**Wiki species – xml format**

The task was to run the wikispecies program by inputting the xml file on fully distributed mode.

I ran the iterator code for 21 iterations in the loop.

I used the value of seed as 0.5 in the program.

**Issues faced:**

1. The ip address of the virtual machine was changed due to change in network. Had to configure “/etc/hosts” file again.
2. When starting the namenode and datanode. The datanode on the slave machines were not starting. After checking the log on the slave machine, found out that the error was of “Incompatible namespaceids”. Researched on the error then applied the solution where I deleted the “VERSION” file in “/app/hadoop/tmp/dfs/data”.

**Bugs removal:**

The code doesn’t contain any bugs that professor earlier injected.

* 1. Seed modification

One can land on a web page broadly via two methods - out links and in links or other factors like directly typing the URL. Page Rank algorithm has a seed factor to cater the second method. In our program I changed the seed in the Reducer code of Builder and Iterator code of Mapper. The value derived here is by using the n/m formula where m is the total no of pages

* + 1. Builder Reducer

Changed the type of count variable to double.

And increment it by 0.1 inside the loop

* + 1. Iterator Mapper

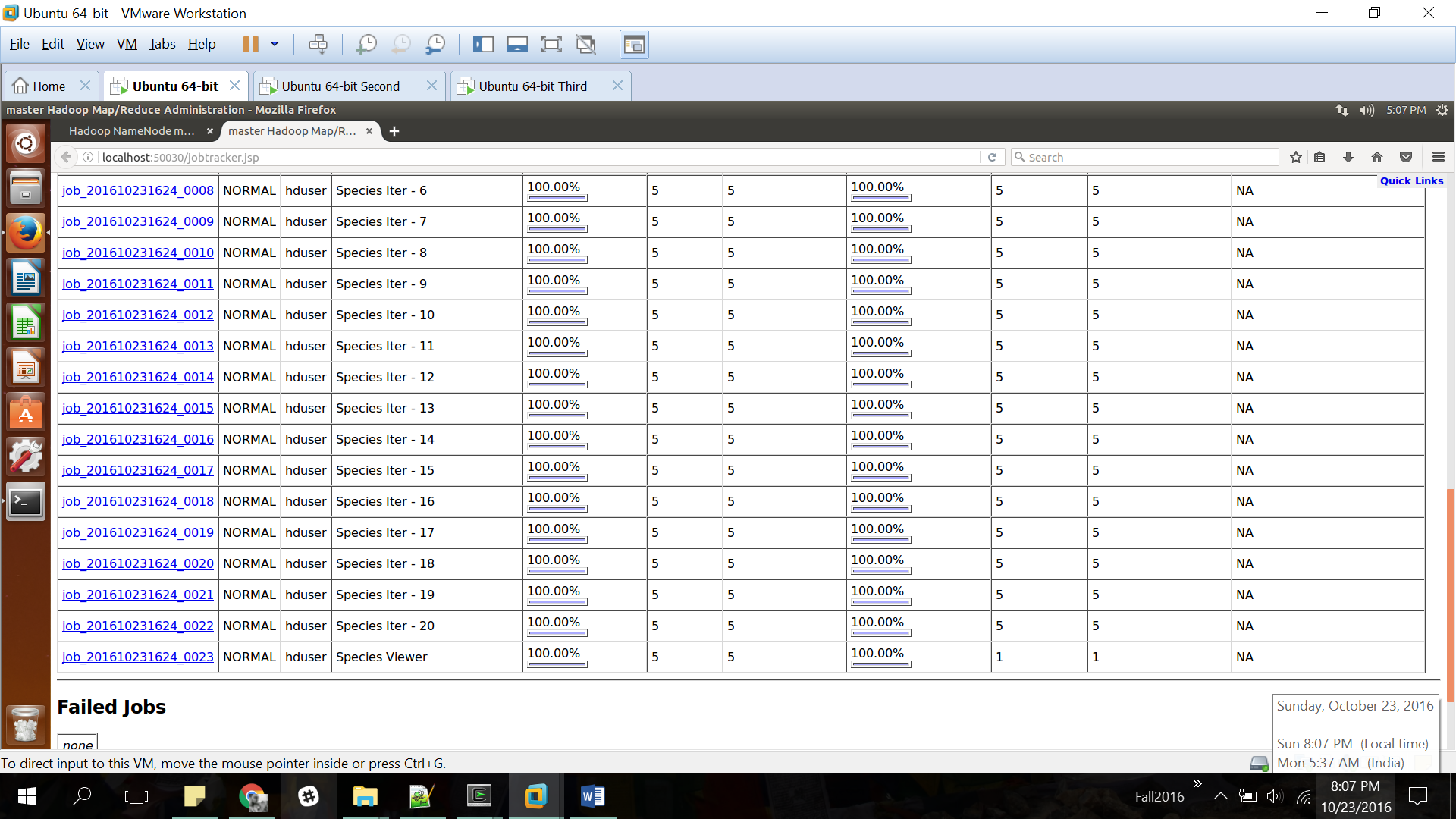
Change the value of the variable *currscore* to 0.5;

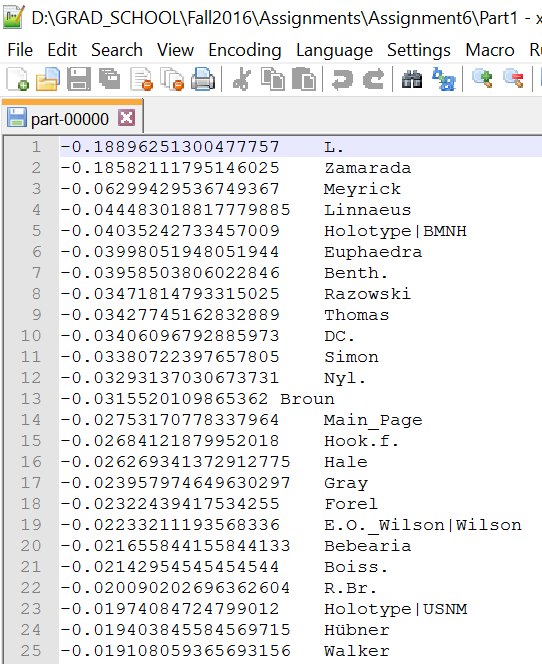
* 1. Other improvements

Changed the code of Builder Mapper to remove the colons and replace them with underscore.

* 1. Other Improvements

Removed the code line that outputs for better results - *new Text(toEmit + ": " + pagetitle)*





**Winnow Algorithm:**

**Dataset:**

The dataset I chose represents a set of possible advertisements on Internet pages. The features encode the geometry of the image (if available) as well as phrases occurring in the URL, the image's URL and alt text, the anchor text, and words occurring near the anchor text.

**The task** is to predict whether an image is an advertisement ("ad") or not ("non-ad").

**Number of features**: 1558, in which 3 are continuous and others are binary.

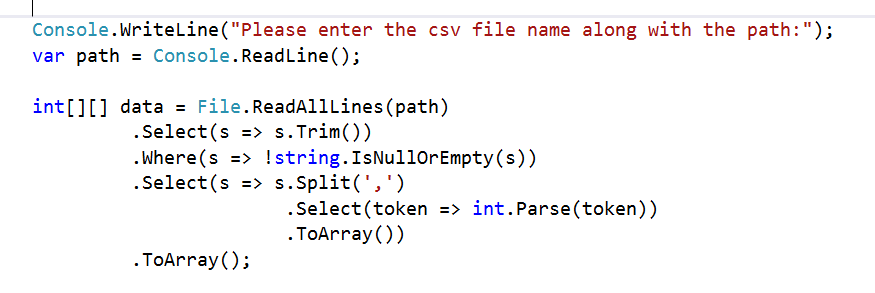
**Feature Engineering:**

* I removed the continuous variables and worked on the remaining 1555 variables.
* 0.5% of the values were missing, Hence, substituted the missing values with the binary digit 0

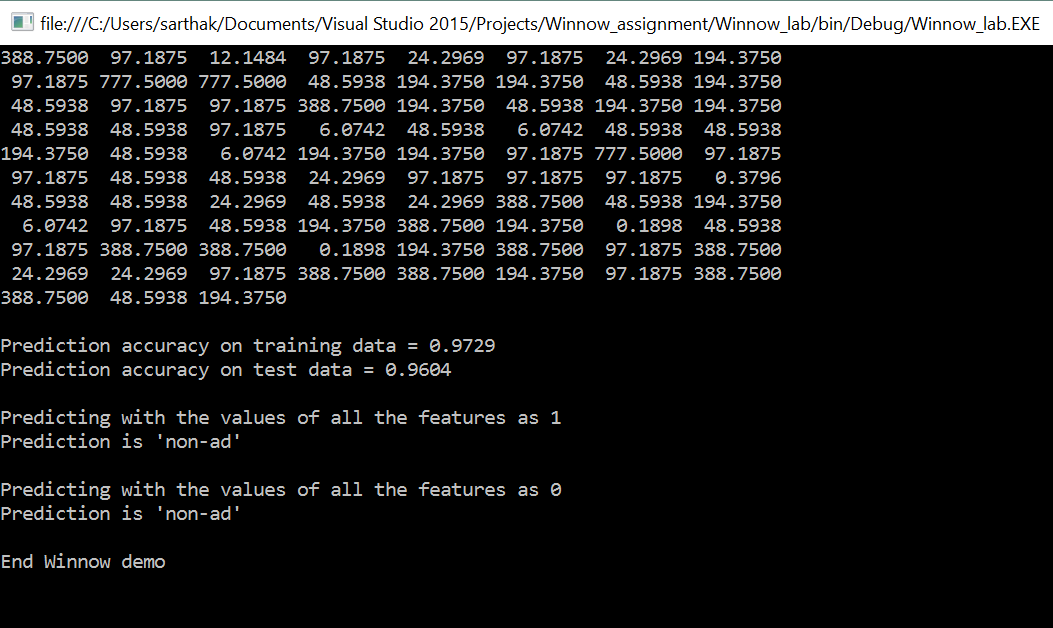
**Total number of instances**: 3279

**Code and Results:**

Since the csv was pretty large, as every row contained 1556 features. It was not efficient to hardcode the data in the code. I added the code where user needs to input the csv file with its path on which Winnow algorithm needs to be run.



Since all the original data is in binary format and there are a lot of features on which Winnow can work therefore the results were pretty good with Accuracy on training data of 97% and on test data of 96%.



**What’s the dataset size after which the algorithm becomes unwieldy?**

The amount of data on which the algorithm becomes unwieldy is a very open ended discussion since it depends on the machine configuration on which the program is running and also the code can be optimized to handle a reasonable amount of data. For example: “Int” can be converted into “short int” to store 16-bit integers in less amount of space as compared to int.

I tried running data which had 100000 rows and 16 variables on my 8GB RAM laptop and did not find any difficulty in running it. I am not submitting the result of that data since the accuracy of that was less than 50% even after I changed the categorical variables into 3 and 4 buckets and stored them into 3 columns as 001, 010 and 100.

Anyways while working on this dataset I learned that this algorithm does not work great if the data contains a lot of continuous variables because even after converting them into categorical variables by breaking them into buckets suing 1-N encoding, there is always high probability that there would not be enough categories and hence the prediction accuracy would be low.

**Improvements:**

1. One of the improvement that can be made is that the data should be shuffled before splitting. I shuffled the data before splitting it into train and test and observed that the accuracy was increased significantly. In the current code, shuffling happens on the training data after the split.
2. Currently, the number of features in the data needs to be hardcoded. The number of features should be read automatically.
3. The data should not be hardcoded and should be read from a file. It is a few lines of code using lambda expression and I have included that in my code.