imposed threshold. Because the "best attribute" is used at each node, the most relevant attributes automatically filter up towards the root node.

KNeighborsClassifier : "Birds of a feather, flock together", this pretty much sums up how K-nearest neighbors algorithm works. In order to predict how late a flight will arrive, we look at other flights with similar characteristics and see what the outcome was for these flights. More concretely, we need a corpus of data for past flights and their descriptive attributes and the outcome (i.e if the flight arrived early or late.) Once we have these data we can pick a new flight for which we also have data on these attributes and try to predict if our flight will be early or late. We do this by finding a group of flights that are most similar to our flight. For this group of flights we find the majority of the outcomes (whether they arrived late or early) and our algorithm says that our flight will probably have this outcome. For example "*if the majority of flights between San Francisco and Los Angeles on Friday are late then our flight with similar characteristics is going to be late*".

GaussianNB: Naive Bayes Classifier is a very simple classifier based on Bayes rules that can be trained a priori and does very well compared to more complicated classifiers achieving almost comparable scores. The main assumption for Naive Bayes Classifier is that the input features are all independent of each other. This is usually not true but the features are usually very weakly correlated and hence to a first approximation can be considered independent. In these cases the Naive Bayes works very well. However in cases where features are very strongly dependent Naive Bayes may not work well as its basic assumption is violated. In such cases it may make

sense to do Principal Component Analysis on the input features before running Naive Bayes Classifier on it. Naive Bayes is simple, fast, easy to understand and very robust under most use cases. It will not work well where the assumption of independence on input features is violated. It has found uses in applications like spam filtering.

RandomForest: RandomForest is similar to a DecisionTree but it is an ensemble of many trees. Moreover each tree is grown with added randomness. Here is how the algorithm is described in the Scikit-Learn package: *"Each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model."*

We chose these regressors/classifiers because they are intuitive, simple and easy to use. They can be used directly from the Scikit-Learn package and they are easy to explain and easy to understand even for a layperson. Decision tree and K-nearest neighbors are exactly how humans think and make choices in real life. Naive Bayes is a little harder to explain to a layperson but the algorithm does make intuitive sense. Finally in my limited machine learning experience RandomForest is one of the most powerful classifier/regressor there are and almost always ends up being the one with the most predictive power. It is easy to understand once the basic idea behind ensemble classifiers is understood and it can be used directly "out of the box" without having to build a model from scratch as required with Neural Networks. I think it would be foolish not to include RandomForest in any classification (especially if you are using DecisionTree)

Note that all of these classifiers and regressors are non-linear i.e they can work on a feature set even if there is no linear relationship between a feature and the predicted variable. The predicted variable is a continuous variable and hence a regressor is the natural choice for a machine learning model for this problem. Before we start any model training we define and quickly find R^2 between some features and the predicted. We find a strong linear relationship between the DEP_DELAY and ARR_DELAY:

```
Model has a coefficient of determination, R^2, of 0.859.
```

No other feature shows a strong linear relationship with the predicted. A scatter plot shows this visually:

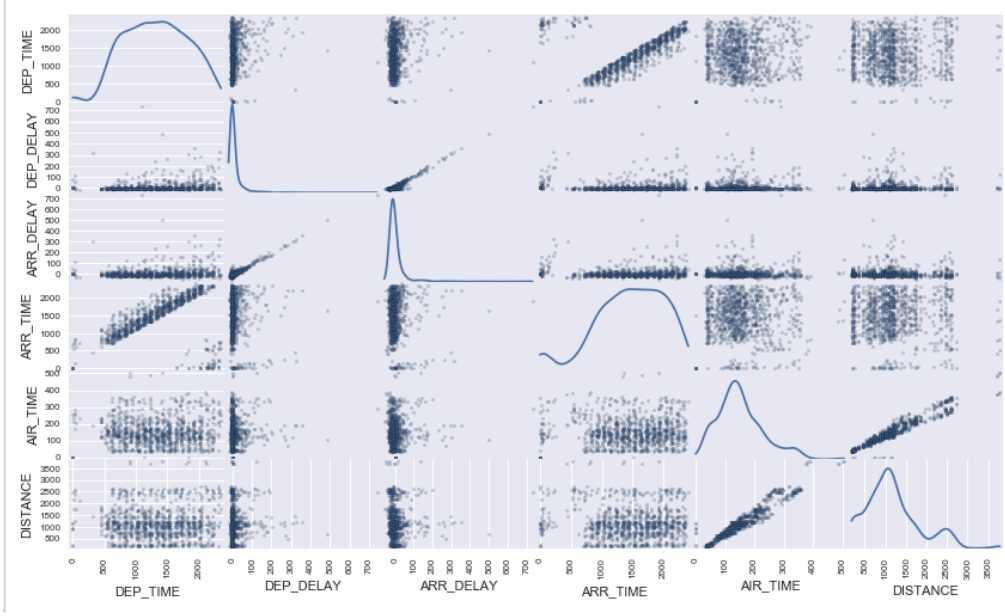

Figure 12: Scatter plot between some selected features

Now we proceed to write an train a model. First we need to preprocess the feature set. For reasons that are explained in the I**mplementation and Refinement** section we hope to be able to drop DEP_DELAY from our model. Unfortunately for us the regressor is showing a large R^2 and a small RMSE for only the dataset that includes DEP_DELAY. On all the other datasets where DEP_DELAY is dropped we are not getting good results for the regressors. As a result we had to experiment with the classifiers as well. The predicted variable is a continuous variable. Even if it is broken down into ranges these ranges turn the predicted into an ordinal variable. Unless the ranges are equal we will be losing classification accuracy since the classifier does not know which range comes after which on a number scale. This loss in classification accuracy is in addition to the loss in precision by moving from a continuous scale to a more granular range. As a result we should prefer a regressor over a classifier for the problem at hand. However as we discuss in the conclusion of the project it is not possible to build a single model to answer how much delay a flight will have in advance. It is only possible to predict whether a flight will be early, late or very late in advance.

We run the regression and classification on two datasets. The first dataset is a reduced dataset including only the numerical continuous features. The second dataset is the full feature dataset including the categorical features that we mapped to numerical to allow for scikit-learn algorithms to run properly (more on this in the **Data Preprocessing** section**)**

Benchmark

We aspire our models on average to be able to predict the arrival delays accurately when given the departure delay. However it is not possible to predict every data point accurately and we expect our model to make mistakes on some data points (sometimes spectacularly.) But on average we expect the model to be accurate and when the model does deviate it does so only a small number of minutes.

Thus we expect our model on average to predict arrival delays close to the actual arrival delays (i.e average difference between predicted and actual values to be close to 0) and on average deviate very slightly from the actual value (standard deviation of difference between predicted and actual values is very "small")

Data Preprocessing

We carry out the following data preprocessing (in some cases upfront for all the models and sometimes specifically just before a model is used)

1) In our analysis portion of the data we clearly saw that there are spikes on Friday and Sunday for flights that are delayed by more than 180 minutes. This is not surprising. In general there are more flights on Fridays and Sundays (probably the weekend warriors hitting the road.) It is tempting to create another feature (DAY_OF_THE_WEEK) for the model to train on. There is significant information in the variances between flights on each specific day. And this needs to be captured in the model. On the other hand the information in FL_DATE does not capture any information that we can use to identify a trend. For instance the large spike in flights on the 10th of January carries no information for a model to learn on (unless we had data that showed similar spikes on 10th in February and March etc, which then would be meaningful for a model to learn.) We thus drop the FL_DATE field in our analysis and change it to DAY_OF_THE_WEEK. We also need to map the DAY_OF_THE_WEEK from (Monday through Sunday) to (0 through 6.) This is a requirement for the regressors/classifiers in the Scikit-Learn package. More on that in the next point…….

2) Scikit-Learn classification does not allow for categorical variables. This seems like a severe limitation of Python Scikit-Learn. In any case we either use mapping (recommended) or dummies (very expensive) to change categorical variables to numerical to be allow them to be used in regression/classification.

3) Finally the predicted variable cannot be continuous for classification. Therefore we break it into ranges for classification problem. Classification on a continuous numerical variable (as is in our case for ARR_DELAY) is hard because it is hard to get the correct values of the ranges and the classifier cannot understand the ordinal nature of the data series once a continuous numerical variable is broken into ranges. This is why we try to limit the number of classes for the classification to a small number.

4) There is no protection for domestic air travelers in the USA. Airlines are not required to compensate passengers for delayed flights. However recently US DOT has required airlines to compensate passengers after three hours of tarmac delay. It would be intuitive to think that an airline that needs to start compensating passengers and let them deplane after three hours will probably end up canceling the flight at that point. So in our analysis we use a breakpoint around 180 minutes (3 hours) to add finer granularity in our analysis.

Implementation and Refinement

1) Having `DEP_DELAY` in the regression/classification feature set is a problem. `DEP_DELAY` is not known until the flight has departed. Thus any model that relies on `DEP_DELAY` to make predictions is useless by being dependent on a variable that cannot be known ahead of time.

Thus in some of the models that we implement we drop `DEP_DELAY` from our model. We also drop the following columns 'CARRIER_DELAY', 'WEATHER_DELAY', 'NAS_DELAY', 'SECURITY_DELAY', 'LATE_AIRCRAFT_DELAY' when we run the model on the full feature set because these delays are strongly related to `DEP_DELAY`.

2) We had to refine the ranges of the predicted variable in the classification to make the algorithm result more realistic. Because of the ordinal nature of the data and large range of the predicted variable the ranges that python comes up with automatically are so broad that any weak algorithm would also work fine. We refined the delay ranges with more precision and broke up the delays as

[-79, 0), = "early"
[0, 60), = "late",
[60 180), = "very late"
[180 MAX), = "are you kidding?"

Model Evaluation and Validation

The following are the final prediction scores for our models.

Regression:

Regressor	Reduced Feature Set	Reduced Feature Set (Drop `DEP_DELAY`)	Full Feature Set
DecisionTreeRegressor	R^2 : 0.722172868791	R^2 : -0.365622958218	R^2 : -0.198564848978
	RMSE : 200.169645277	RMSE : 988.731902471	RMSE : 1876.11794013
RandomForestRegressor	R^2 : 0.852078364297	R^2 : 0.273219609667	R^2 : 0.32267806514
	RMSE : 106.574981423	RMSE : 526.200115257	RMSE : 1060.21450097

Table 8: R^2 and RMSE for two different kinds of regressors on three different kinds of datasets

Classification:

Classifier	Reduced Feature Set	Full Feature Set
DecisionTreeClassifier	F1 score for training set: 0.9070	F1 score for training set: 1.0000
	F1 score for test set: 0.6640	F1 score for test set: 0.7870
KNeighborsClassifier	F1 score for training set: 0.7424	F1 score for training set: 0.8542
	F1 score for test set: 0.6502	F1 score for test set: 0.7827
GaussianNB	F1 score for training set: 0.5634	F1 score for training set: 0.5701
	F1 score for test set: 0.5619	F1 score for test set: 0.5706
RandomForestClassifier	F1 score for training set: 0.9036.	F1 score for training set: 0.9977
	F1 score for test set: 0.6675.	F1 score for test set: 0.7941

Table 9: F-score for four different kinds of classifiers on two different kinds of datasets

Clearly DEP_DELAY is needed for the regression to work. It is not possible to get a precise value of ARR_DELAY without knowing the value of DEP_DELAY.

Justification and Refinement

From our analysis we see that we need to make a choice between two options neither of which are particularly appealing. We get good regression results when `DEP_DELAY` is included in the REDUCED feature set. As we had noted in our pre-experiment work, `DEP_DELAY` was the best predictor of `ARR_DELAY` and removing it sinks the regression results. We also see that RandomForestClassifier on a full feature set performs reasonably well but the predicted variable `ARR_DELAY` is continuous and hence breaking it into categories results in loss of precision.

In conclusion we decide to go with an ensemble of two different models as the final choice and use the right model depending on what is asked:

1) It is possible to make a prediction for Arrival delay but only after the flight has departed and we have a value of `DEP_DELAY`. It is not possible to make a precise prediction for the flight delay before the flight has left. There is not enough predictive power in the data to make such a confident conclusion. We use a `RandomForestRegressor` on a REDUCED feature set but including the `DEP_DELAY` to make a prediction of the `ARR_DELAY`

2) It is possible to make broad predictions regarding arrival delays flights on a full feature data set ahead of time (i.e without knowing `DEP_DELAY`.) Thus we can know reasonably well if a flight will arrive early (negative arrival delay), be late (arrival delay between 0 and 60 mins), be very late (arrival delay between 60 and 180 mins) and might as well be canceled (arrival delays more than 3 hours). For this we use `RandomForestClassifier` on a full feature set (but without the `DEP_DELAY`)

So to conclude we can predict with good accuracy the arrival delay after a flight has left. But in advance we can only predict in a broad range whether a given flight will be early or late.

Refining the Regression model using GridSearch

Note: Running grid search is very expensive computationally. Our laptop stopped responding for a large grid. We had to run grid search piecemeal to find out the best options for max_depth and n_estimators. From the table we see a big jump in R^2 going from 20 to 25 for both the parameters. The change from 25 to 30 is fairly small (slowing rate of improvement; plateauing effect.) The RMSE does show an increase of less than 5%. **We choose 30 as the optimal parameter for n_estimators and max_depth.**

	R ^2	RMSE
n_estimators=20, max_depth=none	0.852078364297	106.574981423
n_estimators=25, max_depth=25	0.916863178143	114.631639809
n_estimators=30, max_depth=30	0.919151954372	111.475803846

n_estimators=30, max_depth=25	0.917351018429	113.958990426

We see that the model makes reasonably good predictions when DEP_DELAY is known. Thus we can use the model to find out how late a flight will arrive once it has departed.

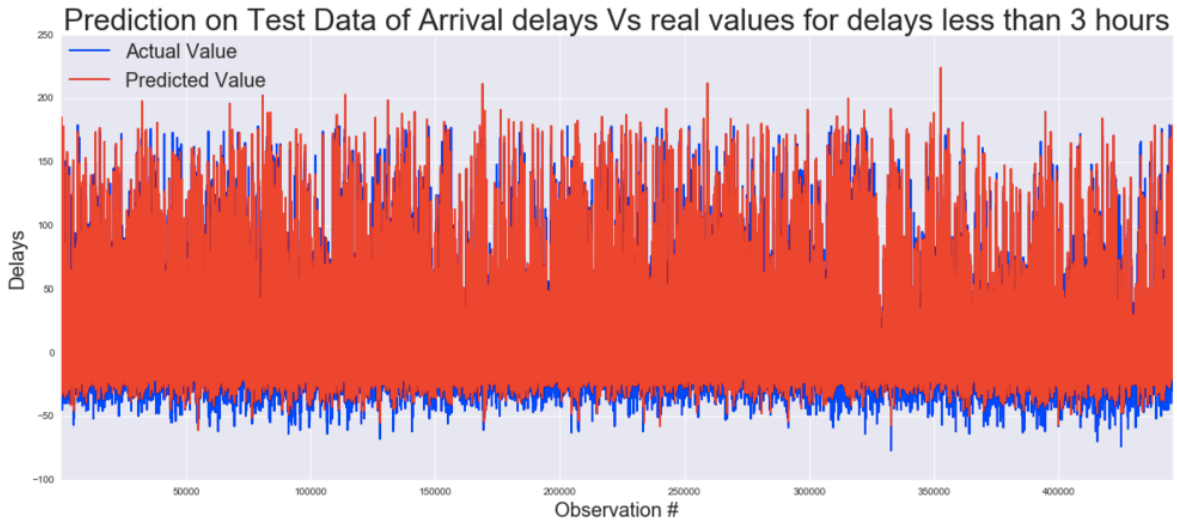

Figure 13: Comparing predicted and actual values of the output variable from a regressor (for values less than 180 minutes ……..)

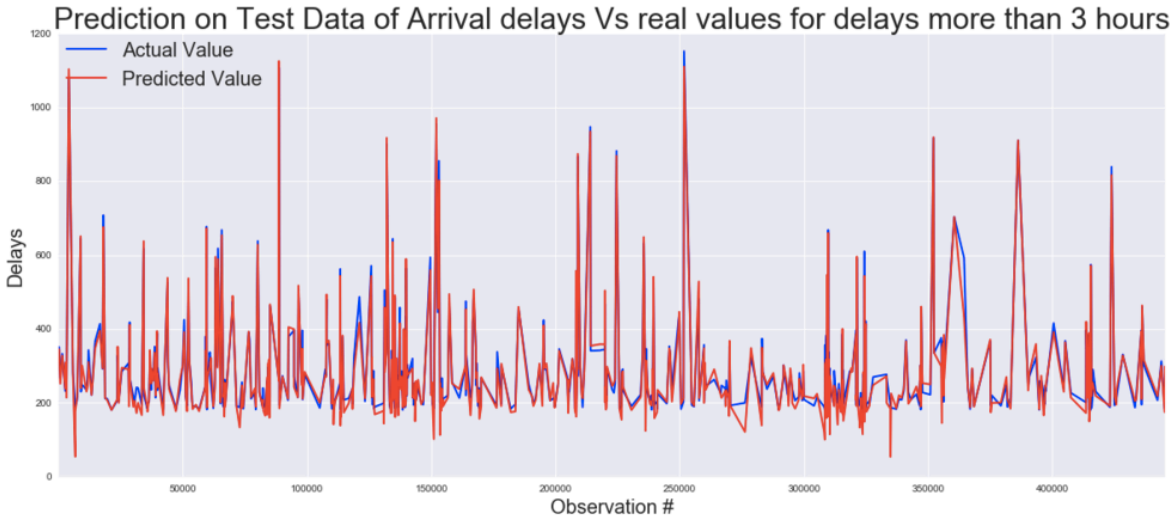

Figure 14: Comparing predicted and actual values of the output variable from a regressor (……. and values more than 180 minutes)

Comparing our predicted arrival delay vs actual arrival delay below we see that on average the model gets it right. The mean of the difference is `-0.111320` which is very close to 0. Further the standard deviation of the difference is 10 minutes. We did not define this benchmark earlier

but 10 minutes seem a reasonably "small" deviation from the mean. Thus on average our model will predict accurately and if it makes a mistake it will usually only be wrong by about 10 minutes on either side (i.e there is no skew while making mistakes)

Comparison to Benchmark

```
Predicted  - Actual Test:
count    89166.000000
mean        -0.111320
std         10.557683
min       -159.800000
25%         -4.670676
50%          0.312831
75%          5.592587
max         57.100000
Name: ARR_DELAY, dtype: float64
```

Table 11: Descriptive Statistics of difference between predicted and actual values of the output variable

However when the model does fail, sometimes it fails spectacularly. We see that the minimum difference and maximum difference is `-159.800000 and 57.100000` but thankfully these are few and hence the standard deviation stays small.

Refining the Classification model using GridSearch

	F-Score (Train)	F-Score (Test)
n_estimators=20, max_depth=none	0.9977	0.7941
n_estimators=25, max_depth=25	0.9591	0.7695
n_estimators=30, max_depth=30	0.9952	0.7873
n_estimators=30, max_depth=25	0.9607	0.7714
n_estimators=20, max_depth=40	0.9977	0.7922

For classification we see that the best values are at (**n_estimators=20, max_depth=40**) however we decide to prune the tree and increase the number of estimators at a slight loss of F-score. This has two advantages. The classification forest matches the regression forest and we reduce the variance at the expense of slight increase in bias.

Using this optimized classifier we can also make broad predictions about a flight in advance. These are the predictions on 50 data points that were not used for training or testing of the model. These 50 data points were held out before the rest of the data was made available to the model. But for a few errors the predictions are fairly accurate.

```
          Actual  Predicted
305916      late       late
72918       late      early
146471     early      early
117426     early      early
16314      early      early
394372      late      early
23962       late  very late
```

65533	late	early
274375	late	late
168402	late	late
46218	late	late
393148	early	early
142764	late	late
70156	early	early
87109	early	early
57638	late	late
239737	very late	late
404376	early	early
432850	early	early
129480	early	early
425255	early	early
251919	early	early
73494	late	early
94448	early	early
31628	early	early
111336	late	late
90677	early	early
215432	early	early
239571	early	early
433784	early	early
389099	early	early
392806	late	early
403995	early	early
404374	late	early
342127	early	early
364729	early	early
138222	early	early
158539	late	early
241728	late	late
250478	early	early
337105	early	early
418021	late	late
157228	late	early
183148	late	late
206393	late	late
294955	late	late
281963	early	early
138709	late	late
82284	early	early
201306	early	early

Table 10: Comparing predicted and actual values of the output variable from a classifier

Thus we can get an idea if a flight will be late or early in advance but the exact value of the delay can only be known after the flight has departed.

In conclusion:

We created an ensemble of models containing two models. Using this ensemble we can predict the delays for any flight. The following two questions can be answered using this ensemble.

1) What is the likely arrival delay `ARR_DELAY` at the destination after my flight has departed `DEP_DELAY` from the origin?

```
    Use the `RandomForestRegressor` model on the reduced feature set
with `DEP_DELAY` as a feature. The model has max_depth=30 and
n_estimators=30.
```

2) Will my flight in the future likely to arrive early, late ot very late at the destination?

```
   Use the `RandomForestClassifier` model in the full feature set
without the `DEP_DELAY` as a feature. The model has max_depth=30 and
n_estimators=30.
```

Reflection

We started this project with an open mind and no pre-conceived notions. This truly was

completely new data and new ground for us. We in fact wanted to do a model about optimal solutions for choosing between frequent filer miles, and cash to pay for various flight routes in an airline's itinerary. However not being able to find data for this we settled on this project. We went through multiple iterations selecting which data to download to be able to a predictive model. Most of the data on the original website is redundant and it took us a few tries before finally deciding on the data to download.
Being able to predict flight delays even before you set your foot in the plane would be so cool and this was the final working model that we tried to work towards. We struggled through normal learning pains to build the first regression model (which are detailed in the Implementation section.) Once over the learning pains, it was easy to build a regression model (using dates, delays, airport ids etc) and the model was working very well giving good results.
However we were given the first informal feedback regarding the model that we had built and we were asked to remove the departure delay as well as the flight dates to make the model generalized. This made sense! Departure delay cannot be known until after the flight has departed and a model that has learnt for a specific date cannot be useful for flights on a future date. However it took us a few tries before finally coming up with this final feature set (switching flight date to day of the week feature.)
Dropping the departure delay results in a large increase in bias. It is not possible to create a regression without knowing the departure delay. Because of the bad results we were getting in the newer regression models we had to explore classification models as well. Using classification models on a continuous variable is frowned upon and is fraught with peril (as we have talked about previously in the report.) Our first attempt at classification produced great results but pretty soon we figured that the output classes were so unbalanced that the model was pretty useless. We needed a model with finer precision but that would have made the issues with ordinality of the output variable worse. We finally settled on a model with broad ranges but the range boundaries at better cut off points to make the model more useful.
Finally we reached a compromise and to our conclusion that we needed to use an ensemble of two models (one regression based on departure delay and one classification not based on departure delay but based on board ranges for the output variable) for our project to the useful. From out model **we can get an idea if a flight will be late or early in advance but the exact value of the delay can only be known after the flight has departed.** Which model from the ensemble is used will depend on what question is asked of us.
Along the way we understood the richness of the data we had and decided we would answer some questions through interesting visuals to satisfy our curiosity.

Finally we highlight a few errors that we made while pursuing this project:

1) The project went though multiple iterations. We were getting very bad results for our regressors and then we changed/moved some of the cells around and started getting good results. It was surprising but we looked and didn't see any issues. We were getting R^2 values of 0.9999 and RMSE of less than 1. Our project appraisers were rightfully skeptical about the excellent results we were getting and yet we found no issues. Until we found the problem…….

In our code we had
features_set1.drop(['ARR_DELAY'], axis =1)
instead of
features_set1 = features_set1.drop(['ARR_DELAY'], axis =1)

The idea was correct but the implementation was flawed. We had forgotten that the .drop method

does not work in-place. As a result the feature was never getting dropped. As luck would have it this was the predicted variable. Thus the predicted variable was in the feature set. No wonder we were getting excellent fit. This was a special case of collinearity which is a high correlation between features and the predicted variable (in our case this was 1.) The model was able to do such a great job of predictions because it was being asked to predict what it had already been given.

2) One other mistake we made was that because we were reusing variable names (in particular X_train, X_test, y_train, y_test which are the outputs of train_test_split) we copied some of the commands for data splitting. But we forgot to change the dataset on which to split. In effect for regressors on full feature set we were actually splitting on dataset for reduced feature set without DEP_DELAY (i.e the wrong dataset.) (Un)Fortunately for us the changes in the first point above made both the regressors results so bad that the second point didn't matter (dropping DEP_DELAY from any feature set screws up the regression results.)

In summary we made some mistakes which made our model unrealistic. Correcting the mistakes makes the model more realistic.

Improvement (PCA)

We can run PCA on the full feature set to see if we can reduce the number of dimensions to define the output variance without losing much of the defintion of output variance. Running PCA we see that only the first three dimensions explain 90% of the variance in the output. We also see that just three features predominate the first three dimensions. These three feature are: `'FL_NUM', 'ORIGIN_CITY_MARKET_ID', 'DEST_CITY_MARKET_ID'`. Thus our PCA tells us that all we need to know is the flight number, the origin city and the destination city to find out all the variance in the data. Yet if we run a classifier with just these three features (`'FL_NUM', 'ORIGIN_CITY_MARKET_ID', 'DEST_CITY_MARKET_ID'`) and try to predict the `ARR_DELAY` it does poorly. PCA is valuable on this data but needs to be thought through more carefully. At a minimum we know that `DISTANCE` and `AIR_TIME` are correlated, `ARR_TIME` and `DEP_TME` are correlated and hence PCA should be helpful in reducing at least a few features.

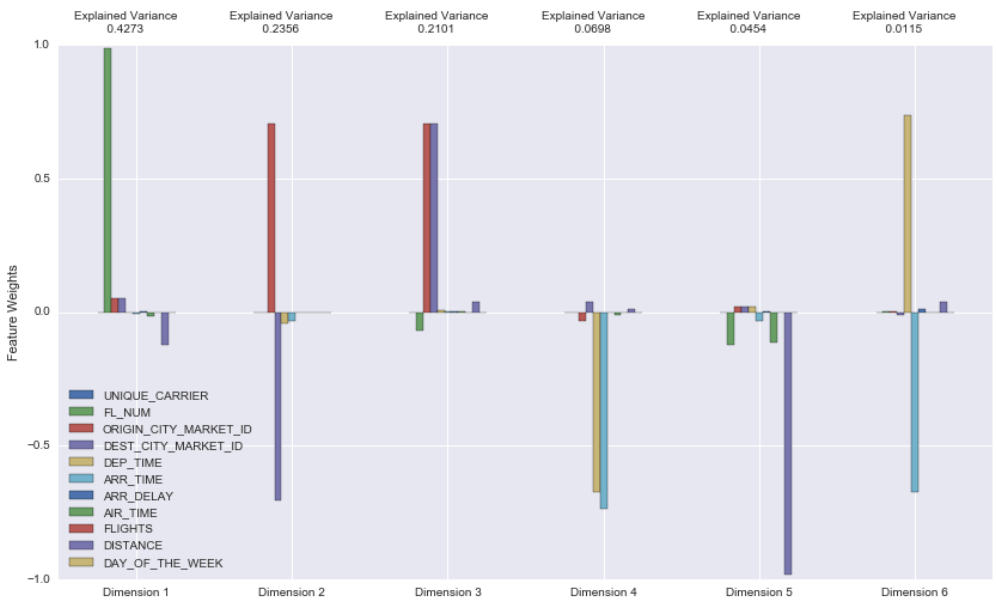

Figure 15: PCA results on our DataSet

Improvement (More data)

That regression results returned such low R^2 scores for the training data just means that we need more data. Yes we have a lot of data already (more than 400k datapoints) but there are also a lot of features (and more importantly a lot of classes within each of the features.) The predicted variable is a continuous variable and hence can take infinite values. The regressor relies on piecemeal linear regression to find predictions close to actual values. When the data are all over the place the regressors cannot rely on these piecemeal linear regressions to find a good value (i.e. these piecemeal linear regressions have a broad range for the regressors to be able to predict properly.)

As an example if we treat all the data above 180 minutes (3 hours) arrival delay as an outlier (i.e. these are not repeatable events which can be learnt but these happen due to events that cannot be predicted or learnt then the regression on a full feature set improves significantly.

Quality of the regressor when data points for ARR_DELAY <= ∞ are included in the regression:

```
'Regression Score for regression prediction using a RandomForestRegressor with DEP_DELAY NOT
part of the FULL feature set'

R^2 score on training : 0.89251192581
R^2 score on test     : 0.32267806514
RMSE score on test    : 1060.21450097
```

Quality of the regressor when data points for ARR_DELAY <= 180 are included in the regression:

```
'Regression Score for regression prediction using a RandomForestRegressor with DEP_DELAY NOT
part of the FULL feature set'

R^2 score on training : 0.919793781763
R^2 score on test     : 0.493304087186
RMSE score on test    : 367.381159175
```

Quality of the regressor when data points for ARR_DELAY <= 120 are included in the regression:

```
'Regression Score for regression prediction using a RandomForestRegressor with DEP_DELAY NOT part of the FULL feature set'

R^2 score on training : 0.926149381193
R^2 score on test     : 0.544959980224
RMSE score on test    : 238.055754075
```

Quality of the regressor when data points for ARR_DELAY <= 60 are included in the regression:

```
'Regression Score for regression prediction using a RandomForestRegressor with DEP_DELAY NOT
part of the FULL feature set'

R^2 score on training : 0.937146280099
R^2 score on test     : 0.614863897604
RMSE score on test    : 114.142733094
```

Quality of the regressor when data points for -30 <= ARR_DELAY <= 30 are included in the regression:

```
'Regression Score for regression prediction using a RandomForestRegressor with DEP_DELAY NOT
part of the FULL feature set'

R^2 score on training : 0.939240047835
R^2 score on test     : 0.627359511647
RMSE score on test    : 58.9496460641
```

We see that as the range of the output variable is tightened the R^2 for the model increases. What does this mean? If we give guarantees of prediction over smaller ranges for the output, the better confidence we have for those prediction (this is statistics 101.) In real terms what this means is:

1) We can build a model with the current data and make good predictions but over smaller range for the output variable or,

2) We need more data if we need to be able to predict over the current range.

There is no upper bound for a flight (or for that matter anything) to be delayed. And hence it is still a good idea to put an upper bound for the flights delay in our analysis (we alluded to 3 hours delay being a good cut off point for analysis previously)

In either case the current model is STILL not good enough. We still need more data for our regressors to learn the patterns well but our discussion shows it is possible to build a regression model with good predictive pose that does not rely on departure delays by controlling the ranges of the output variable and getting more data.

Another way to think about this: By adding DEP_DELAY back the regressor performs very well, by removing DEP_DELAY the regressor does poorly. This means flights don't have a pattern of arrival delays (or at least not to a fine enough resolution for it to become a consistent pattern.) Assuming each flight (the tuple of flight number, carrier, origin and destination city) occurs only once per day we have at most 31 datapoints for each flight. More data over more months will help the model learn better/more about each flight.

Improvement (Better Modeling)
The same planes end up operating multiple flights. A plane leaving from SF and going to LA turns around in an hour and may next fly to Las Vegas from where it may fly to Austin before flying back to SF all in the same day. The plane is the same, the flight numbers change for each leg of the journey. Delays caused in one leg of the journey may result in cascading delays in other legs of the journey. Our model assumes that each data point and each flight is independent. This is not true. The dependence we highlighted above is lost in our model and carries a lot of information. Arrival delay of one flight is likely the Departure delay on another flight. This dependence will be hard to model but will likely improve the model a lot once correctly accounted for.