**Fly on-time by Booking Smart**

As a traveler, I have always wondered if I could have avoided wasting time at the airport due to delays by booking my flight differently. Flight delays are inevitable and we can all blame the weather for it but there may be trends which can help us book better to possibly avoid delays. Certain months or days may be prone to delays. Also there may be certain airports or airlines which are more prone to delays. Knowing where delays often happen may be the key to predicting delays in your itinerary and book better.

As a travel agency, I may be able to utilize a delay prediction system to warn my clients about potential delays, especially when the client is travelling with a child or a senior citizen who will be affected the most by delays.

As an airline, I can take steps to identify why my airline has more delays and plan to avoid them.

There is data about airlines’ performance available on the [website](http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time) of RITA (Research and Innovative Technology Administration) the coordinator for the U.S. Department of Transportation research programs.

The data retrieved from the site would have below fields:

|  |  |
| --- | --- |
| Field | Datatype |
| Year | int64 |
| Month | int64 |
| DayofMonth | int64 |
| DayOfWeek | int64 |
| DepTime | float64 |
| CRSDepTime | int64 |
| ArrTime | float64 |
| CRSArrTime | int64 |
| UniqueCarrier | object |
| FlightNum | int64 |
| TailNum | object |
| ActualElapsedTime | float64 |
| CRSElapsedTime | float64 |
| AirTime | float64 |
| ArrDelay | float64 |
| DepDelay | float64 |
| Origin | object |
| Dest | object |
| Distance | int64 |
| TaxiIn | int64 |
| TaxiOut | int64 |
| Cancelled | int64 |
| CancellationCode | object |
| Diverted | int64 |
| CarrierDelay | int64 |
| WeatherDelay | int64 |
| NASDelay | int64 |
| SecurityDelay | int64 |
| LateAircraftDelay | int64 |

Since we are interested in analyzing arrival delays (to avoid waiting at airports for our flights), we will perform exploratory analysis and build a prediction model, using python scripts, for delays by looking at below fields:

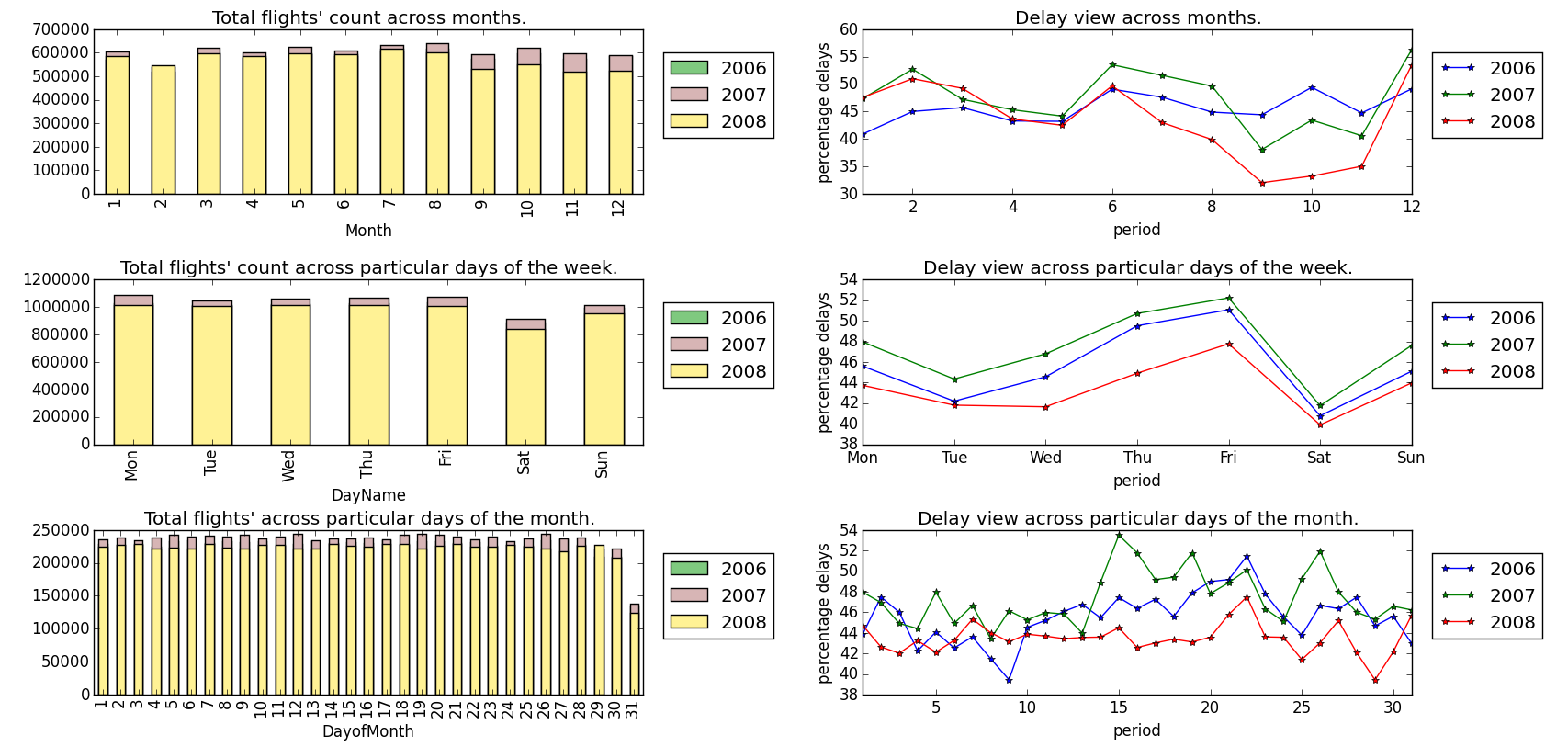
|  |  |
| --- | --- |
| Field | Datatype |
| Year | int64 |
| Month | int64 |
| DayofMonth | int64 |
| DayOfWeek | int64 |
| UniqueCarrier | object |
| ArrDelay | float64 |
| Origin | object |
| Dest | object |
| Distance | int64 |

**Exploratory Analysis:**

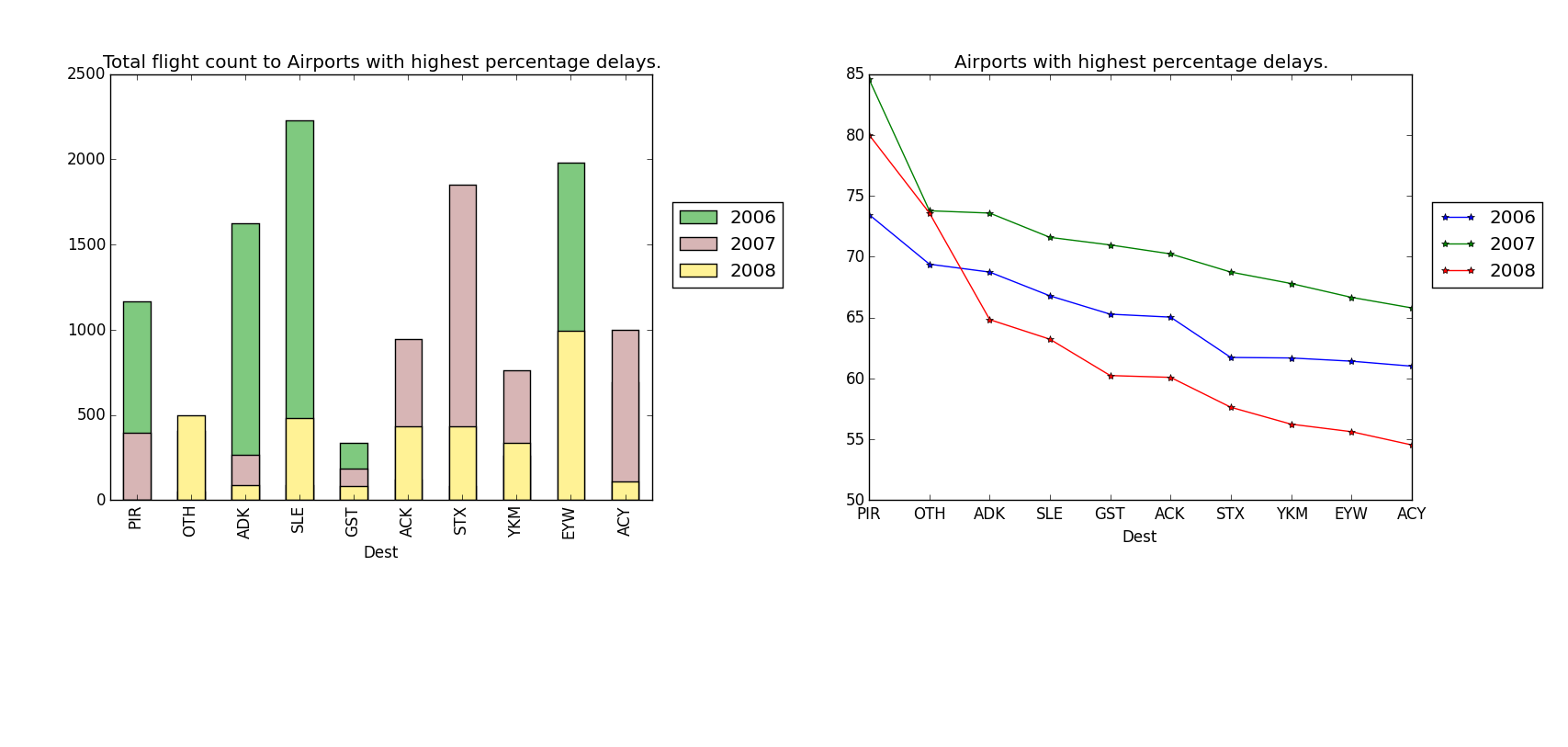
We will be looking at data for 3 years (2006 – 2008). Analyzing the data, we see that there < 3% samples with nulls for ArrDelay column per year. Since this number is small relative to our large dataset, we will safely ignore these and look at all the other data we have.

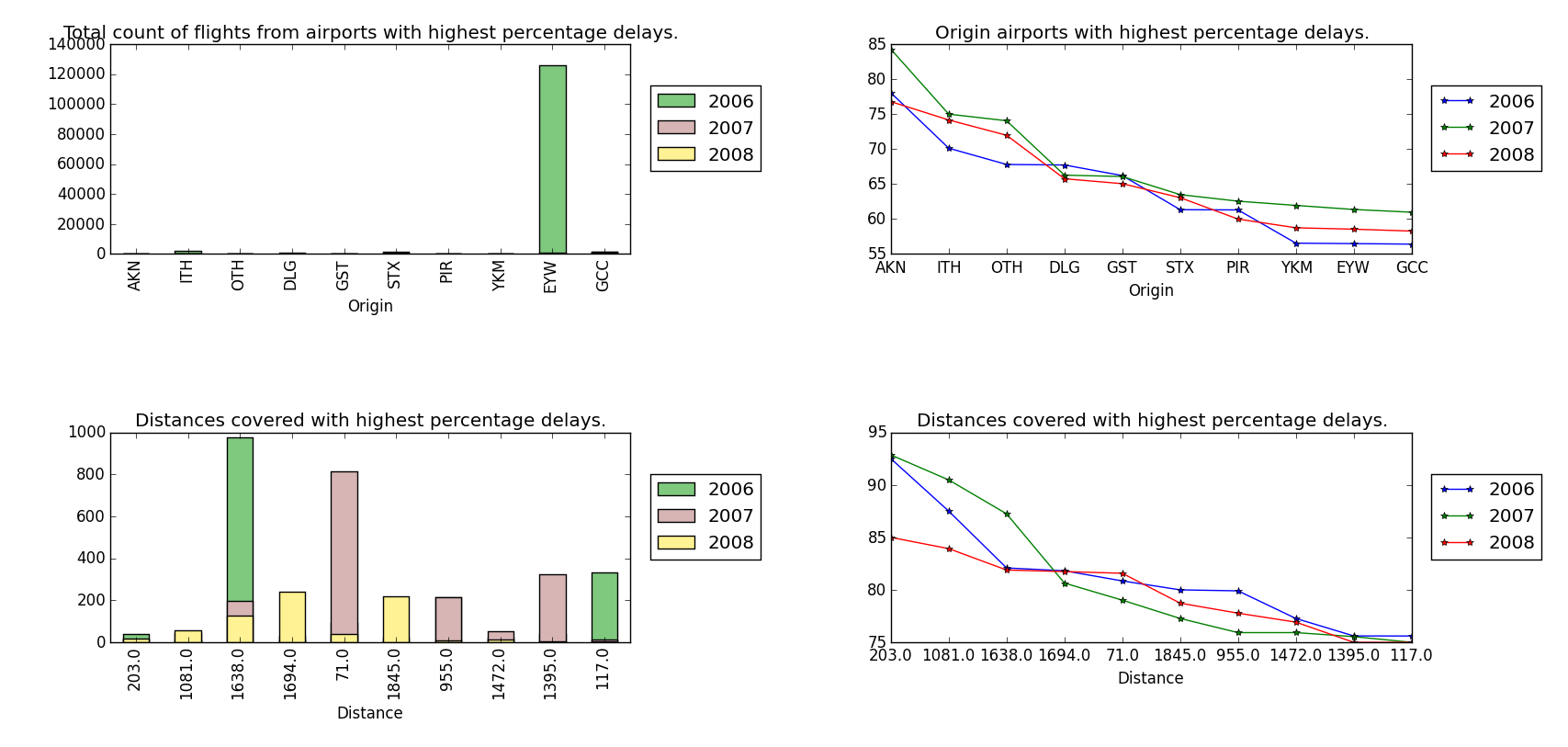
We can plot the total number of flights and percentage delays, calculated as: ((number of flights arrived on-time + number of flights delayed) / number of total flights) \* 100, and we can see trends for various attributes:

By Period:

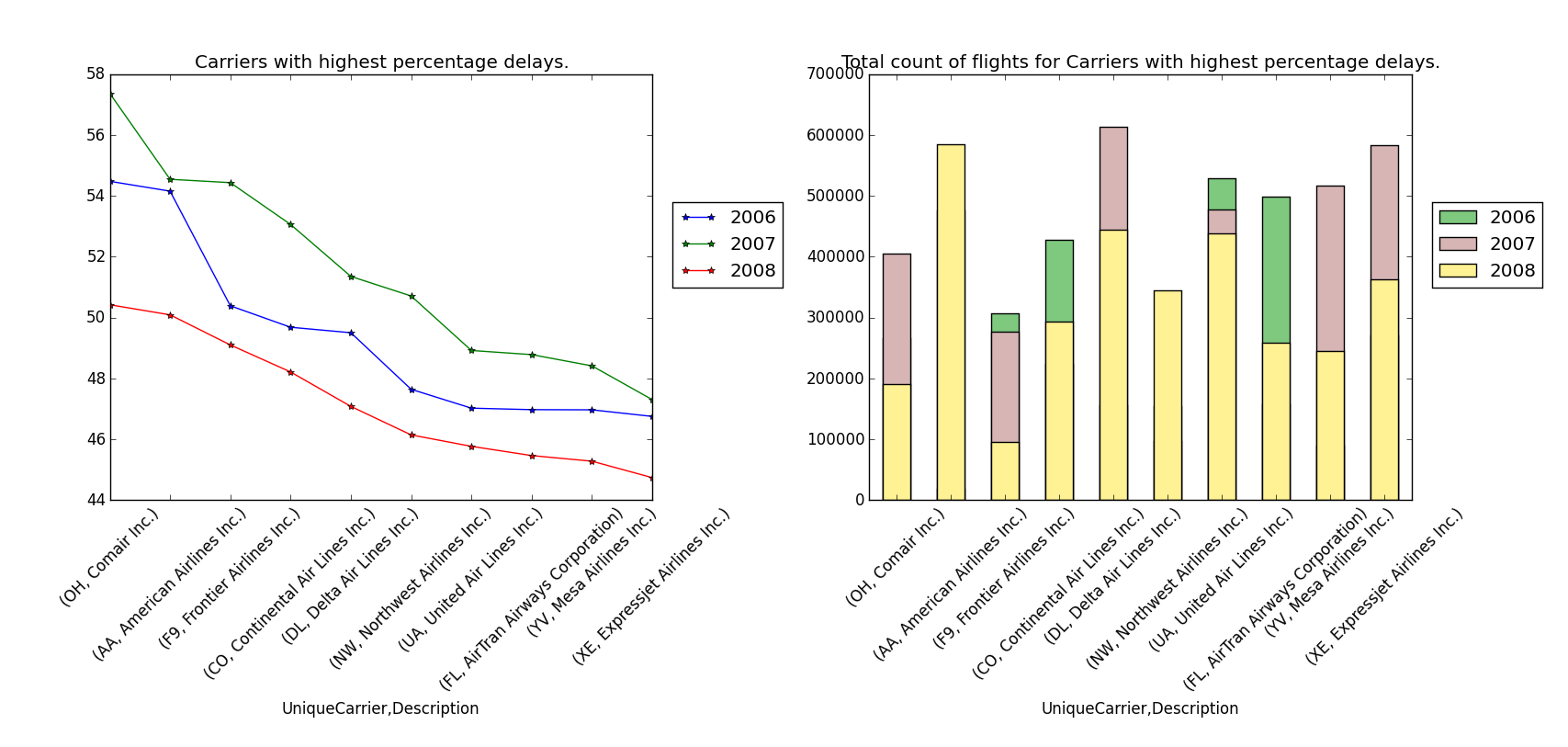


By Airport:





By Airline:



We can see the airline names since the data was merged before plotting with data available [here](http://stat-computing.org/dataexpo/2009/carriers.csv), mapping airline codes to airline names.

As a traveler booking my flight, I would be aware of the airport codes, hence the plot contains those. However, for any person not actually booking a flight to or from that airport, below is a table for the top 10 codes in the Destination as well as Origin airport plots, with data merged from [here](http://stat-computing.org/dataexpo/2009/airports.csv), mapping airport codes to airport locations (city, state):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Destination | location |  | Origin | location |
| PIR | Pierre, SD |  | AKN | King Salmon, AK |
| OTH | North Bend, OR |  | ITH | Ithaka, NY |
| ADK | Adak, AK |  | OTH | North Bend, OR |
| SLE | Salem, OR |  | DLG | Dillingham, AK |
| GST | Gustavus, AK |  | GST | Gustavus, AK |
| ACK | Nantucket, MA |  | STX | Christiansted,VI |
| STX | Christiansted, VI |  | PIR | Pierre, SD |
| YKM | Yakima, WA |  | YKM | Yakima, WA |
| EYW | Key West, FL |  | EYW | Key West, FL |
| ACY | Atlantic City, NJ |  | GCC | Gillette, WY |

We can see clear patterns in the graphs above.

1. More flights generally get delayed in February, June to August and December. More flights get delayed Thursday to Friday (right before the weekend).Flights tend to get delayed more between the 15th and 27th of each month.
2. some airlines definitely have a trend of having more delayed flights than others. (Eg. Comair and American Airlines see the most percentage of delays. United Air Lines has a better rate of delays as compared to American Airlines.)
3. We definitely have some airports which see a higher rate of Arrival delays than others. (highest is PIR in Pierre, SD)
4. We also see higher rate of delays for flights arriving from specific airports as well (highest is from AKN in King Salmon, AK). Also it seems like a small airport based on number of flights flying from there, which makes the higher percentage of delays even more significant.
5. However, with distances covered, it seems like delays are not necessarily more with long distance flights and there is not a trend like that with short-distance flights either. The top ten most delayed paths seem to have a mix of long as well as short distances.

Hence we take the time periods, airlines and airport (destination as well as origin) data-points and try to use them for predicting airline delays.

We will use below fields to predict airline delays:

|  |  |
| --- | --- |
| Field | Datatype |
| Year | int64 |
| Month | int64 |
| DayOfWeek | int64 |
| UniqueCarrier | object |
| ArrDelay | float64 |
| Origin | object |
| Dest | object |

We will be classifying the data as delayed or not based on the value of ArrDelay. If ArrDelay is > 0, then delayed, else not delayed.

Since this is a binary classification problem with labeled data, we will use one of the supervised learning sci-kit learn classification modules.

**Data Wrangling and Cleansing:**

1. The dataset contains nulls for ArrDelay columns and since that is the focal point of our predictive model, we need to rectify that. Since we have an extremely large dataset > 1M records per year, we can delete the rows with nulls (< 3% of data per year).
2. We can also convert the ArrDelay column into a column called status with binary values (if ArrDelay <= 0, then status = 0 (on-time) and if ArrDelay > 0, then status = 1 (delayed)).
3. We can convert all our data into numerical form for ease of use for prediction modules. We will use bash commands to update the airports.csv and carriers.csv to include an id column:

# print line numbers at the start of each line in csv

awk '{print NR, ",",$0}' <original>.csv >> <original>\_lookup.csv

# remove trailing spaces from line numbers in new csv

sed -i "s/\ \,\ /\,/g" <original>\_lookup.csv

# replace the first line with column header

sed -i "s/^1\,/\"id\"\,/g" <original>\_lookup.csv

**Predictive Model:**

Once we have the data we need, we can merge the data to get below columns for each Year:

Month,

DayOfWeek,

DayofMonth,

oid (Origin airport id (merged from airports\_lookup.csv)

aid (Destination airport id merged from airports\_lookup.csv)

cid (Carrier id merged from carriers\_lookup.csv)

We will use the feature encoding module called [OneHotEncoder](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html#sklearn.preprocessing.OneHotEncoder) to convert this dataframe with numerical values to a sparse matrix (since we have a lot of data, a sparse matrix is better than a dense one). The OneHotEncoder, which is a transformer class provided by the sklearn.preprocessing package, is a one-of-K encoder which will transform the set of possible values for categorical features into a set of binary features with only active.

Once the sparse matrix is created we choose the (Stochastic Gradient Descent classifier) [SGDClassifier](http://scikit-learn.org/stable/modules/sgd.html) for training and testing purposes. Since SGDClassifier scales better with large data and works on sparse matrix (which is what our data is converted to due to memory restrictions).

We use 2006 and 2007 data for training and 2008 data for testing purposes. Since we have 43% samples in delayed class and 57% samples in the non-delayed (on-time) class, we need to make sure we train the SGDClassifier with class\_weight=”auto” (to balance out the unequal samples for each outcome (delayed and non-delayed) in our training dataset).

However, once we train and test the data, we see the below metrics for prediction (we are using the accuracy score, confusion matrix and f1-score (which are the basic and most popular metrics)):

accuracy score: 0.547335248326

f1-score : 0.509877073999

confusion\_matrix:

[[2137951 1737574]

[1365456 1614048]]

In order to improve the accuracy, we can try various things:

1. try to modify the threshold of the classes (delayed > 15 min)

accuracy score: 0.608212160736

f1-score : 0.359597147499

confusion\_matrix:

[[3415274 1973564]

[ 712153 754038]]

We can see the accuracy improved but the f1 score drops.

1. Keeping delayed > 15 min, we can then try to reduce the training data by limiting the prediction to particular airports. We try to predict flights arriving late only in DAL (Dallas Love Field) and DFW (Dallas-Fort Worth International) airports.

accuracy score: 0.594255804012

f1-score : 0.334629447512

confusion\_matrix:

[[160054 99948]

[ 31985 33176]]

There is not much change in the scores.

1. We try to add more features and hence keeping 1 and 2 above, we add in another column for Distance.

accuracy score: 0.606209808619

f1-score : 0.333815449929

confusion\_matrix:

[[165036 94966]

[ 33080 32081]]

This gives us a very slightly better accuracy score but the f1 score still remains bad.

**Future Tasks:**

It seems like even though we see definite patterns with delays for certain time periods, carriers and airports, it is certainly not enough to be able to predict arrival delays very accurately.

Weather information per day may be helpful.

Adding more years’ data might help as well. We can probably use [Apache Spark](https://spark.apache.org/docs/1.5.2/) to process more years’ data for training to get a better spread of delayed and non-delayed flights.

If we use Apache Spark (which is a cluster computing system), we can utilize the distributed resources to balance the dataset by oversampling the delayed samples using SMOTE or borderline SMOTE. Due to the large dataset and limited resources right now, implementing SMOTE was unfruitful leading to “Out of Memory” error.

**Conclusion:**

We can definitely see that airline delays are not always caused by weather alone. Consumers can make better choices while booking flights to avoid delays by booking in the months of January, March, April, May, September, October, November. They can avoid end of the months for travel if possible and book between Saturday and Wednesday to avoid delays. Avoiding certain airports, like (OTH) North Bend, OR, (GST) Gustavus, AK, (PIR) Pierre, SD, (YKM) Yakima, WA and (EYW) Key West, FL which have flights arriving late and have flights departing from there arriving late at other places more often, will also help avoid delays. Knowing certain airlines tend to have more delays can help us manage and prepare for delays as well.

However, to actually predict airline delays, due to the large dataset, we would definitely need more time and resources to get an accurate prediction model by implementing any or all of the below techniques:

1. more data
2. more features
3. resampling methods
4. Cluster Computing Framework like Apache Spark having a distributed storage system.