# A7 Report

*Abhishek Ravi Chandran, Chinmayee Vaidya, Chintan Pathak, Mania Abdi*

*March 19, 2016*

## High Level Design:

We used ** QuickML ** library in this assignment. We have three jobs pipelined to process the data.

### Job 1

**Design:**

The first job reads the historical and test input files and separates them with the keys carrier and year. We then find all possible connections from the data.

In the mapper class we use a custom key where we write every record twice differentiating them by a flag(type). 0 for incoming flights and 1 for outgoing flights.

The logic for handling overnight flights is handled in the method where we set values to our object after parsing it.

We sort the data to make sure that all incoming flights are sent to the reducer first, followed by all the outgoing flights. This is achieved by having a custom key, custom sort comparator and a custom group comparator. By using the group comparator we ensure that each carrier and year are chosen as the natural key to send all records grouped with this key to a single reducer.

In the reducer, we put all the incoming flights into a map with the key as city(destination), so all flights incoming to city 'a' will be under the key 'a'. The records are stored in a TreeSet which will sort all the flights stored according to their scheduled arrival time.

When we start getting outgoing flights, for each record we fetch all flights incoming to the same city as the origin city of this record, by getting the records from the map. Since the values are sorted, we keep checking for connections until we reach a record where the time difference is more than 60 minutes, meaning any flight after this will not form a connection as it is outside the time window. This ensures that not all flights are not compared reducing the number of comparisons.

Conditions for a connection: A Connections is made when A.destination = B.Origin && [(B.scheduledDeparture - A.ScheduledArrival) >=30 mins && <= 1 hr]

A connection is said to be bad if flight A is cancelled or ( B.actualDeparture - A.actualArrival) < 30 mins

We write all the connections and create models for both test and historical data.

### Job 2

**Design:**

The second job uses the models to create random forest classifiers, which we serialize and persist. All the classifiers are created for every carrier and month data. We did this believing that patters are not carrier specific, and monthly patters make sense, since weather patterns are seasonal and monthly data from accross year would give us an accurate model

## Job 3

**Design:**

The third job reads the request file and the test model, at the reducer we read the serialized classifier that we need and then use it to get the predicted value(connections missed/not) for each itenary. Here as we a single request could have multiple connections, we predict if all the connections will be missed or not and then choose the best connection based on, if the connections is not going to be missed and the least layover time.

## Attributes used for classification

- day of the week(1-7)
- origin
- intermediate destination
- destination
- layover
- total elapsed time
- distance group

## Runtime

- First job - 6 minutes
- Second job - 2 minutes
- Third job - 3 minutes

When run on a cluster of 1 master 5 core machines of size m3.xlarge each.

## Output

```
## Total duration = 802625
## Number of Good connections chosen:7541
## Number of missed connections chosen:264
## Total itenaries:7805
## % of good connections chosen:96.61755%
## % of bad connections chosen:3.382447%
```