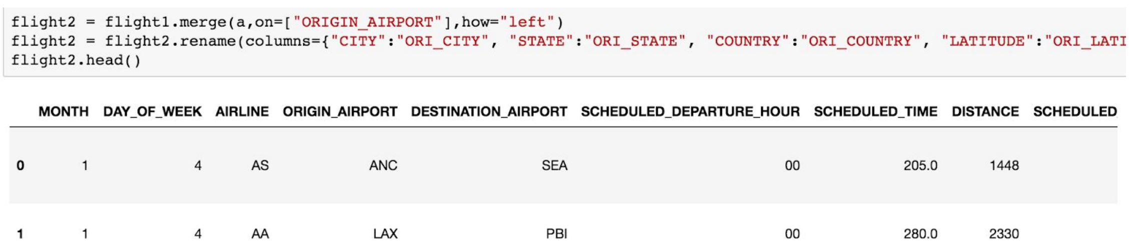
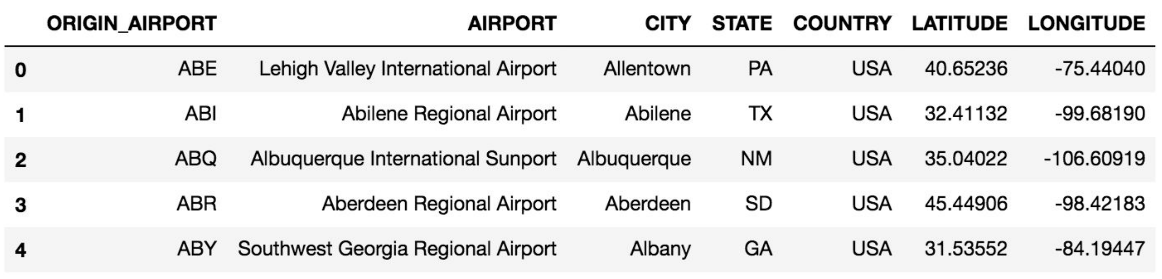
➢ Data Overview - Project background & raw input data

- For our team, we all love traveling during the winter or summer break, and all have suffered a lot by the flight delay problem. So we decide to do this project analyzing some key factors that might influence this delay problem and develop a prediction model for us to use in the future before buying the ticket.
- The raw data were collected and published by the U.S. Department of Transportation’s (DOT), we downloaded the 2017 one-year data from their official website. In the downloading package there are three files: 1. the first one is the flight information on a daily base update frequency (which contains the departure/ arrival airport, air carrier, scheduled departure/ arrival time and the total distance, etc.), 2. the second one is the airport information lookup file, which gives us the geographical information of all the airports within the U.S., the last one for the airlines or air carriers abbreviation name lookup.
- So in order to do this project, we decided to merge all the three files together and got the initial input file which is more than 2 Gigabytes large and including around 6 million records and 40 features.
- We did the file merging part by using the Pandas DataFrame package, after that we got some new features such as the state and city of the airports and their corresponding latitude & longitude.



```
flight2 = flight1.merge(a,on=["ORIGIN_AIRPORT"],how="left")
flight2 = flight2.rename(columns={"CITY":"ORI_CITY", "STATE":"ORI_STATE", "COUNTRY":"ORI_COUNTRY", "LATITUDE":"ORI_LATI
flight2.head()
```

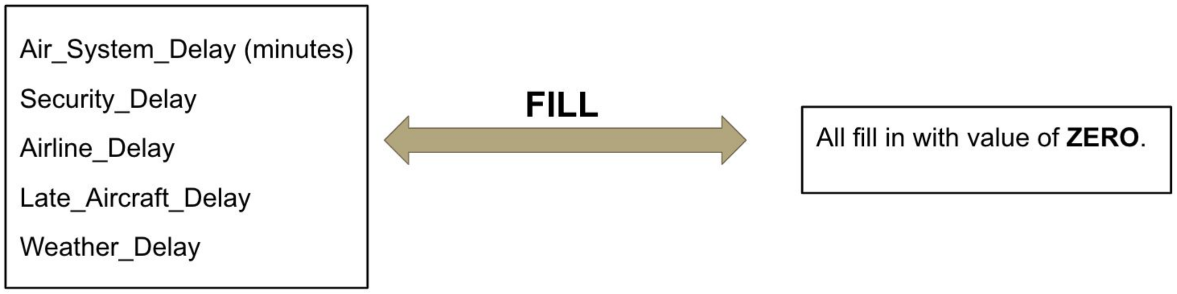
	MONTH	DAY_OF_WEEK	AIRLINE	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDULED_DEPARTURE_HOUR	SCHEDULED_TIME	DISTANCE	SCHEDULED
0	1	4	AS	ANC	SEA	00	205.0	1448	
1	1	4	AA	LAX	PBI	00	280.0	2330	



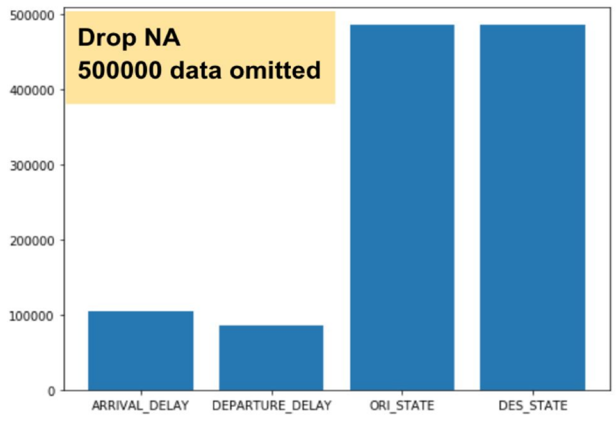
	ORIGIN_AIRPORT	AIRPORT	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
0	ABE	Lehigh Valley International Airport	Allentown	PA	USA	40.65236	-75.44040
1	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
2	ABQ	Albuquerque International Sunport	Albuquerque	NM	USA	35.04022	-106.60919
3	ABR	Aberdeen Regional Airport	Aberdeen	SD	USA	45.44906	-98.42183
4	ABY	Southwest Georgia Regional Airport	Albany	GA	USA	31.53552	-84.19447

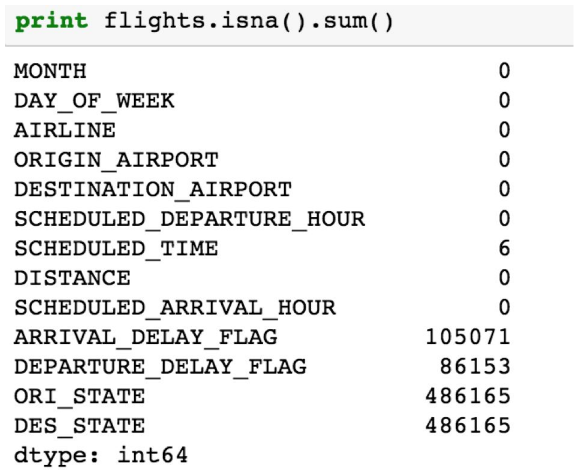
➢ Data Cleaning

- As we all probably know that for this kind of raw input dataset there might be some missing values or outliers, so before analyzing and training models, we cleaned the raw data at first, by using Pandas Dataframe again. For example, the air system delay feature, it gives us the amount delay time in minutes due to air system problem, but if the delay caused by other reason or arrives on time or earlier the value would be empty, so we decide to fill in the missing values for those five columns with zero.



- Unfortunately, there are some missing values we have no idea how to fill in. For the month of October, all the airport and flight time information were missing, which takes up to 8% percent of the whole dataset. Since is no way for us to fill in the missing values and we already have ~6 million records which is sufficient, so we eliminated all the records for October.

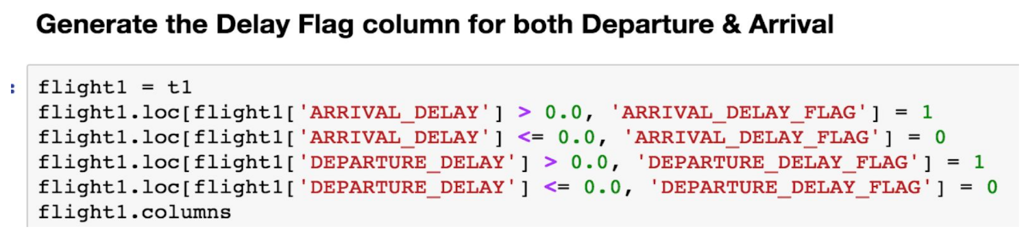




```
print flights.isna().sum()
```

```
MONTH                          0
DAY_OF_WEEK                    0
AIRLINE                        0
ORIGIN_AIRPORT                 0
DESTINATION_AIRPORT            0
SCHEDULED_DEPARTURE_HOUR       0
SCHEDULED_TIME                 6
DISTANCE                       0
SCHEDULED_ARRIVAL_HOUR         0
ARRIVAL_DELAY_FLAG        105071
DEPARTURE_DELAY_FLAG       86153
ORI_STATE                 486165
DES_STATE                 486165
dtype: int64
```

- Last we also generate some new features, the first one: “Delay or not” is our target, 1 means the flight delayed, 0 means not delay. So we calculate the difference between actual arrival time and the scheduled arrival, if the difference is larger than zero that means a delay, if it’s smaller or equals to zero, which means not a delayed flight. So the final input file contains 5.5 million records and 44 features.



Generate the Delay Flag column for both Departure & Arrival

```
flight1 = t1
flight1.loc[flight1['ARRIVAL_DELAY'] > 0.0, 'ARRIVAL_DELAY_FLAG'] = 1
flight1.loc[flight1['ARRIVAL_DELAY'] <= 0.0, 'ARRIVAL_DELAY_FLAG'] = 0
flight1.loc[flight1['DEPARTURE_DELAY'] > 0.0, 'DEPARTURE_DELAY_FLAG'] = 1
flight1.loc[flight1['DEPARTURE_DELAY'] <= 0.0, 'DEPARTURE_DELAY_FLAG'] = 0
flight1.columns
```

➢ Visualizations & Analytics

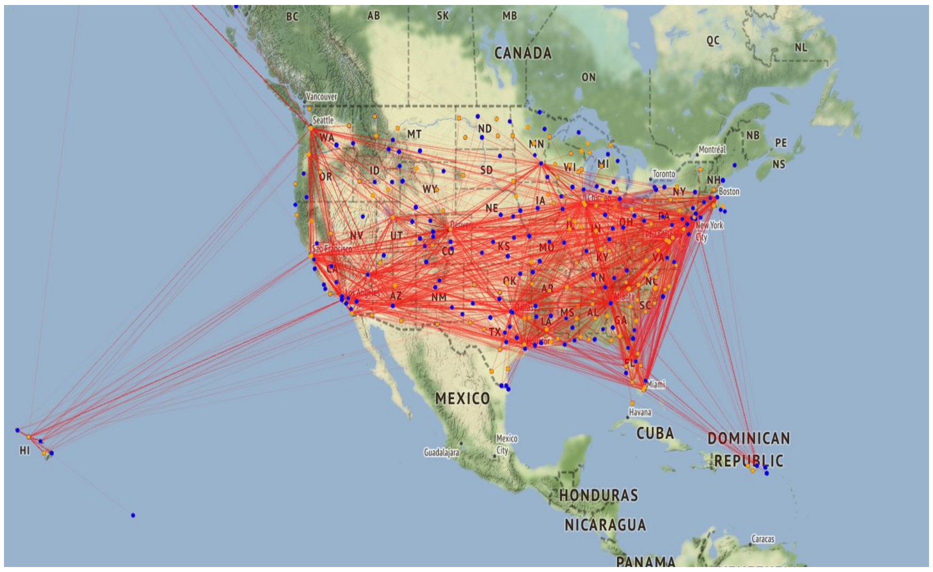
1. Route Map (10,000 records)

- Based on package “Folium” (topographic basemap):

  Blue Point: Departure Airport;

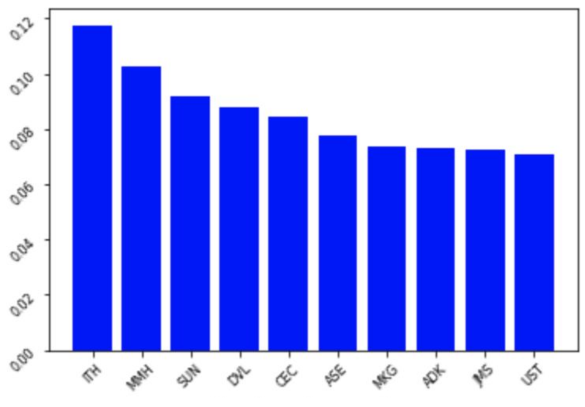
  Orange Point: Arrival Airport;

  Red line: Airline Route;

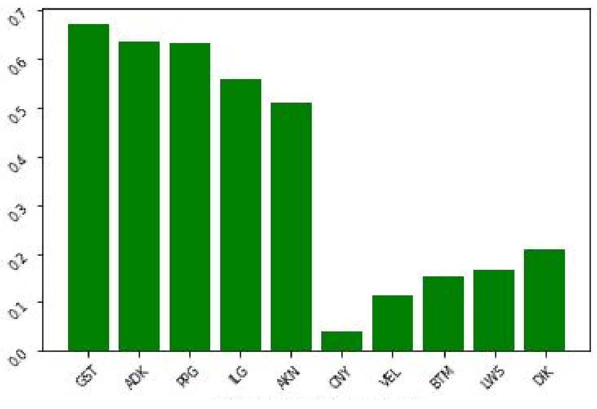


2. Cancellation rate and delay rate on airports

- By computing the cancellation rate and delay rate of every airport, we plot the histograms and pie charts by choosing top 10 airports with high cancellation rate and top 5 / last 5 airports with delay rate.

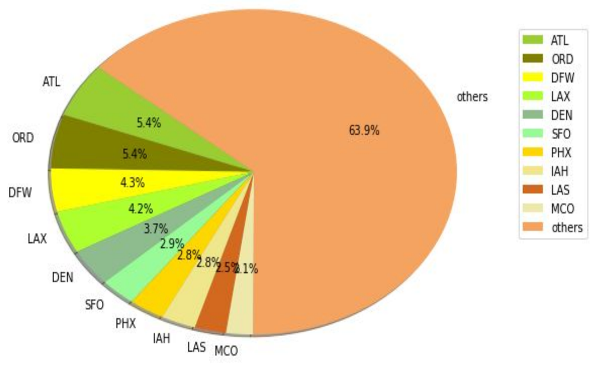


Cancellation rate by Airport

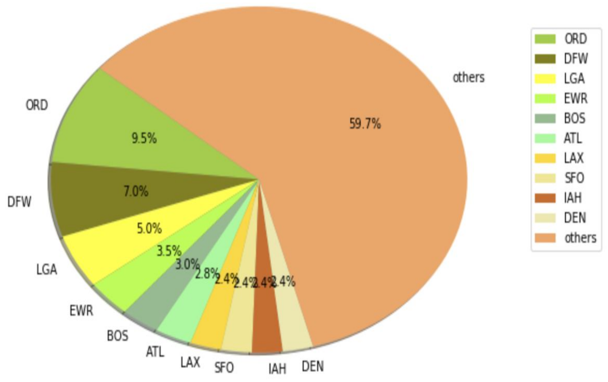


Delay rate by Airport

- From the histograms above, we can know that "Ithaca Tompkins Regional Airport" has the highest cancellation rate (0.12). Delay rate of "Gustavus Airport" is 0.7, while "Canyonlands Field Airport" performs best with the lowest delay rate (0.05).



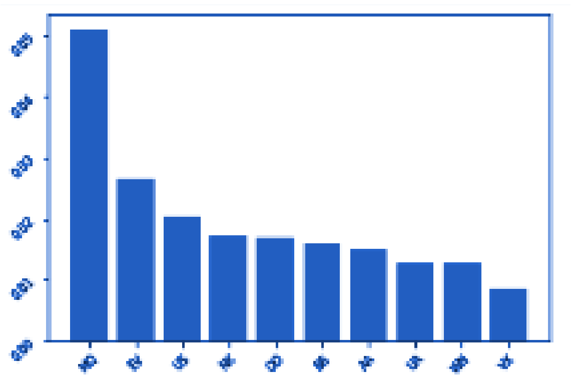
Percentage of cancelled flight by Airport

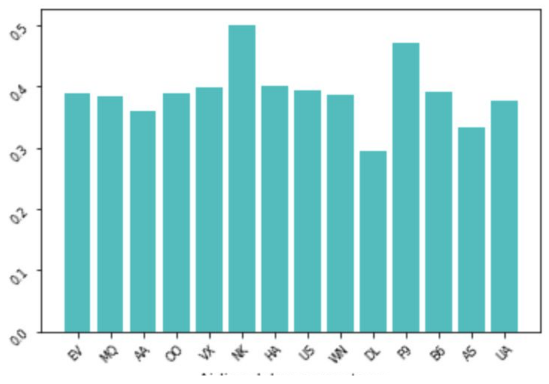


Percentage of delayed flight by Airport

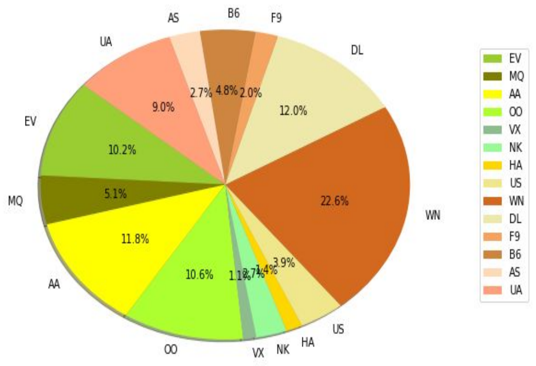
- In these two pie charts, "O'hare international airport" and "atlanta international airport" had more cancelled or delayed flights than other airports.

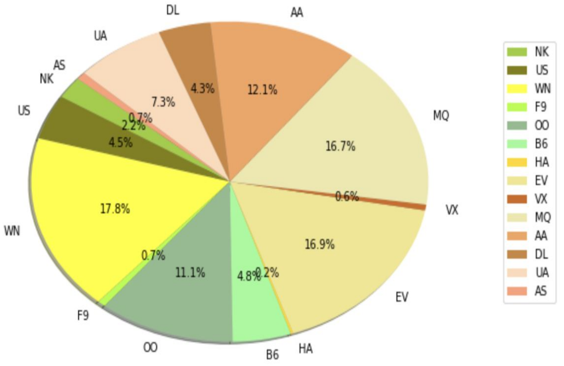
3. Cancellation rate and delay rate on airline/Carrier





Cancellation rate by Airline **Delay rate by Airline**



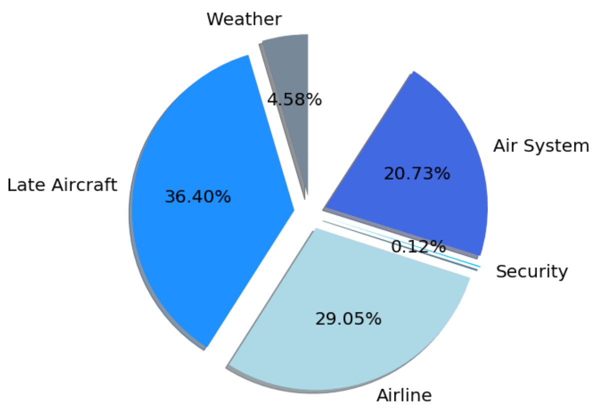


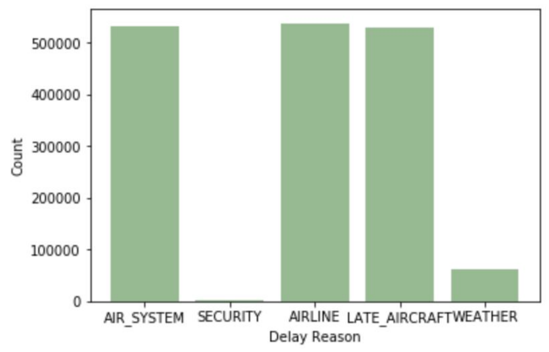
Percentage of cancelled flight by Airport **Percentage of delayed flight by Airport**

- Here, VX(Alaska Airline) is the best one that we recommend customers to choose, for it's lowest cancellation rate and tiny percentage of cancelled or delayed flights. While (WN) Southwest Airline has a lot of problem flights with highest percentage both in cancelled and delayed flights.

4. Visualization of Delay reason

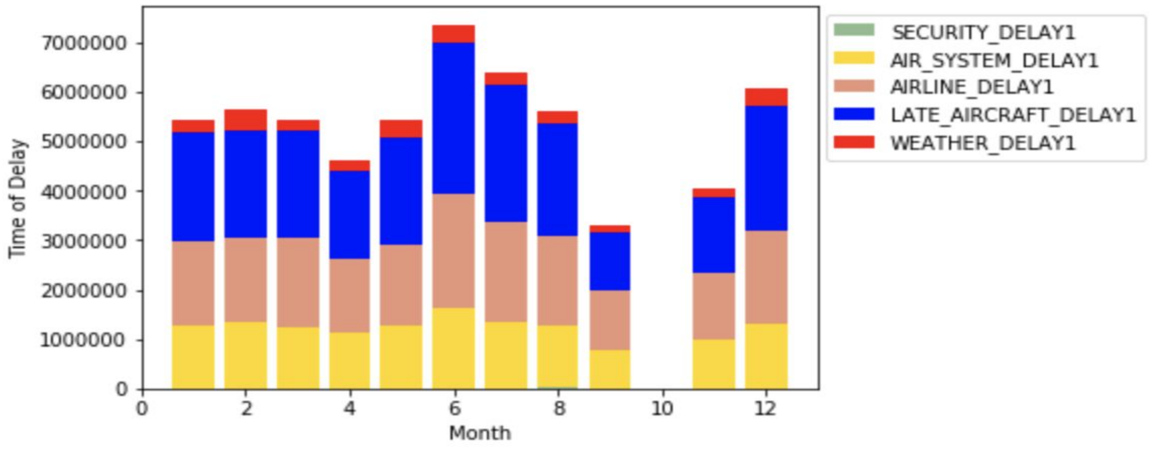
- There are five reasons: Late Aircraft, Airline, Air System, Weather, Security.



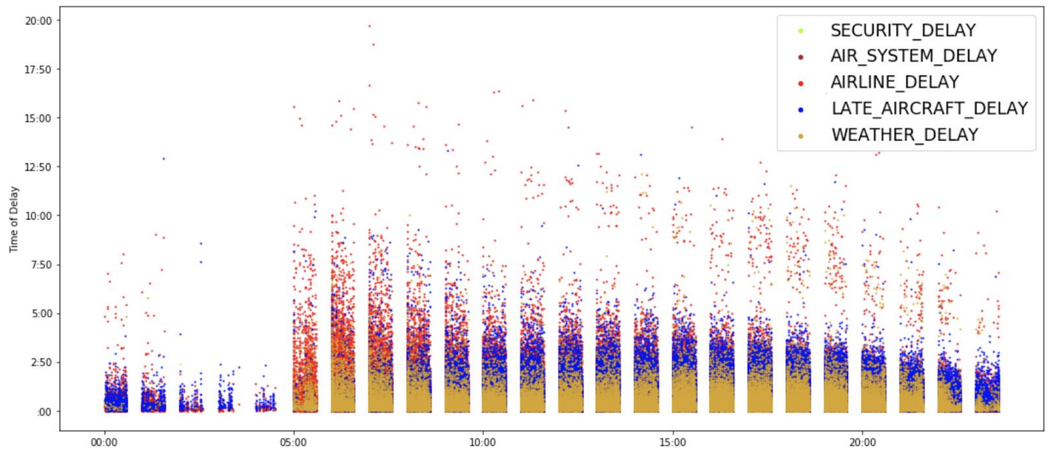


Accumulated time of Delay Reason **Counts of Delay Reason**

- From the Pie chart, the delay time caused by late aircraft is more than one third, while security just take 1.2%. Also in the bar plot, the counts of delayed flights are caused by Air-system, Airline and Aircraft are similar, much more than security and weather.



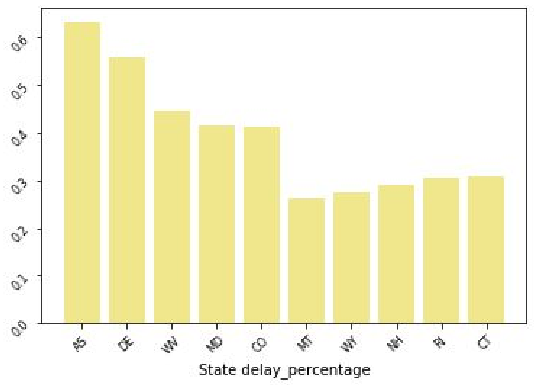
- By looking at the distribution of these reasons based on month, we can see that the highest accumulated delay time was in July, while the lowest in September.

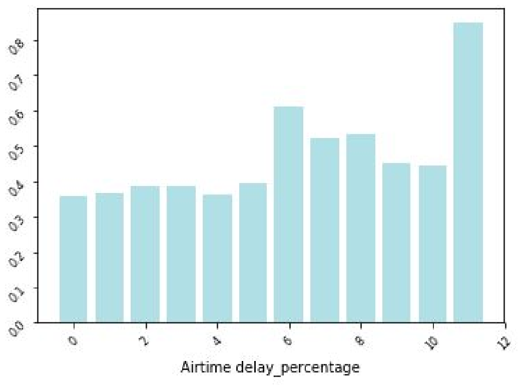


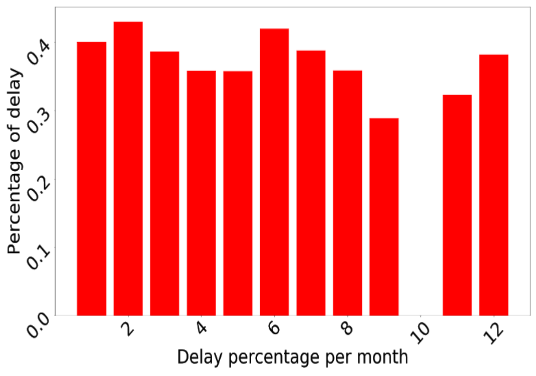
- In Scatter Plot, the points concentrated at the bottom of the figure, while some red points are on the top, which means the delay time caused by Airline Delay are usually longer than other reason.

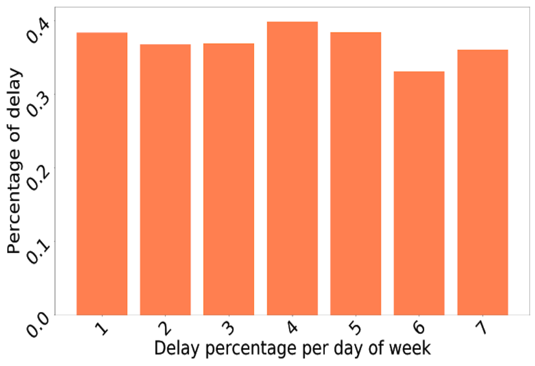
5. Delay rate on State / Airtime / Month / Week / Hour

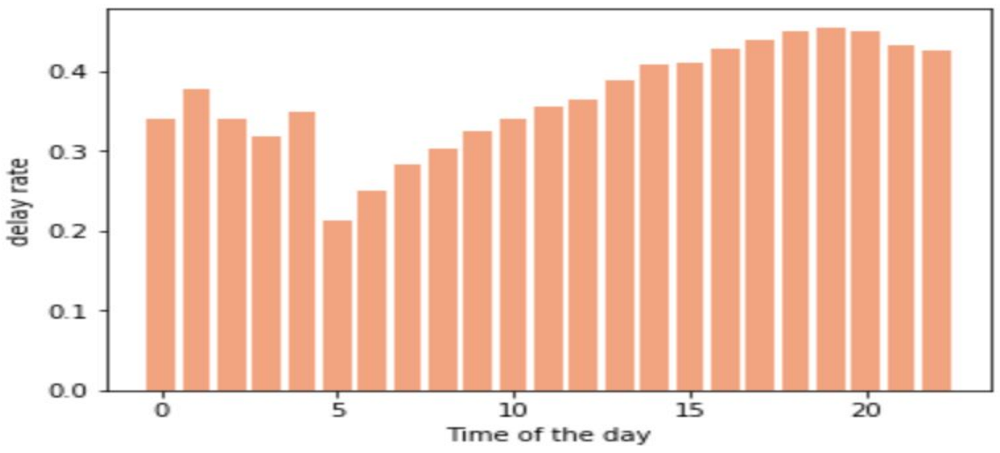
- Firstly we plotted the top 5 states that have the highest delay rate as long with the 5 states have the lowest delay rate. States like AS(an unknown island), DE, WN, MD have a common feature that they are offshore,in which weather reason may count more on those states. While the states like MT, WY, NH have somehow low delay rate. The reason may be they are located interior.
- Secondly we plotted how delay rate is influenced by airtime. We can see 11-hour flight has the highest delay rate while 1~5 hour are lower.
- Thirdly we went through the relationship between month and delay rate. We found February and June have the highest delay rate while September and November are safer. That may be caused by potential extreme weather in winter and summer.
- Fourthly it comes to the relationship between day_of_week and delay rate. It's obvious that Saturday has a relatively low delay possibility, and it's not distinct for other days.
- Finally we plotted the var plot between 24 hours of a day with delay rate. We can notice that the delay rate goes up after 5 am, and at 7pm reaches peak. So we recommend people to try to take earlier flights.





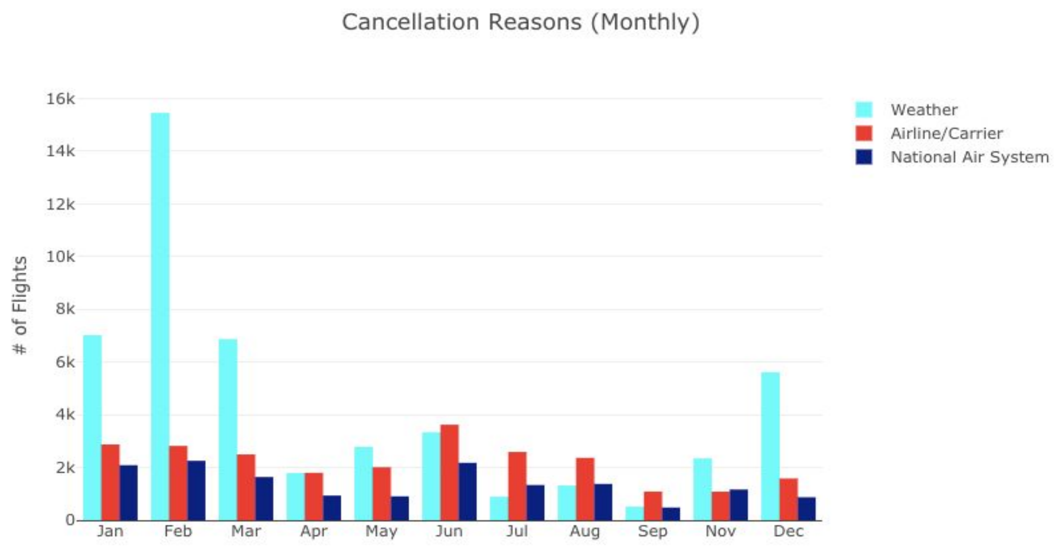


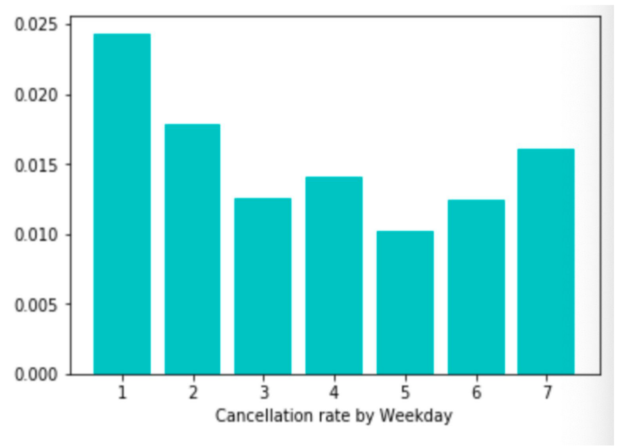


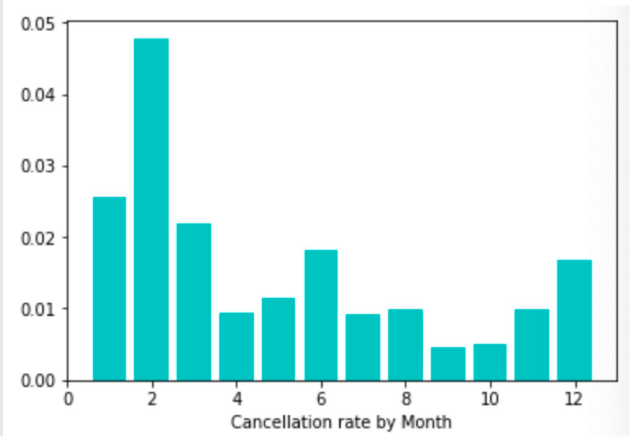


6. Cancellation rate on Month / Day

- We plotted the cancel reason by month, the general trend is weather reason counts mostly, next comes to Airline / Carrier reason, and last is National Air System reason. It's impressive that in the first quarter weather reason cause many cancellation especially in February.
- Then for the trend by weekday, we can see Monday has a relatively high cancel rate while Friday is lower, but we didn't figure out if there is exact causality existing.



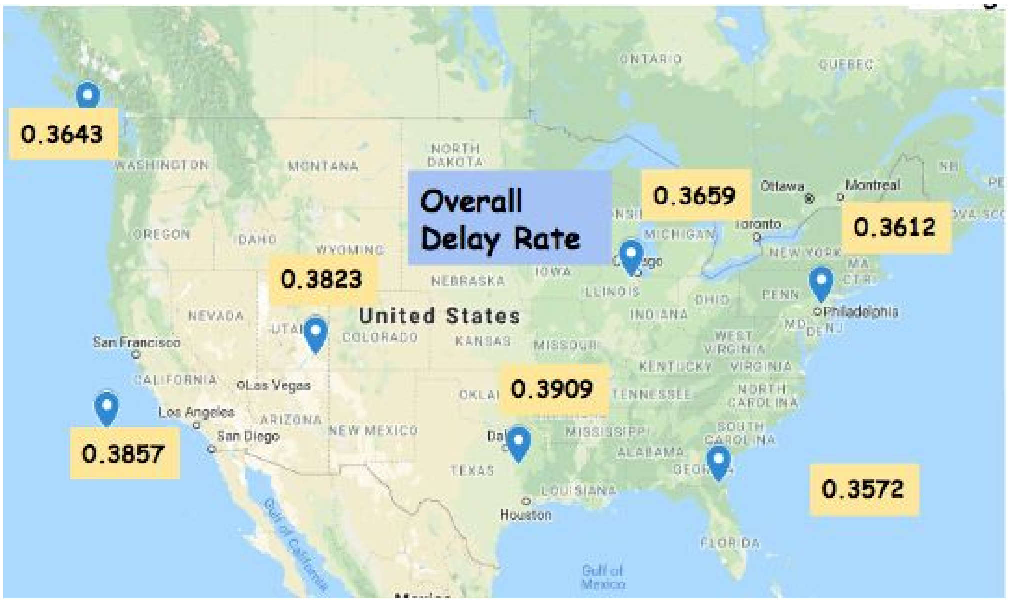


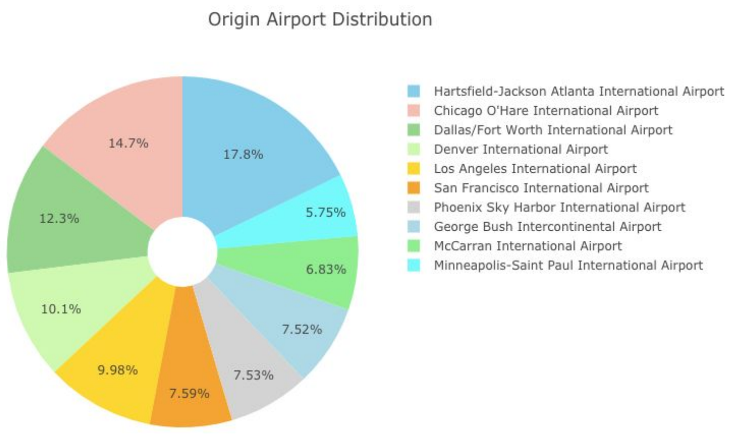


7. Origin Airports Clustering

- We used Kmeans to cluster the origin airports and we divided them into 7 clusters. Then we calculated the delay rate in every cluster and we found the cluster near to Texas and California has a relatively high delay rate.
- We also got how each cluster is affected by weather, the result shows that the cluster near to Texas and Florida are mostly affected by weather.
- Finally we got the states included in each cluster, and the busiest city in each cluster, they are :

  Los Angeles International Airport / **Dallas**/Fort Worth International Airport / Hartsfield-Jackson **Atlanta** International Airport / **John F. Kennedy** International Airport / **Denver** International Airport / **Seattle-Tacoma** International Airport / **Chicago** O'Hare International Airport.
- While they are exactly corresponding to the busiest 7 airports in America, so this clustering totally makes sense.





➢ Prediction Models

1. Data preprocessing

After analyzing the data, we reprocessed data based on the one we used in visualization.

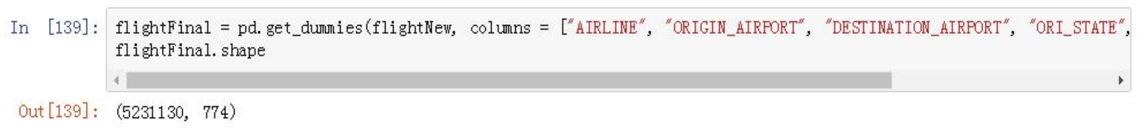
- Categorical to dummy

  Since we can not just simply change categorical variables into labels like 1-10(because if we did that then the difference between each category would be difference, which may cause the variable weight difference when training the model, e.g. the difference between 1 and 10 equals to 9, however, the difference between 1 and 2 is only 1, which may give a misleading signal to the model), sp we decided to change them to dummy variables. So finally we got datasets with 5231130 rows and 774 features.

e.g. A changing to: 0 0 1

B changing to: 0 1 0

C changing to: 1 0 0



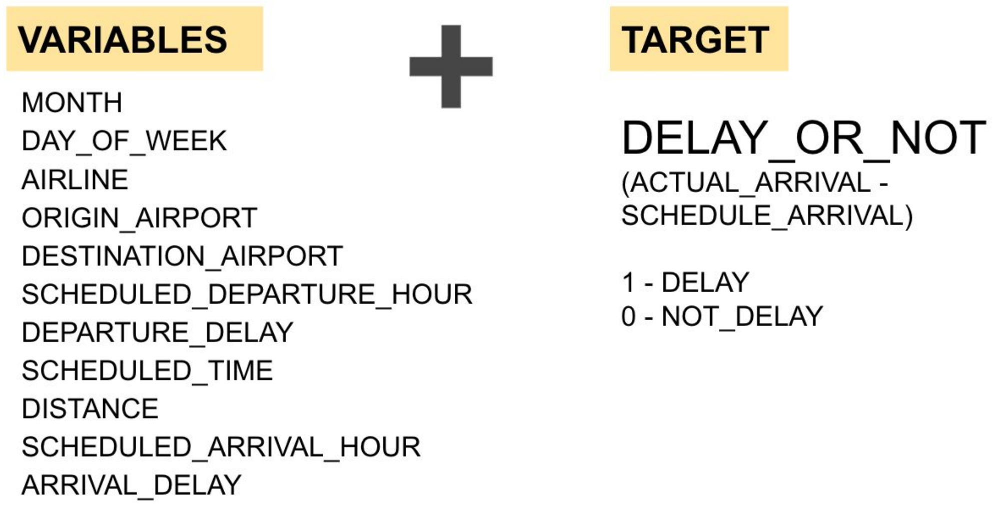
```
In [139]: flightFinal = pd.get_dummies(flightNew, columns = ["AIRLINE", "ORIGIN_AIRPORT", "DESTINATION_AIRPORT", "ORI_STATE",
          flightFinal.shape

Out[139]: (5231130, 774)
```

- Model variable selection

  Since our model was real-life demand oriented, we chose only variables that we could acquire from real-life, like airlines official websites.

  So we chose 11 variables, and 1 target as shown below:



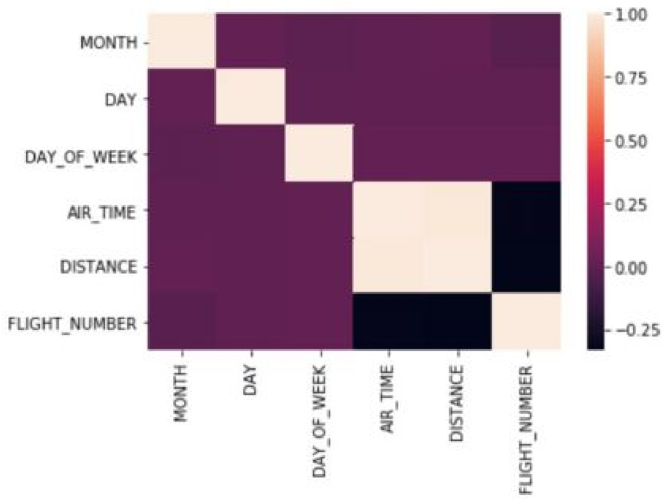
VARIABLES + **TARGET**

MONTH
DAY_OF_WEEK
AIRLINE
ORIGIN_AIRPORT
DESTINATION_AIRPORT
SCHEDULED_DEPARTURE_HOUR
DEPARTURE_DELAY
SCHEDULED_TIME
DISTANCE
SCHEDULED_ARRIVAL_HOUR
ARRIVAL_DELAY

DELAY_OR_NOT
(ACTUAL_ARRIVAL - SCHEDULE_ARRIVAL)

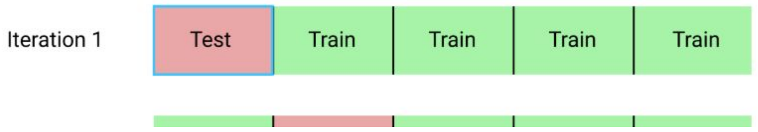
1 - DELAY
0 - NOT_DELAY

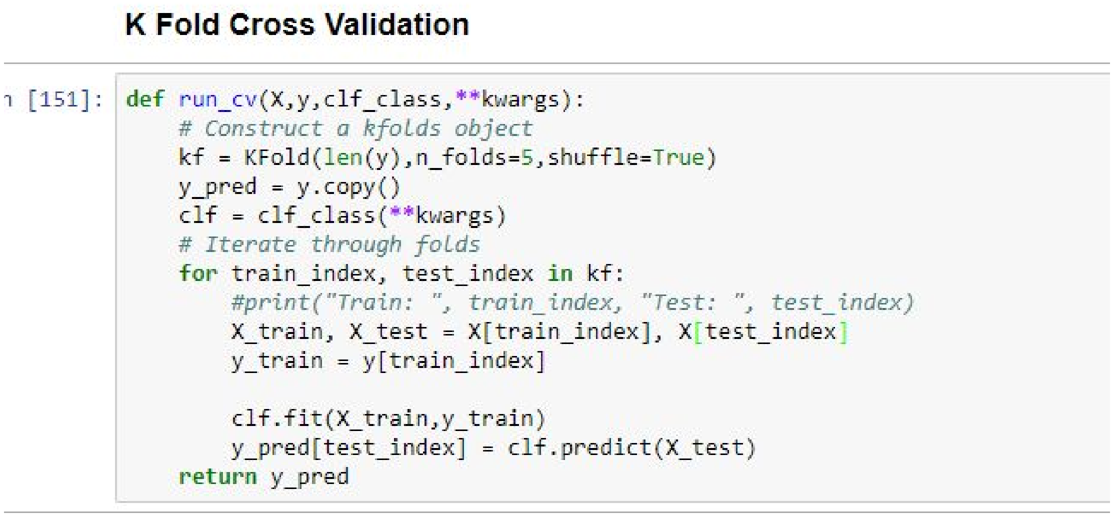
- We also used Pearson correlation to select continuous variables. White blocks means high correlation between two variables. The darker the color, the less correlation two variables will have. So from the plot, we can see that ” air_time” and “distance” are highly correlated, so we chose only one of them which was distance as our variable.



2. K-fold cross-validation

- Since the data was ready, we split it into training and test data by using K-fold cross-validation. It is a good method to split data. We could fully use the whole dataset and avoid overfitting and selection bias which might caused by random split.





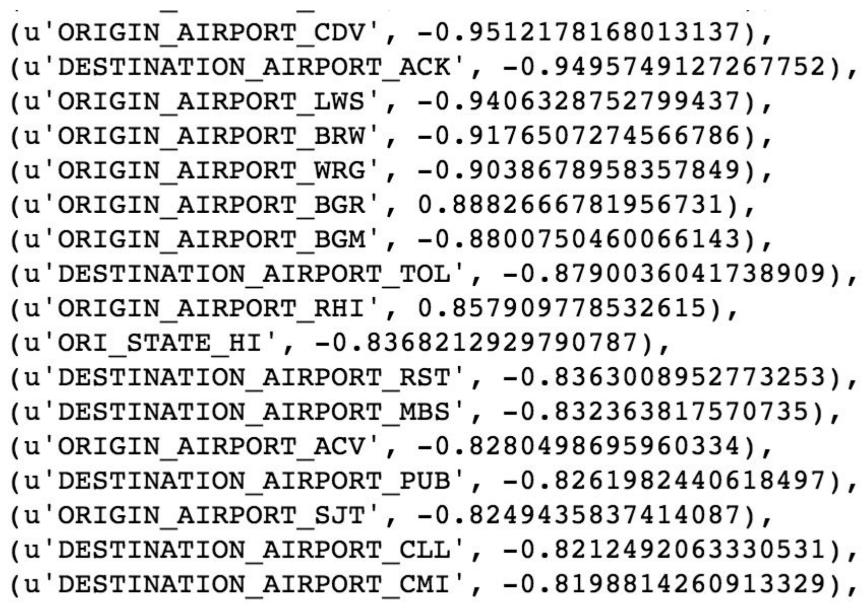
K Fold Cross Validation

```
n [151]: def run_cv(X,y,clf_class,**kwargs):
             # Construct a kfolds object
             kf = KFold(len(y),n_folds=5,shuffle=True)
             y_pred = y.copy()
             clf = clf_class(**kwargs)
             # Iterate through folds
             for train_index, test_index in kf:
                 #print("Train: ", train_index, "Test: ", test_index)
                 X_train, X_test = X[train_index], X[test_index]
                 y_train = y[train_index]

                 clf.fit(X_train,y_train)
                 y_pred[test_index] = clf.predict(X_test)
             return y_pred
```

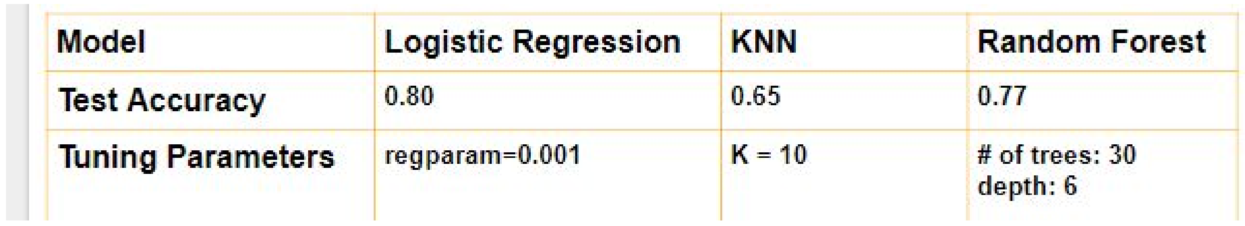
3. Prediction analysis

- We totally used 3 different models, which are logistic regression, K nearest neighbor and random forest. We used both sklearn and mllib packages to train our data.
- In Logistic Regression, we also computed feature weights, which means how much each feature contributes to predicted results. The higher positive weight a feature has, the more delay possibility a flight could have.



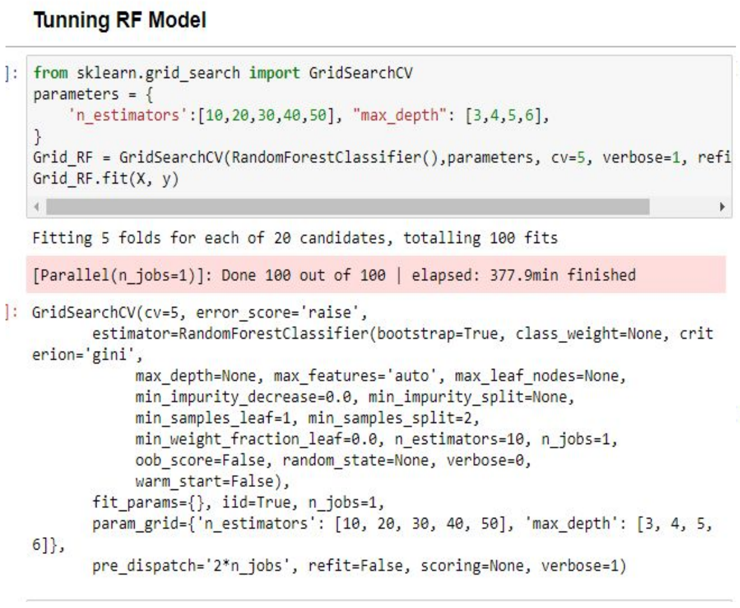
```
(u'ORIGIN_AIRPORT_CDV', -0.9512178168013137),
(u'DESTINATION_AIRPORT_ACK', -0.9495749127267752),
(u'ORIGIN_AIRPORT_LWS', -0.9406328752799437),
(u'ORIGIN_AIRPORT_BRW', -0.9176507274566786),
(u'ORIGIN_AIRPORT_WRG', -0.9038678958357849),
(u'ORIGIN_AIRPORT_BGR', 0.8882666781956731),
(u'ORIGIN_AIRPORT_BGM', -0.8800750460066143),
(u'DESTINATION_AIRPORT_TOL', -0.8790036041738909),
(u'ORIGIN_AIRPORT_RHI', 0.857909778532615),
(u'ORI_STATE_HI', -0.8368212929790787),
(u'DESTINATION_AIRPORT_RST', -0.8363008952773253),
(u'DESTINATION_AIRPORT_MBS', -0.832363817570735),
(u'ORIGIN_AIRPORT_ACV', -0.8280498695960334),
(u'DESTINATION_AIRPORT_PUB', -0.8261982440618497),
(u'ORIGIN_AIRPORT_SJT', -0.8249435837414087),
(u'DESTINATION_AIRPORT_CLL', -0.8212492063330531),
(u'DESTINATION_AIRPORT_CMI', -0.8198814260913329),
```

- For best results, we also tuned parameters for all 3 models,and we used grid search for models in sklearn. We finally found logistic regression performed the best in both packages, with accuracy 0.8.



Model	Logistic Regression	KNN	Random Forest
Test Accuracy	0.80	0.65	0.77
Tuning Parameters	regparam=0.001	K = 10	# of trees: 30 depth: 6

- The snapshot shows below talked about how we performed the Grid Search function embedded in the sklearn package, which help us a lot to process of tuning model parameters easily:

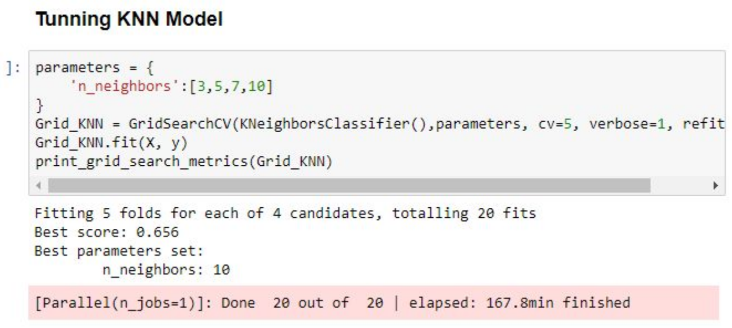


Tunning RF Model

```
]: from sklearn.grid_search import GridSearchCV
   parameters = {
       'n_estimators':[10,20,30,40,50], "max_depth": [3,4,5,6],
   }
   Grid_RF = GridSearchCV(RandomForestClassifier(),parameters, cv=5, verbose=1, refi
   Grid_RF.fit(X, y)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed: 377.9min finished
```

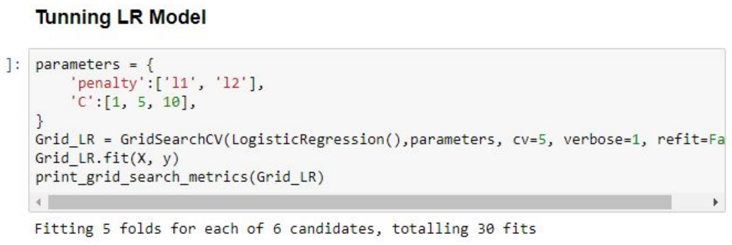
```
]: GridSearchCV(cv=5, error_score='raise',
          estimator=RandomForestClassifier(bootstrap=True, class_weight=None, crit
   erion='gini',
               max_depth=None, max_features='auto', max_leaf_nodes=None,
               min_impurity_decrease=0.0, min_impurity_split=None,
               min_samples_leaf=1, min_samples_split=2,
               min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
               oob_score=False, random_state=None, verbose=0,
               warm_start=False),
          fit_params={}, iid=True, n_jobs=1,
          param_grid={'n_estimators': [10, 20, 30, 40, 50], 'max_depth': [3, 4, 5,
   6]},
          pre_dispatch='2*n_jobs', refit=False, scoring=None, verbose=1)
```



Tunning KNN Model

```
]: parameters = {
       'n_neighbors':[3,5,7,10]
   }
   Grid_KNN = GridSearchCV(KNeighborsClassifier(),parameters, cv=5, verbose=1, refit
   Grid_KNN.fit(X, y)
   print_grid_search_metrics(Grid_KNN)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Best score: 0.656
Best parameters set:
        n_neighbors: 10
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 167.8min finished
```



Tunning LR Model

```
]: parameters = {
       'penalty':['l1', 'l2'],
       'C':[1, 5, 10],
   }
   Grid_LR = GridSearchCV(LogisticRegression(),parameters, cv=5, verbose=1, refit=Fa
   Grid_LR.fit(X, y)
   print_grid_search_metrics(Grid_LR)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
```

➢ Future Improvements

1. Fix the missing data problem for October. We have found some other source than the website of the U.S Department of Transportation (DOT).
2. More years of data to train the model, which will definitely reduce the bias for just using the single year input data.
3. Better feature engineering (more new features, etc.). For example, as the TA suggests us to consider the delay influences between each air carrier, like if the Ameriacan Airline was delayed, which may also caused the delay of the Alaska Airline as well.
4. Apply more complex model, like Gradient Boosting Tree, Recurrent Neural Network (LSTM) to play with this trend data.
5. Since the runtime is really ulgy right now, due to the amount of dummy variables we have in the data. So as the TA suggests, we should use some smarter way to store the data, such as storing them as a sparsevector, etc.. And we also Google about this kind of problem that we could use the function called Recursive Feature Elimination (RFE), which is also embedded in the python sklearn package, this function might help us to eliminate some useless dummy variables to reduce the redundancy of our data.