Assignment 1

(1) URL of the dataset:

White House Visitor Logs (Obama version as it has more data): https://obamawhitehouse.archives.gov/goodgovernment/tools/visitor-records

Details about the data and its usage can be found here: https://github.com/omkarprabhu-98/mining-white-house-visitor-logs/blob/master/DATA.md

(2) The software packages and tools or library you use in your programs

- Java
- HDFS & Hadoop Mapreduce
- Make

Installation and Setup instructions can be found here: https://github.com/omkarprabhu-98/mining-white-house-visitor-logs/blob/master/SETUP.md

(3) The open source code you leveraged in developing your visitor log miner

Created a personal project from scratch: https://github.com/omkarprabhu-98/mining-white-house-visitor-logs

(4) The experiments you conducted to report your mining accuracy and time, ideally over three different datasets in three different sizes (e.g., 2x, 5x, 10x of your initial dataset.)

Can be found here: https://github.com/omkarprabhu-98/mining-white-house-visitor-logs/blob/master/Experiments_and_Results.pdf

Accuracy is not reported as it was verified with a smaller dataset that the program behave correctly.

(5) Analysis of your hand-on experiences, with three observations you wish to make

Analysis:

1. Time:

   The time taken by mappers does not increase to the scale at which data increases, this may be because the no. of mapper changes (goes up) as the the data scales. But interestingly, if we reduce the no of mappers (increase the split size) the time take reduces than in the run with more no of mappers (for the same dataset). This could be because of the contention between mappers for memory and cpu resulting in added wait times.

   For Application 1:

   Looking at the time mapper and reducer jobs time it seems like reducers take really less time and don't actually increase (in the second job). But when thinking about the application and the records, the reducer is only just collecting 10 records from each mapper so the processing size for it is relatively quite small.

   For Application 2:

The time taken by reducers doesn't change relatively as much as mappers, this was interesting as here we are not doing top 10, but still the input records to the reducers are a lot less (due to grouping by months and removal of unclean data) than the input to mappers.

2. Data

   A large no. of records are ignored since the data does not fit the specification (either no. of columns are not correct or eg. the date format is not consistent). This is clearly visible from the Application 2 as we have a lot of NULL-NULL records as month-year combination.

Some observations from doing the experiments:

Observation1: Scripts help automate a lot of manual work. The makefile I created helped run experiments quickly and clean state. CLI's provided by Hadoop help to run things smoothly and start over in case services fail.

Observation2: Monitoring and metric collection helps aid in debugging and understanding what's actually happening when running on large amounts of data. I was curious to see reducer time not increasing as much as map time, but when I saw the no. of input records to reducers it was obvious.

Observation3: No data is clean, always clean the data and always expect unclean data as input when coding. A large amount of data was unusable eg. dates were not in proper format so had to add code to discard it.