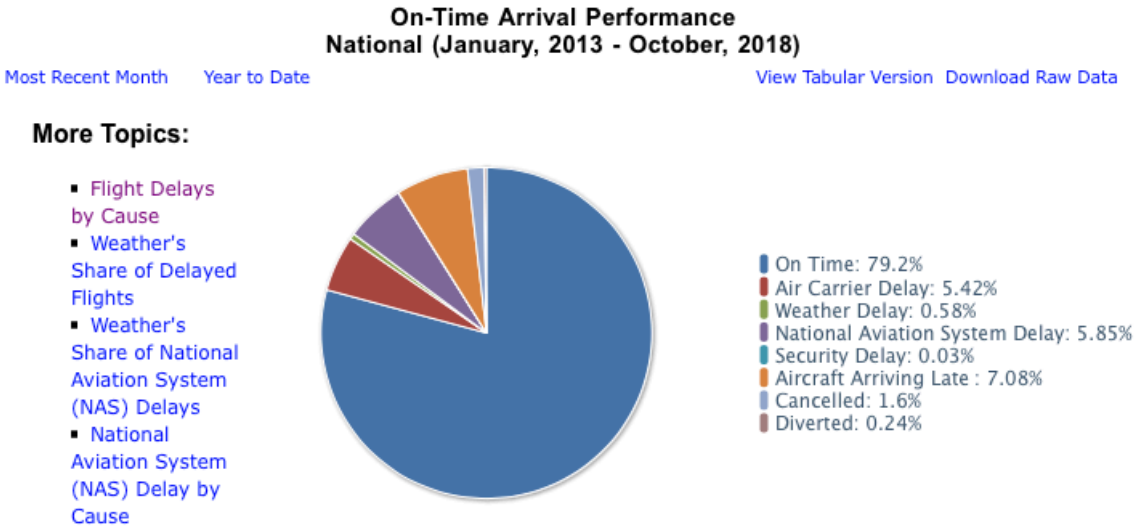
Big Data - Projet - Predicting Airline Delays with Hadoop

SU Hao 20140603
WANG Zhaoxia 20140610

- Introduction

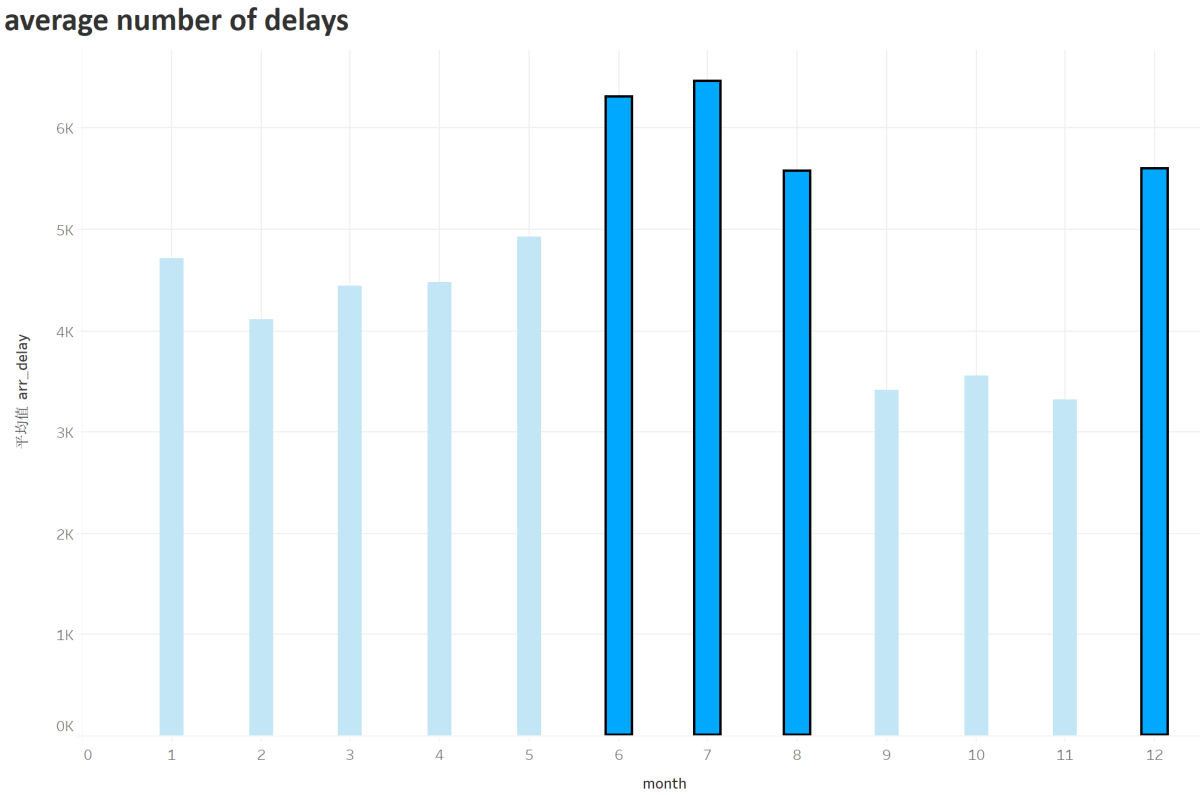
According to the dataset from “Bureau of Transportation Statistics”, we can see that approximately 20%of airline flights are delayed or cancelled. That made a high costs to both travelers and airlines.
Our project is managed to build a supervised learning model, that can predict airline delay from the historical flight data, and maybe some weather informations.
It is possible that some new technologies have been popularized in recent years, and we have found that the overall trend of flight delays is improving over time. Considering the timeliness of information and the performance of the model, we chose the flight data from 2013 - 2018.



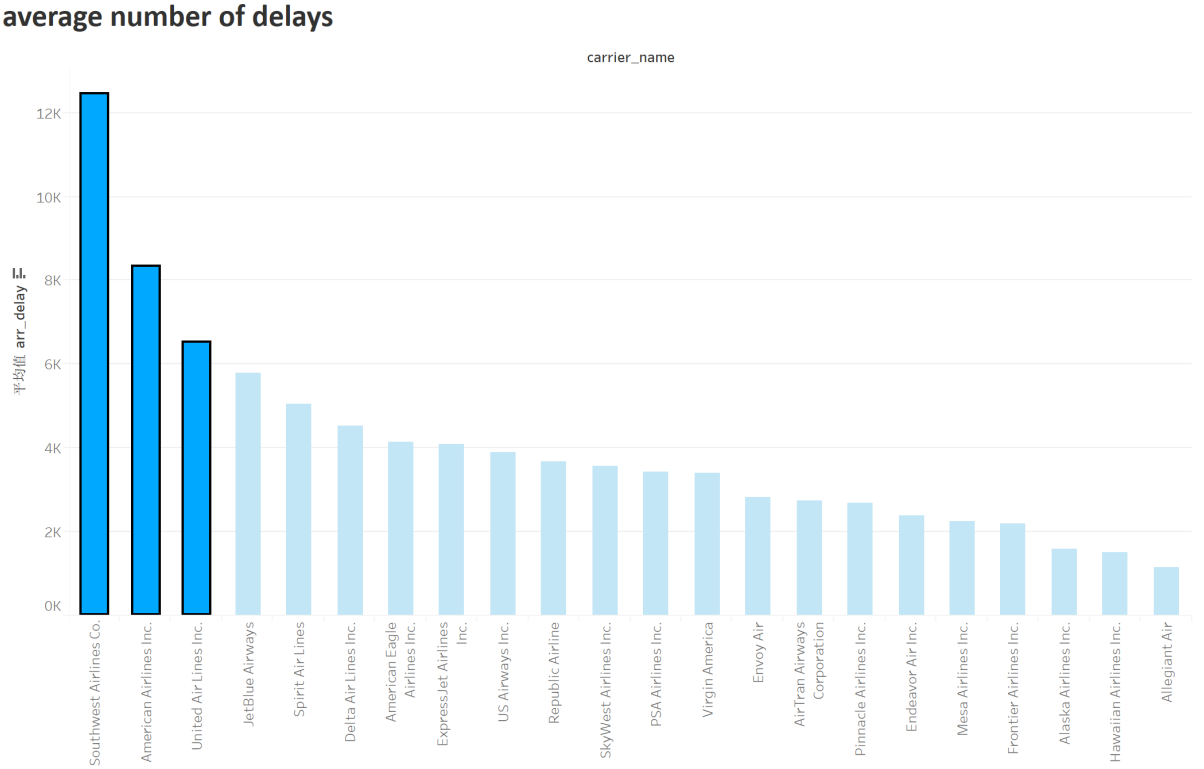
Our Raw Data:
Every row in the dataset includes 21 variables:

year	month	carrier	carrier_name	airport	airport_name	arr_flights	arr_del15	carrier_ct	weathe

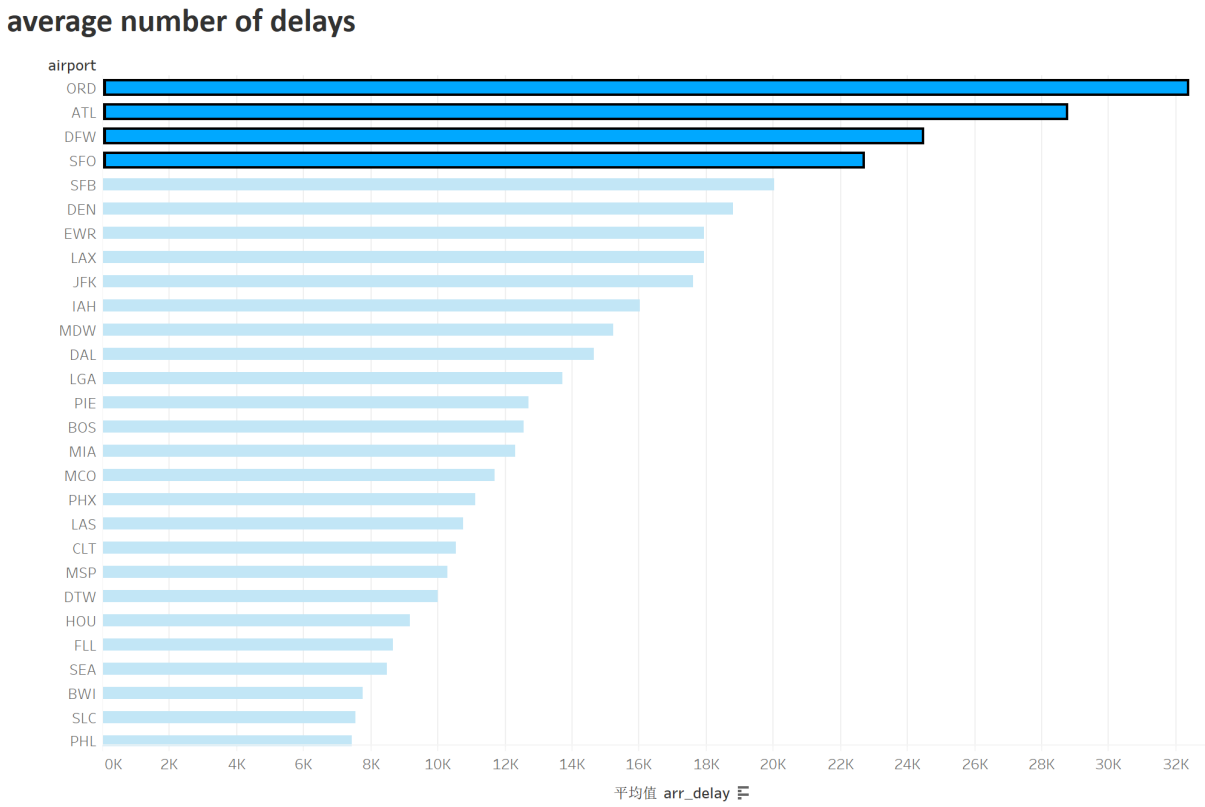
Here are some visualizations and analysis of the datasets.



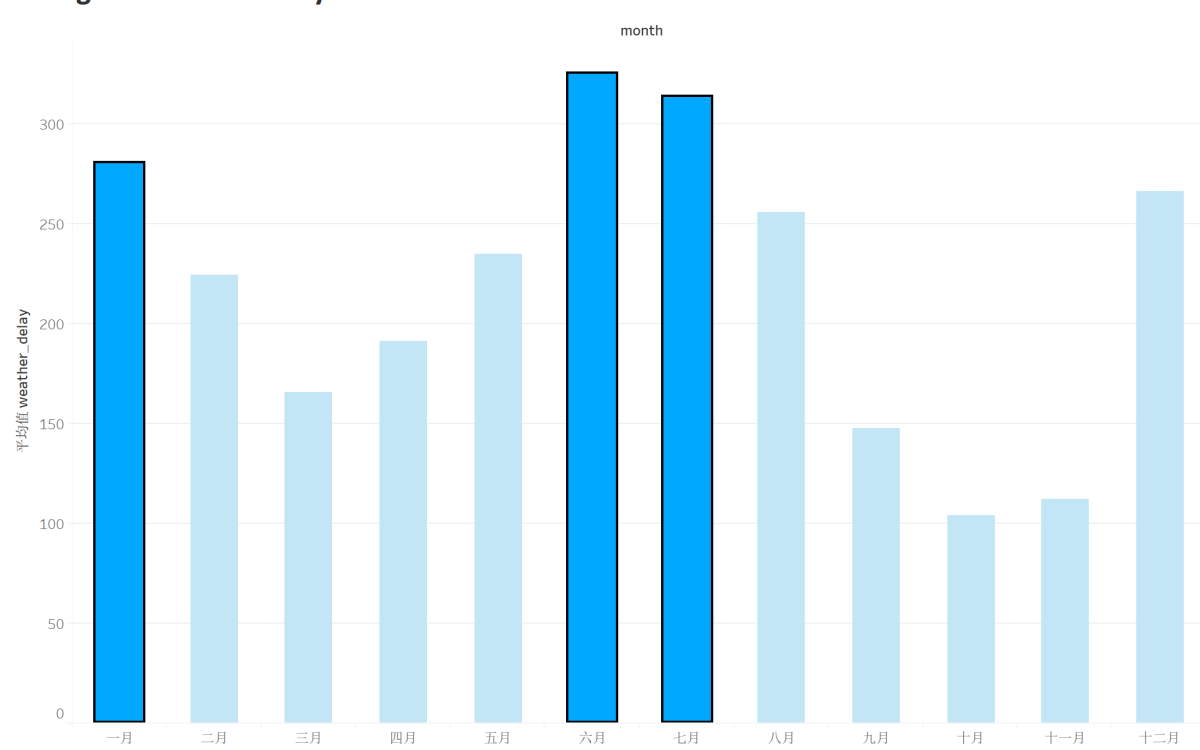
This bar chart shows the average number of delays of each month, we can know that the June July August and December are the months that delay occurs most frequently, we deduce the reason is mainly caused by the vacation.



This chart shows the average delay number of each carrier, we can see that the 3 carrier which has the highest number of delay. As expected, some airlines are better than others.

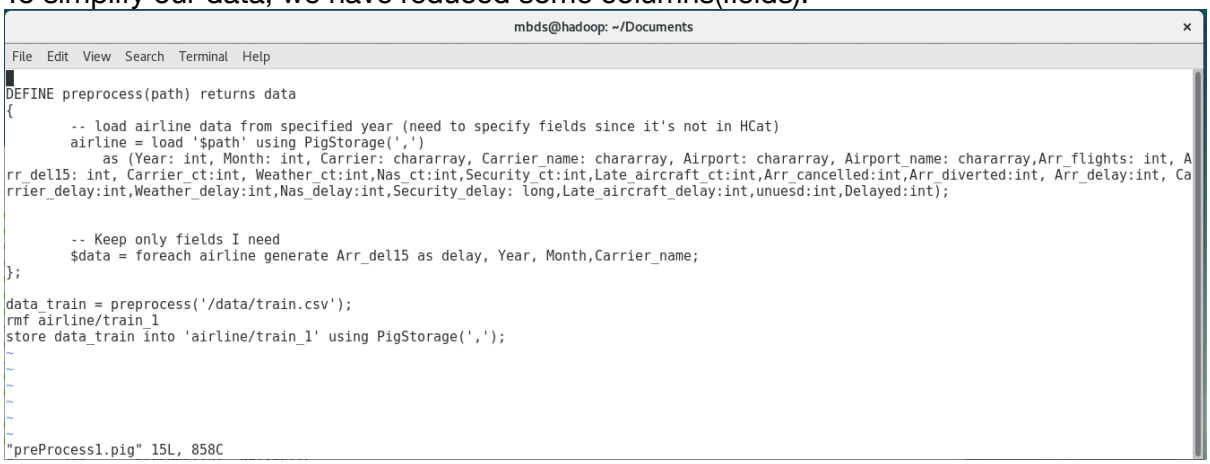


This chart shows the top 4 airports which has the highest number of delays.

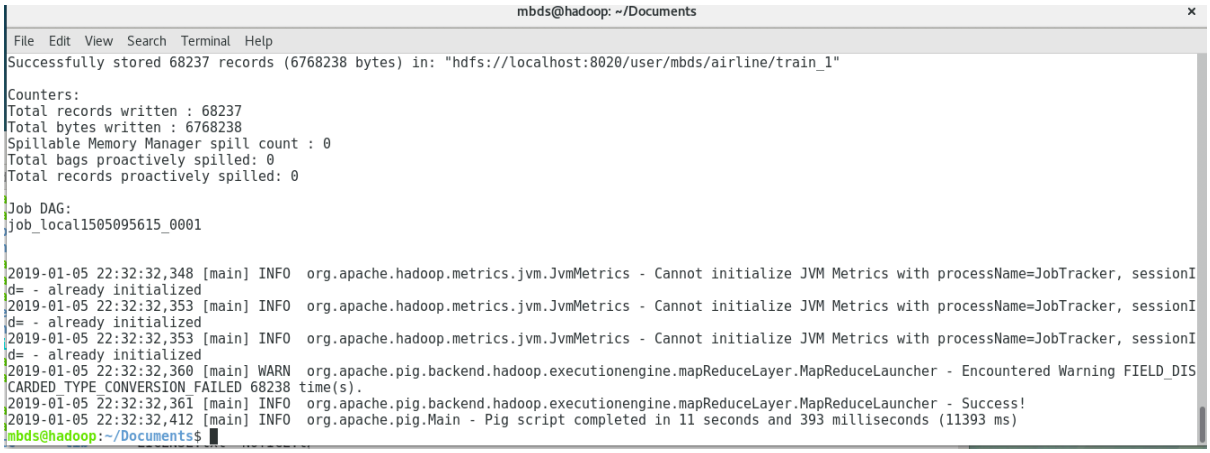


From this chart, we can find that the weather delay are highly occur on January, June and July.

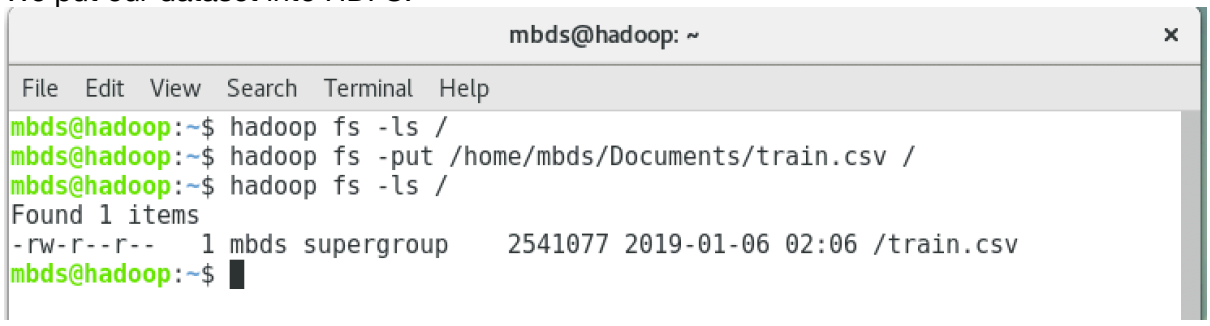
To simplify our data, we have reduced some columns(fields).



We will use “arr_del15”, “year”, and “Month”.



We put our dataset into HDFS.



Downloading the cleaned data from HDFS.
To train a available model, we split the dataset into 2 part:

- The dataset from 2013-1 to 2017-12 will be trained.
- The dataset in 2018 will be used to check and test our model.

```
df = data_2018.dropna(subset=['arr_del15'])
df['ArrDelayed'] = df['arr_del15']#.apply(lambda x: x > 15)
#df['ArrTotal'] = df['arr_flights'].apply(lambda x: x > 15)
print ("total flights:" + str(df['arr_flights'].sum()))
print ("total delays: " + str(df['ArrDelayed'].sum()))
rate = float(df['ArrDelayed'].sum())/float(df['arr_flights'].sum())
print("We can see that the rate of delay is %f" % (rate))
```

```
total flights:6033423.0
total delays: 1132251.0
We can see that the rate of delay is 0.187663
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

We found that the delay rate is under 20%.

```
mbds@hadoop:~/Documents$ python process2.py
/usr/local/lib/python2.7/dist-packages/sklearn/linear_model/logistic.py:433: FutureWarning
a solver to silence this warning.
  FutureWarning)
/usr/local/lib/python2.7/dist-packages/sklearn/linear_model/logistic.py:460: FutureWarning
ify the multi_class option to silence this warning.
  "this warning.", FutureWarning)

Confusion matrix
      0     1     2     3     4     5     6     7     8     9     ...   1003  \
0      526     0     0     0     0     0     0     0     0     0  ...      0
1       55   439    39    16     2     9     0     0     1     0  ...      0
2        0   281   114    49     6    22     0     7     2     1  ...      0
3        0   109   159   109    28    59     2    17     8     3  ...      0
4        0    17   122   135    61    79     9    45    10     4  ...      0
5        0     2    53   106    73   103    19    49    18     5  ...      0
6        0     0    18    55    77    86    48    64    29    20  ...      0
7        0     0     3    30    37    56    84    67    40    24  ...      0
8        0     0     0     9    40    37    75    64    53    32  ...      0
9        0     0     0     2    18    15    72    48    35    26  ...      0
10       0     0     0     0    10     8    50    38    37    24  ...      0
11       0     0     0     0     6     0    38    18    31    23  ...      0
12       0     0     0     0     0     2    37     9    21    15  ...      0
13       0     0     0     0     0     1    20     4    18     8  ...      0
14       0     0     0     0     0     0     9     2     6     9  ...      0
15       0     0     0     0     0     0     3     2     2     5  ...      0
16       0     0     0     0     0     0     1     0     0     2  ...      0
17       0     0     0     0     0     0     0     0     3     2  ...      0
18       0     0     0     0     0     0     1     0     0     0  ...      0
19       0     0     0     0     0     0     0     0     0     0  ...      0
20       0     0     0     0     0     0     0     0     0     0  ...      0

[1013 rows x 1013 columns]

precision = 0.12, recall = 0.12, F1 = 0.12, accuracy = 0.12
```

In order to speed up the training, I transferred the computing platform to my computer：

```
1025     0     0     0     0     0     0     0     1     0
1026     0     0     0     0     0     0     0     0     0
1027     0     0     0     0     0     0     0     1     0

[1028 rows x 1028 columns]

precision = 0.12, recall = 0.12, F1 = 0.12, accuracy = 0.12
```

We found the precision is quite low.

So we decide find another method to improve our precision.
Then we use the random forest classifier with 50 trees, the result is much more better!
accuracy =0.89

```
# Create Random Forest classifier with 50 trees
clf_rf = RandomForestClassifier(n_estimators=50, n_jobs=-1)
clf_rf.fit(train_x, train_y)

# Evaluate on test set
pr = clf_rf.predict(test_x)

# print results
cm = confusion_matrix(test_y, pr)
print("Confusion matrix")
print(pd.DataFrame(cm))
report_svm = precision_recall_fscore_support(list(test_y), list(pr), average='micro')
print ("\nprecision = %0.2f, recall = %0.2f, F1 = %0.2f, accuracy = %0.2f\n" % \
        (report_svm[0], report_svm[1], report_svm[2], accuracy_score(list(test_y), list(pr))))
```

```
985    0    0    0    0    0    0    0
986    0    0    0    0    0    0    0
987    0    0    0    0    0    0    0
988    0    0    1    0    0    0    0
989    0    0    0    0    0    0    0
990    0    0    0    0    0    0    0
991    0    0    0    0    0    0    0
992    0    0    0    0    0    0    0
993    0    0    0    0    0    0    0
994    0    0    0    0    0    0    0
995    0    0    0    0    1    0    0
996    0    0    0    0    0    0    0
997    0    0    0    0    0    0    0
998    0    0    0    0    0    1    0

[999 rows x 999 columns]

precision = 0.89, recall = 0.89, F1 = 0.89, accuracy = 0.89
```

Proccess1.py

process1.py

```
import pydoop.hdfs as hdfs
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

import sys
import random
import numpy as np

from sklearn import linear_model,  metrics, svm
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix, precision_recall_fscore_support, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler

import pandas as pd
import matplotlib.pyplot as plt

# function to read HDFS file into dataframe using PyDoop
import pydoop.hdfs as hdfs
def read_csv_from_hdfs(path, cols, col_types=None):
  files = hdfs.ls(path);
  pieces = []
  for f in files:
    fhandle = hdfs.open(f)
    pieces.append(pd.read_csv(fhandle, names=cols, dtype=col_types))
    fhandle.close()
  return pd.concat(pieces, ignore_index=True)

```

Process2

process1.py process2.py

```
#ML Process
import pandas as pd
origin = pd.read_csv("/home/mbds/Documents/train_1/part-m-00000",delimiter=",")
data_2018 = pd.read_csv("/home/mbds/Documents/airline/2018_1/part-m-00000",delimiter=",")
cols = ['arrDelay','year','month','arr_flights','arr_del15']
train_y = origin['arr_del15']>15
train_x = origin[cols]

test_y = data_2018['arr_del15']>15
test_x = data_2018[cols]


import numpy as np

from sklearn import linear_model, metrics, svm
from sklearn.metrics import confusion_matrix, precision_recall_fscore_support, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score

import pandas as pd
import matplotlib.pyplot as plt

# Create logistic regression model with L2 regularization
clf_lr = linear_model.LogisticRegression(penalty='l2', class_weight='balanced')
clf_lr.fit(train_x, train_y)

# Predict output labels on test set
pr = clf_lr.predict(test_x)

# display evaluation metrics
cm = confusion_matrix(test_y, pr)
print("Confusion matrix")
print(pd.DataFrame(cm))
report_lr = precision_recall_fscore_support(list(test_y), list(pr), average='micro')
print "\nprecision = %0.2f, recall = %0.2f, F1 = %0.2f, accuracy = %0.2f\n" % \
        (report_lr[0], report_lr[1], report_lr[2], accuracy_score(list(test_y), list(pr)))

```