Text Analytics
Assignment 2

Joseph Pereira
Yuanyi Shen
Sahil Batra
Neerav Basant
Ketan Mudda
Xiaoyu Sun

Task A. Ignore the text (reviews) and run a classification model with the numeric data (you can use standard methods like logistic regression or k-nearest neighbors). What is the accuracy of your model?

The model has an accuracy of 68.5% when a threshold of 50% is used to classify how the predictor determines whether to assign to a 'low' or 'high' value using the three quantitative values of Votes_Cool, Votes_Funny, Votes_Useful. Other variables were ignored such as the type of food the restaurant serves as these are binary responses, not numeric data.

The model can easily tell whether a restaurant has a high review with 98% accuracy. However, it struggles to restaurants that have 3 stars or below. It fails at a rate of 95% in this circumstance.

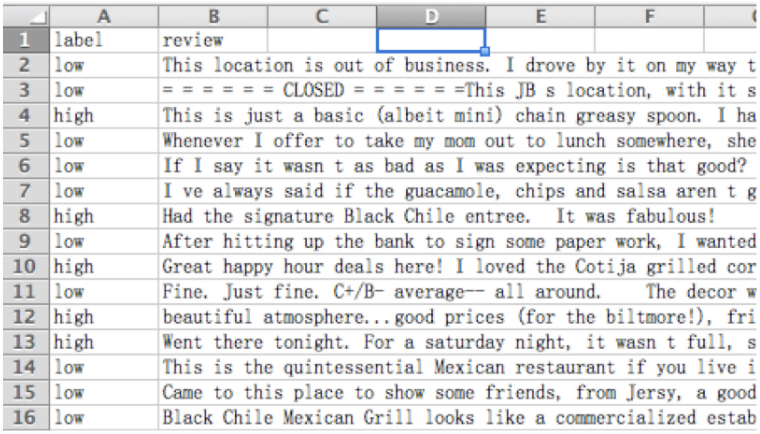
```
yelp=read.csv('yelp.csv')
attach(yelp)
str(yelp)

#Reviews with stars of 3 and below are considered to have a low review, those 4 or 5 are considered high.

yelp$highlow=cut(yelp$star, breaks = c(0,3,5), labels =c("low","high"))
yelp2 = data.frame("Review"=yelp$highlow, "Funny"=yelp$votes_funny, "Useful"=yelp$votes_useful,
"Cool"=yelp$votes_cool)

#create a logistic model to predict a high/low rating using the number of votes that were considered cool, funny, #and
useful.
yelp_logistic = glm(yelp$highlow~votes_cool+votes_funny+votes_useful, data=yelp, family='binomial')
test_pred = predict(yelp_logistic, yelp2, type='response')
test2 = rep('low', length(test_pred))

# a 50% confidence interval is used to set the threshold on whether we classify as low or high.
test2[test_pred > .5] ='high'
table(yelp2$Review,test2)
summary(yelp_logistic)
```

```
> table(yelp2$Review,test2)
      test2
        high   low
  low   6126   320
  high 13373   180
```

```
> summary(yelp_logistic)

Call:
glm(formula = yelp$review ~ votes_cool + votes_funny + votes_useful,
    family = "binomial", data = yelp)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0219  -1.4154   0.8363   0.8608   2.7264

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.80193    0.01900   42.20   <2e-16 ***
votes_cool    0.52088    0.02256   23.09   <2e-16 ***
votes_funny  -0.25794    0.01787  -14.43   <2e-16 ***
votes_useful -0.22609    0.01509  -14.99   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 25143  on 19998  degrees of freedom
Residual deviance: 24517  on 19995  degrees of freedom
  (25973 observations deleted due to missingness)
AIC: 24525

Number of Fisher Scoring iterations: 4
```

Task B. Perform a supervised classification on a subset of the corpus using the reviews. It will be best to use WEKA or LightSIDE for this purpose (WEKA will be better because I will shortly ask you to perform clustering on text, which can't be done in LightSIDE). What accuracy do you get from this text mining exercise? Explain what you did: e.g., TF-IDF scores, Naïve Bayes, etc. What is the best overall accuracy you got? What was the best accuracy for identifying highly rated restaurants?

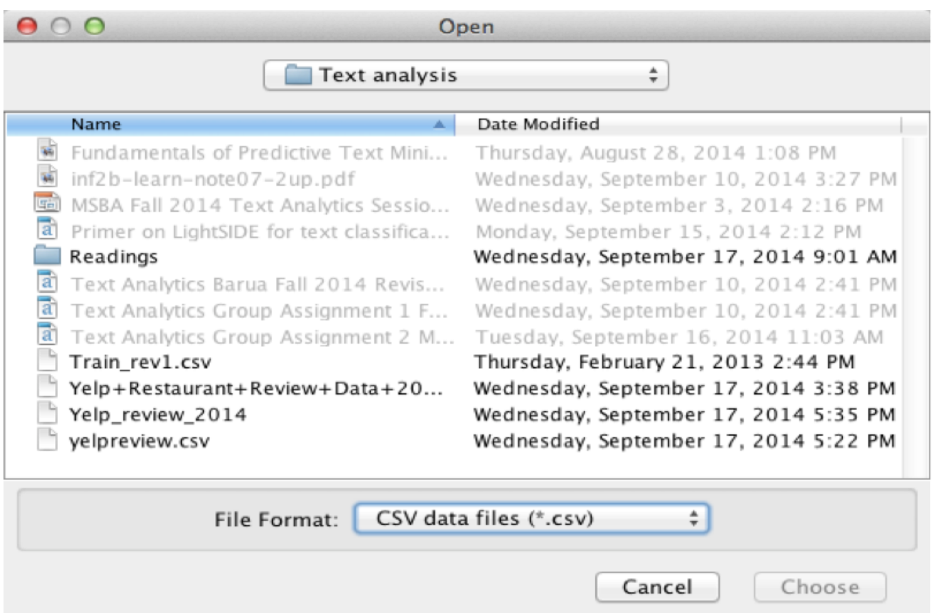
Below is the subsetted dataset that we would be using for this analysis:



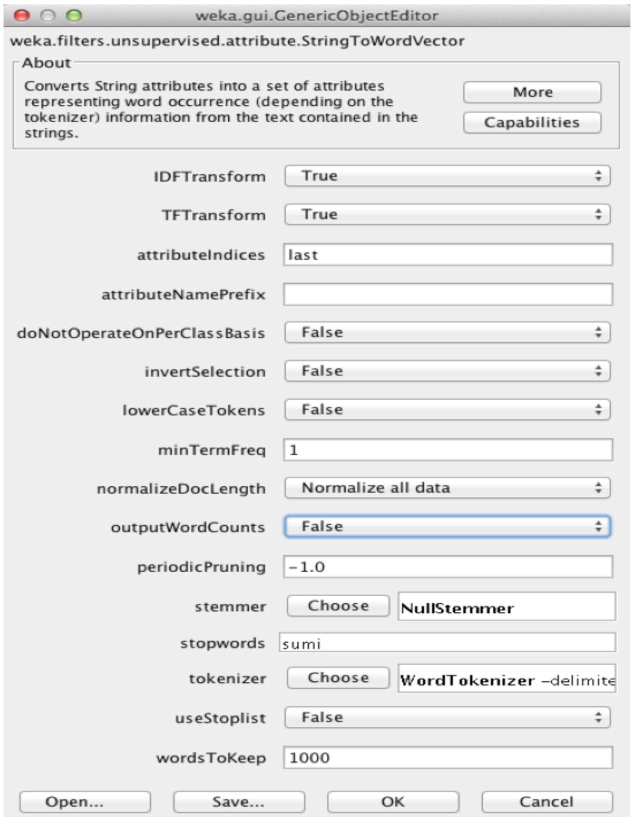
	A	B
1	label	review
2	low	This location is out of business. I drove by it on my way t
3	low	= = = = = = CLOSED = = = = = =This JB s location, with it s
4	high	This is just a basic (albeit mini) chain greasy spoon. I ha
5	low	Whenever I offer to take my mom out to lunch somewhere, she
6	low	If I say it wasn t as bad as I was expecting is that good?
7	low	I ve always said if the guacamole, chips and salsa aren t g
8	high	Had the signature Black Chile entree. It was fabulous!
9	low	After hitting up the bank to sign some paper work, I wanted
10	high	Great happy hour deals here! I loved the Cotija grilled cor
11	low	Fine. Just fine. C+/B- average-- all around. The decor w
12	high	beautiful atmosphere...good prices (for the biltmore!), fri
13	high	Went there tonight. For a saturday night, it wasn t full, s
14	low	This is the quintessential Mexican restaurant if you live i
15	low	Came to this place to show some friends, from Jersy, a good
16	low	Black Chile Mexican Grill looks like a commercialized estab

Preprocess:

1. Import the data to WEKA:



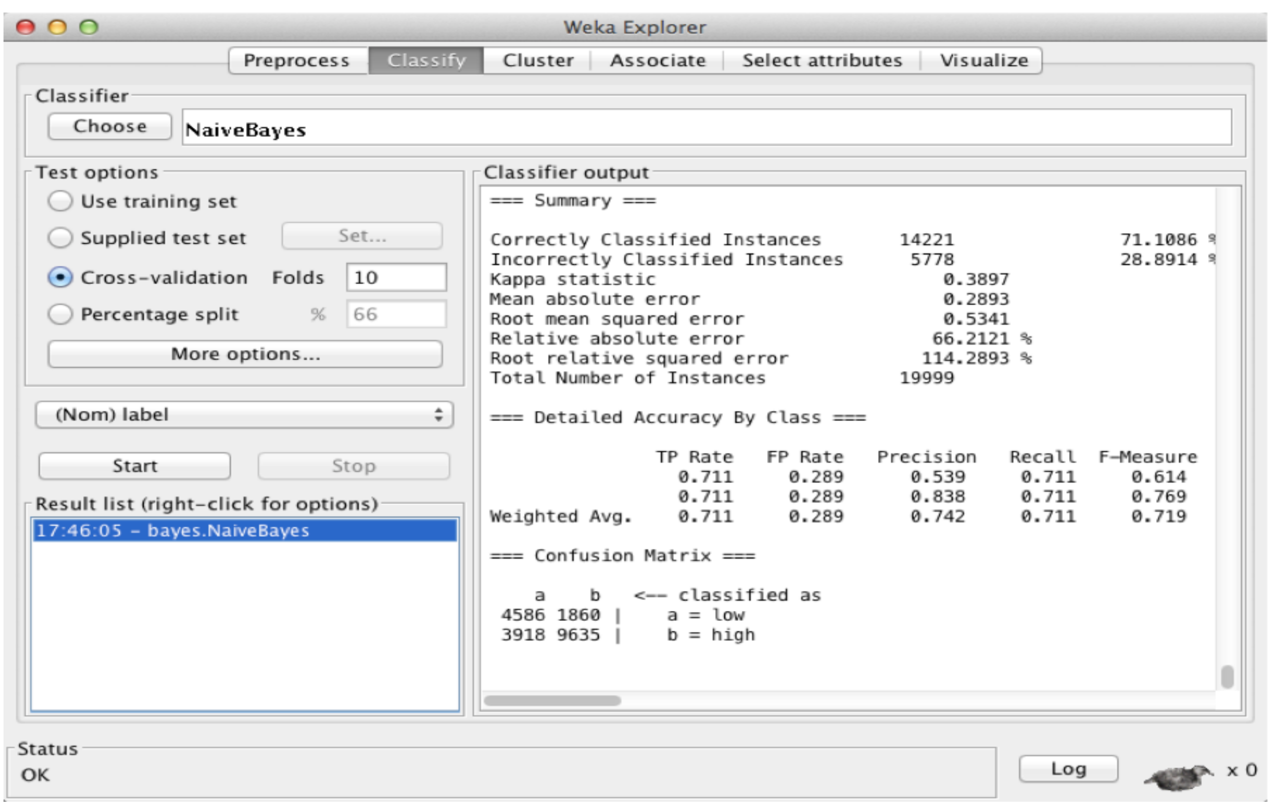
2. Select review-->Click on Button "Choose" under Filter-->Unsupervised--> Nominaltostring-->apply
3. Select review-->Click on Button "Choose" under Filter-->Unsupervised--> Stringtowordvector-->apply



4. Select label--> Click on Button “Choose” under Filter-->Unsupervised--> Reorder-->Click on textbox near it and change the order to “2-last,1”-->apply

Classification:

1. **Naïve Bayes:**



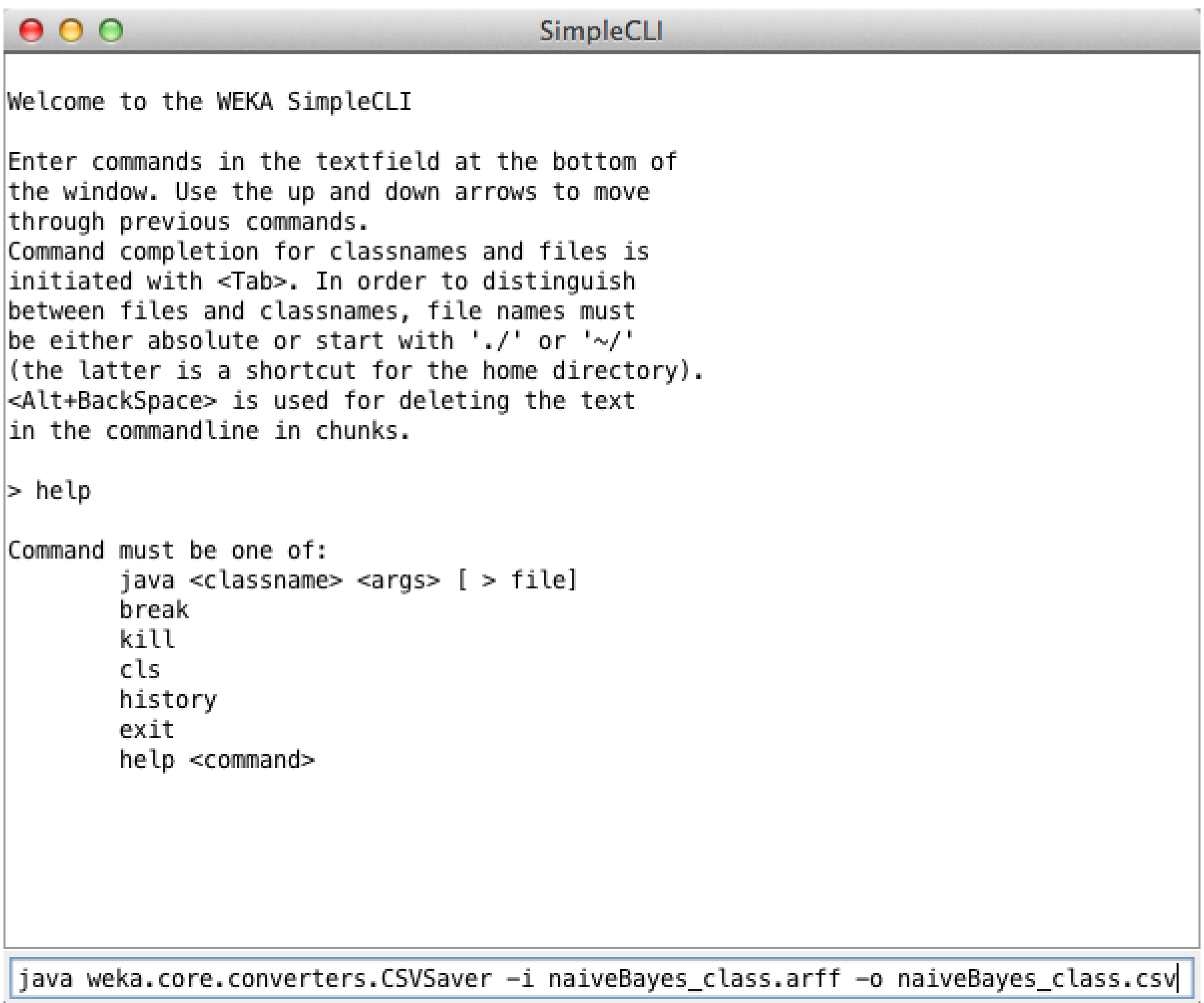
Using this model, we can get an overall accuracy of 71%. This model predicts ‘high’ with 83% accuracy, whereas predicting ‘low’ is almost like a guess.

a. When finished, right-click the model name from the Result List. Click on “Visualize classifier errors.”

b. Click “Save” in that new window and the outputted file will have the new predicted column. Name it as “naiveBayes_class.arff” in the root file(for easy sake)

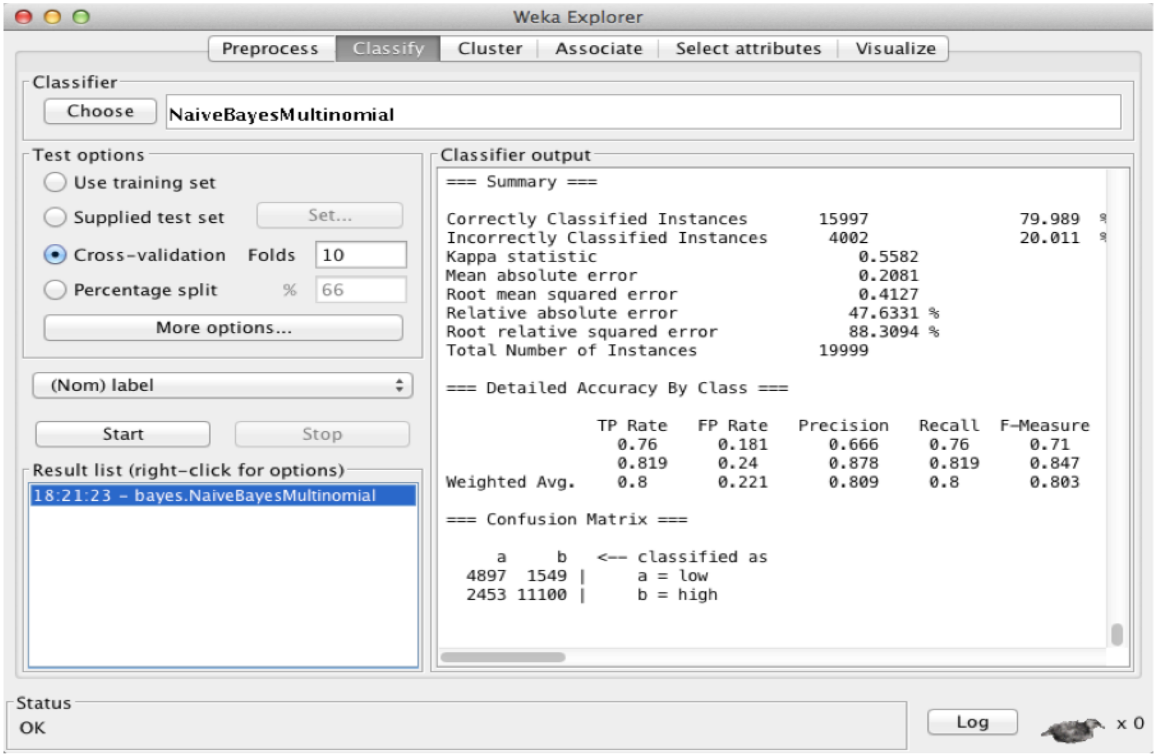
c. Find Simple CLI in beginning page

d. Type “java weka.core.converters.CSVSaver -i naiveBayes_class.arff -o naiveBayes_class.csv”



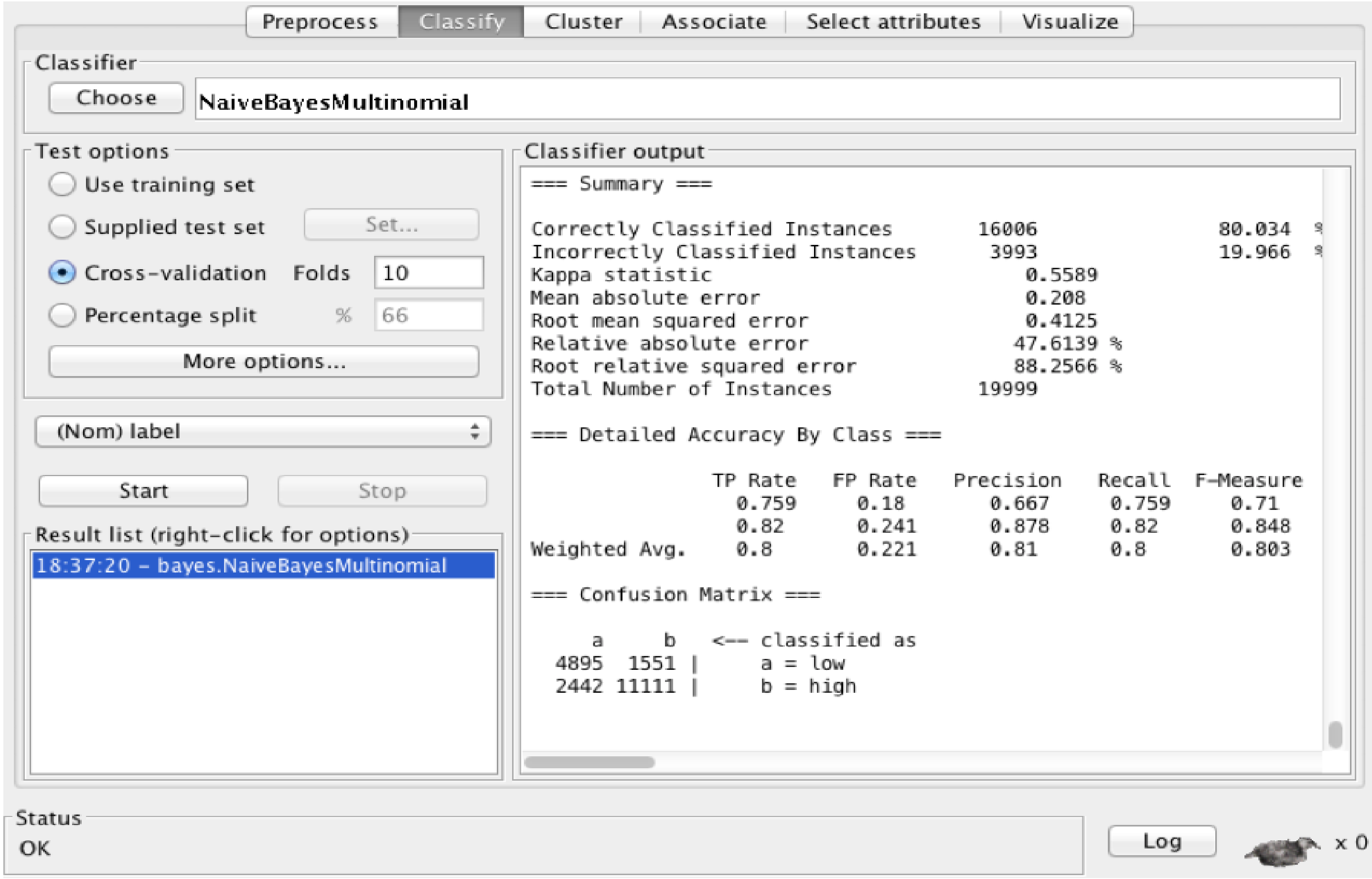
e. Then find the new file named naiveBayes_class.csv, all top 1000 word have it's value based on the existence of a certain review, and the last two columns are it's true value and predict value. Copy the predict value to our original csv file, name this column for "predictedlabel_NaiveBayes"

2. Naïve Bayes Multinomial:



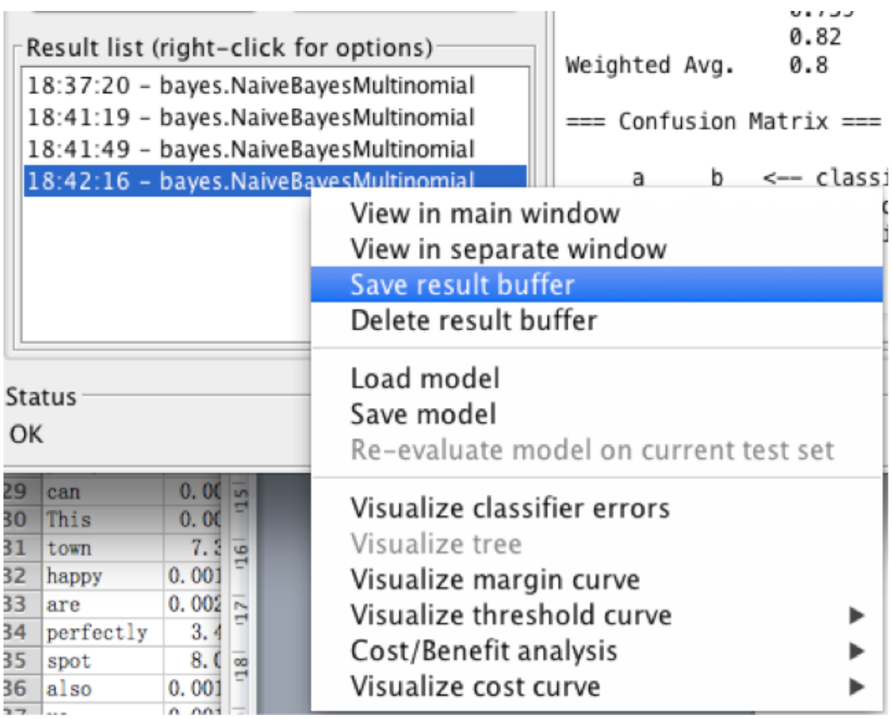
As it shows, using Naïve Bayes Multinomial method, we acquire a better accuracy of almost 80%. This model also increased the accuracy of predicting 'low' from 54% to 66%.

After we removed several stopwords "-" "=" "--", the overall accuracy increased to 80.034%.



3. Top 10 words for High predict:

a) I save the result buffer:



b) Copy the words to an excel file:

result

```
=== Classifier model (full training set) ===

The independent probability of a class
--------------------------------------
low     0.32233388330583473
high    0.6776661166941653

The probability of a word given the class
-----------------------------------------
        low      high
$10     5.411757857347446E-4     4.198905251737754 7E-4
$3      3.8922823051096734E-4    4.231637886554049E-4
$5      4.928269980954806E-4     7.00834397077962 1E-4
00      5.147881026052575E-4     3.880838699276663 7E-4
1       9.851812535126877E-4     6.310489717798874E-4
10      0.0010720497825744285    6.291648226653852E-4
15      6.789596258508484E-4     3.5049969006589537E-4
2       0.0017961685404851546    0.0011657205798 68153
20      7.797783488995103E-4     3.741133575567811E-4
3       0.0018208242543382258    0.0010689960543689818
30      8.103760446026688E-4     5.238312535327879E-4
4       0.0011718239997573763    0.001180103787813 83
5       0.0013481564994972611    0.0014176144879755035
50      6.114313032455909E-4     4.541720178886 28E-4
6       6.176087684575336E-4     5.877029756400804E-4
7       4.737525858727547E-4     3.4906545085924075E-4
8       5.21995325994832E-4      4.227383703325413E-4
A       0.0014193125398179328    0.0016383676551468 14
AZ      2.841333544451633E-4     6.103002768212331E-4
```

c) Create a new column for the High-Low; Sort it by decrease: We can find the top 10 words are all very positive

	A	B	C	D
1	The probability of a word given the class			
2	---			
3		low	high	Determint
4	Great	5.79E-04	0.00317855	2.60E-03
5	delicious	8.91E-04	0.0029546	2.06E-03
6	great	0.00196768	0.0038821	1.91E-03
7	amazing	6.24E-04	0.00246742	1.84E-03
8	love	0.00123995	0.00305563	1.82E-03
9	favorite	6.98E-04	0.0023292	1.63E-03
10	always	0.00129876	0.00283463	1.54E-03
11	best	0.00159336	0.00302716	1.43E-03
12	awesome	6.69E-04	0.00208198	1.41E-03
13	friendly	0.00139683	0.00277573	1.38E-03
14	perfect	4.28E-04	0.00177132	1.34E-03
15	excellent	6.78E-04	0.00200011	1.32E-03
16	Love	2.19E-04	0.00138457	1.17E-03
17	Phoenix	0.00128308	0.0024458	1.16E-03
18	fresh	0.00124591	0.00232735	1.08E-03
19	atmosphere	0.00127153	0.00235111	1.08E-03
20	Best	1.81E-04	0.0012197	1.04E-03
21	fantastic	4.34E-04	0.00144766	1.01E-03
22	wonderful	4.57E-04	0.00142704	9.70E-04
23	recommend	8.12E-04	0.0017278	9.16E-04
24	definitely	0.00113215	0.00203335	9.01E-04
25	staff	0.00153634	0.00242866	8.92E-04
26	loved	4.99E-04	0.00136688	8.68E-04
27	every	8.78E-04	0.00170821	8.30E-04
28	yummy	5.06E-04	0.0013359	8.30E-04

Task C. Combine the numeric data and the text classification model (in task B) to create a “hybrid” model. It is your task to figure out how to do this. Now run this hybrid classification model and compare the results with those in A and B. Explain and justify how you created a mixed model with both numbers (e.g., price ranges, type of food) and text (the reviews). Did you get better results? Discuss the implications of these “hybrid” models based on your “best” attempt.

To build a hybrid model, we regard the result from Task A as a dummy variable and combine it with original numerical data. To do this, we combine the text predicted result with original numerical data in excel. Then we use the new data to fit a model. The result we use is the binomial model result in Task A, which had an accuracy of 71%.

We use logistic regression to build the model. As in Task A, we choose “votes_funny”, ”votes_useful”, “votes_cool” and the prediction from text as variables and implement it using R.

```
# Read the combined new data
yelp=read.csv("C:\\yelp_refined.csv")
# change the type of text predicted result to categorical.
yelp[,21]=as.factor(yelp[,21])
#Reviews with stars of 3 and below are considered to have a low review, those 4 or 5 are considered high.
yelp$rate=cut(yelp$stars, breaks = c(0,3,5), labels =c("low","high"))
#create a logistic regression model using votes_cool, votes_funny, votes_useful and text predicted value as variables.
yelp_lm = glm(yelp$rate~votes_cool+votes_funny+votes_useful+text_pred, data=yelp, family = 'binomial')
yelp2 = data.frame("Review"=yelp$rate, "Funny"=yelp$votes_funny, "Useful"=yelp$votes_useful,
"Cool"=yelp$votes_cool, 'text_pred'=yelp$text_pred)
test_pred = predict(yelp_lm, type='response')
test2 = rep('low', length(test_pred))
# use 50% confidence interval to set the threshold for high and low
test2[test_pred > .5] ='high'
prop.table(table(yelp2$Review,test2),2)
```

The output of logistic regression is:

```
Call:
glm(formula = yelp$rate ~ votes_cool + votes_funny + votes_useful +
    text_pred, family = "binomial", data = yelp)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2775  -1.0448   0.5934   0.6517   2.0863

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.11408    0.02563  -4.452 8.51e-06 ***
votes_cool    0.48652    0.02397  20.298  < 2e-16 ***
votes_funny  -0.21025    0.01835 -11.461  < 2e-16 ***
votes_useful -0.20616    0.01635 -12.610  < 2e-16 ***
text_pred1    1.76170    0.03398  51.846  < 2e-16 ***
---
Signif. codes:  0 ?***?0.001 ?**?0.01 ??0.05 ??0.1 ??1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 24845  on 19738  degrees of freedom
Residual deviance: 21269  on 19734  degrees of freedom
AIC: 21279
```

Confusion matrix is:

```
> prop.table(table(yelp2$Review,test2))
      test2
            high       low
  low  0.1282740 0.1950453
  high 0.5294594 0.1472212
```

The overall accuracy of the hybrid model is 72.4%, increased 1.4% comparing to text mining result and 3.9% comparing to numerical data regression result.

The accuracy for predicting high score review is 80.5% and the accuracy for predicting low score review is 57%.

Task D. Use unsupervised sentiment analysis (with SentiStrength or any other tool) and use the sentiment score to predict high/low rating. Compare and contrast the results **of tasks B and D.**

Using SentiStrength, we got the following results :

Confusion Matrix

		Actual		
		Low	High	Total
Predicted	Low	3647	2924	6571
	High	2799	10629	13428
	Total	6446	13553	19999
Overall accuracy		**71.38%**		
Accuracy for Highly Rated Restaurants		**78.43%**		

SentiStrength gives us an output in the form of positive and negative emotion ratings on a scale of +1 to +5 and -1 to -5 respectively, for each line of text (in this case, restaurant review). We have added these two ratings and assigned a high rating to the restaurant if the total emotion rating is greater than or equal to +1. Using this methodology, we achieved an overall accuracy of 71.38% and an accuracy of 78.43% for highly rated restaurants. This overall accuracy is marginally higher than the accuracy of 71.11% gained by the Naive-Bayes method and significantly lower than the 79.99% gained by the Naive-Bayes Multinomial method (as illustrated in Part B). However, when it comes to rating the highly rated restaurants, the accuracy gained with SentiStrength is much lower than the 83% gained by Naive Bayes and 87.8% gained by Naive Bayes Multinomial methods.

Please find below a snapshot of the SentiStrength ratings used:-

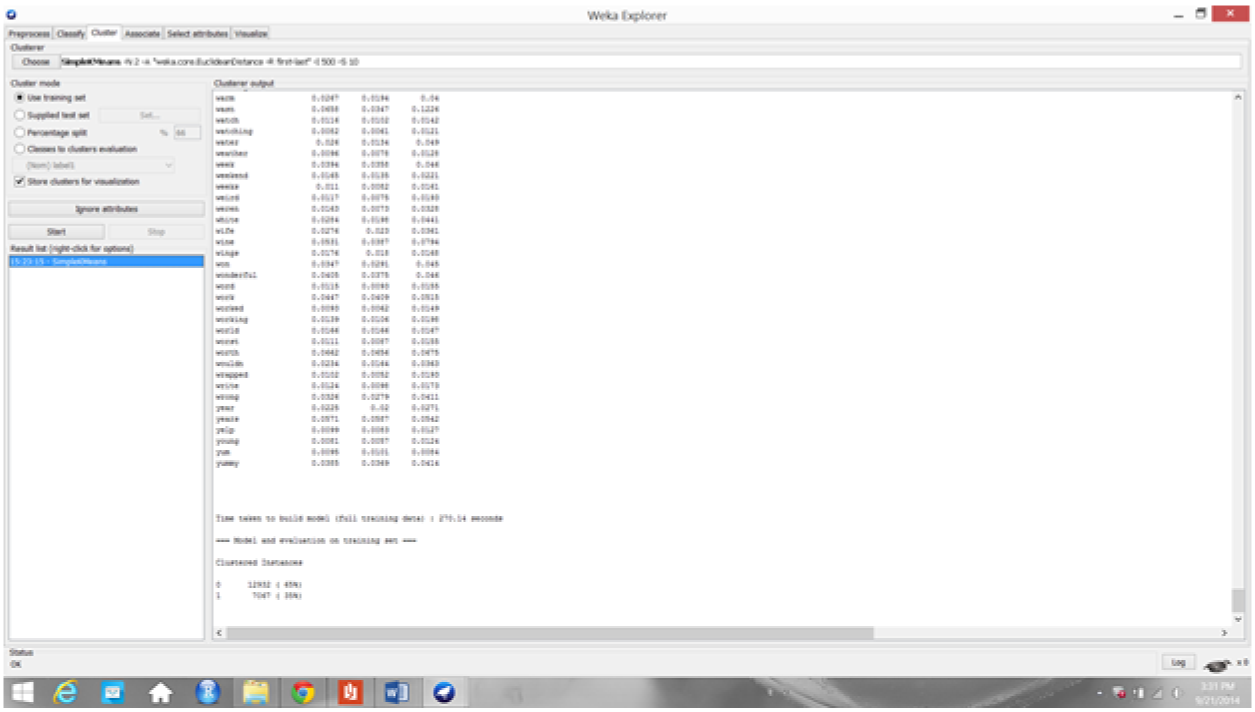
Actual Rating	Actual Rating Level	Review	Positive Emotion	Negative Emotion	Total Emotion	Senti Rating	Actual VS Senti
low	0	This locati	1	-1	0	0	1
low	0	CLO	2	-2	0	0	1
high	1	This is just	2	-2	0	0	0
low	0	Whenever	4	-4	0	0	1
low	0	If I say it w	3	-3	0	0	1
low	0	I ve always	3	-3	0	0	1
high	1	Had the sig	3	-1	2	1	1
low	0	After hittir	4	-4	0	0	1
high	1	Great happ	4	-5	-1	0	0
low	0	Fine. Just f	4	-3	1	1	0

Task E. Use unsupervised clustering on the text (use the distance measures available in WEKA). Does clustering achieve satisfactory separation between high and low rated restaurants?

We prepared the dataset for clustering by following the preprocessing steps from Task B. Then we built the clusters using K means clustering with Euclidean Distance for various values of K.

1. **K = 2:**

   Clustering Output:



Distribution of ‘low’ and ‘high’ in each cluster:

Rating	Cluster 0	Cluster 1	Total
high	9472	4081	13553 (68%)
low	3460	2986	6446 (32%)
Grand Total	**12932 (65%)**	**7067 (35%)**	**19999**

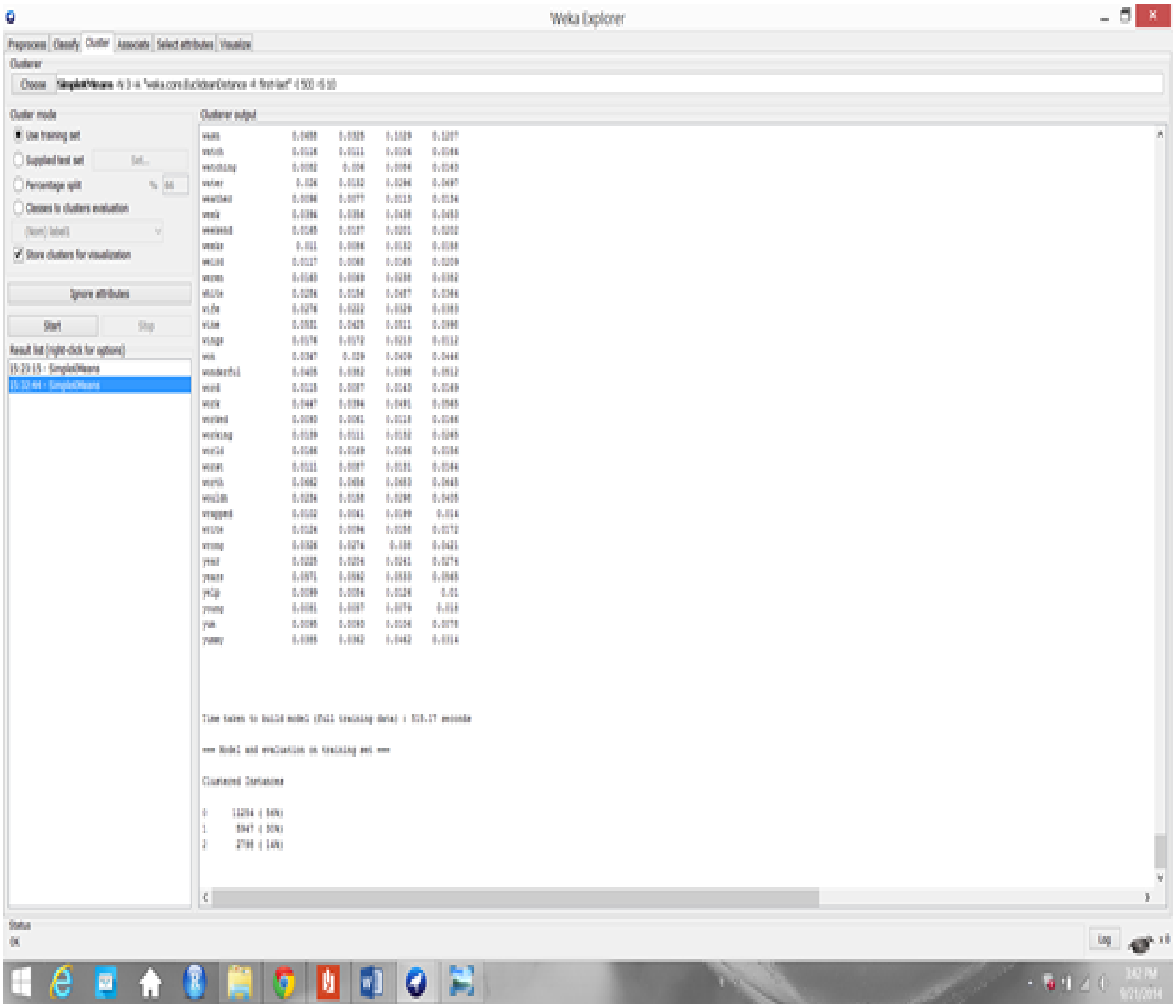
Explanation:

K means clustering created with cluster 0 containing 65% of data points and cluster 1 containing 35% of data points. However, ‘high’ is the dominant rating in both the clusters (around 73% in cluster 0 and 58% in cluster 1). This suggests that clusters are not able to separate the ratings properly.

Assuming that cluster 0 represents reviews with 'high' rating and cluster 1 represents reviews with 'low' rating, clustering gave us an accuracy of 62% approximately. Therefore, we decided to try clustering with other values of K to understand the clusters better.

2. K = 3:

Cluster Output:

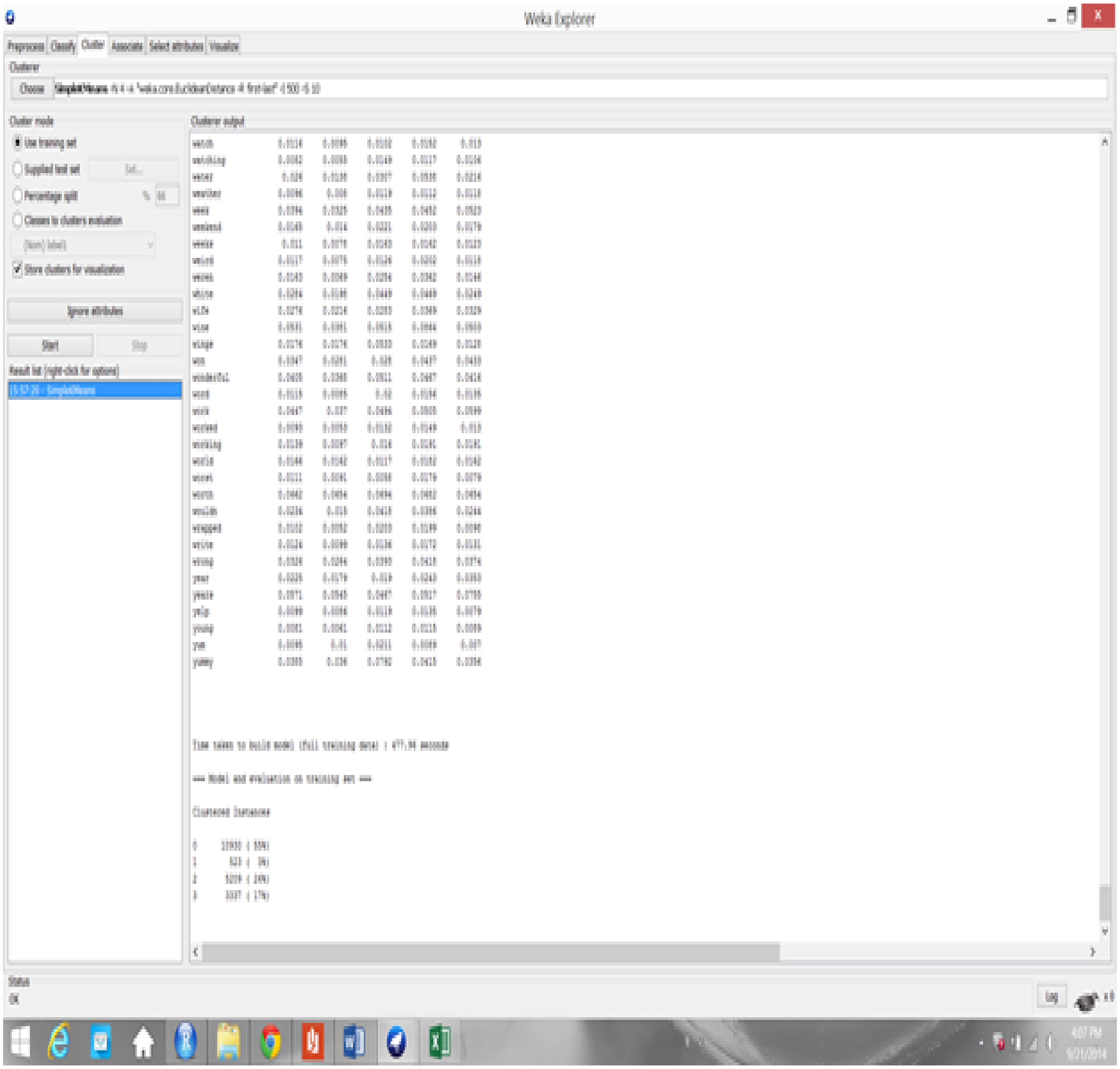


Distribution of 'low' and 'high' in each cluster:

Ratings	**Cluster 0**	**Cluster 1**	**Cluster 2**	**Total**
high	8299	3774	1480	13553 (68%)
low	2955	2173	1318	6446 (32%)
Total	**11254 (56%)**	**5947 (30%)**	**2798 (14%)**	**19999**

3. K = 4:

Cluster Output:

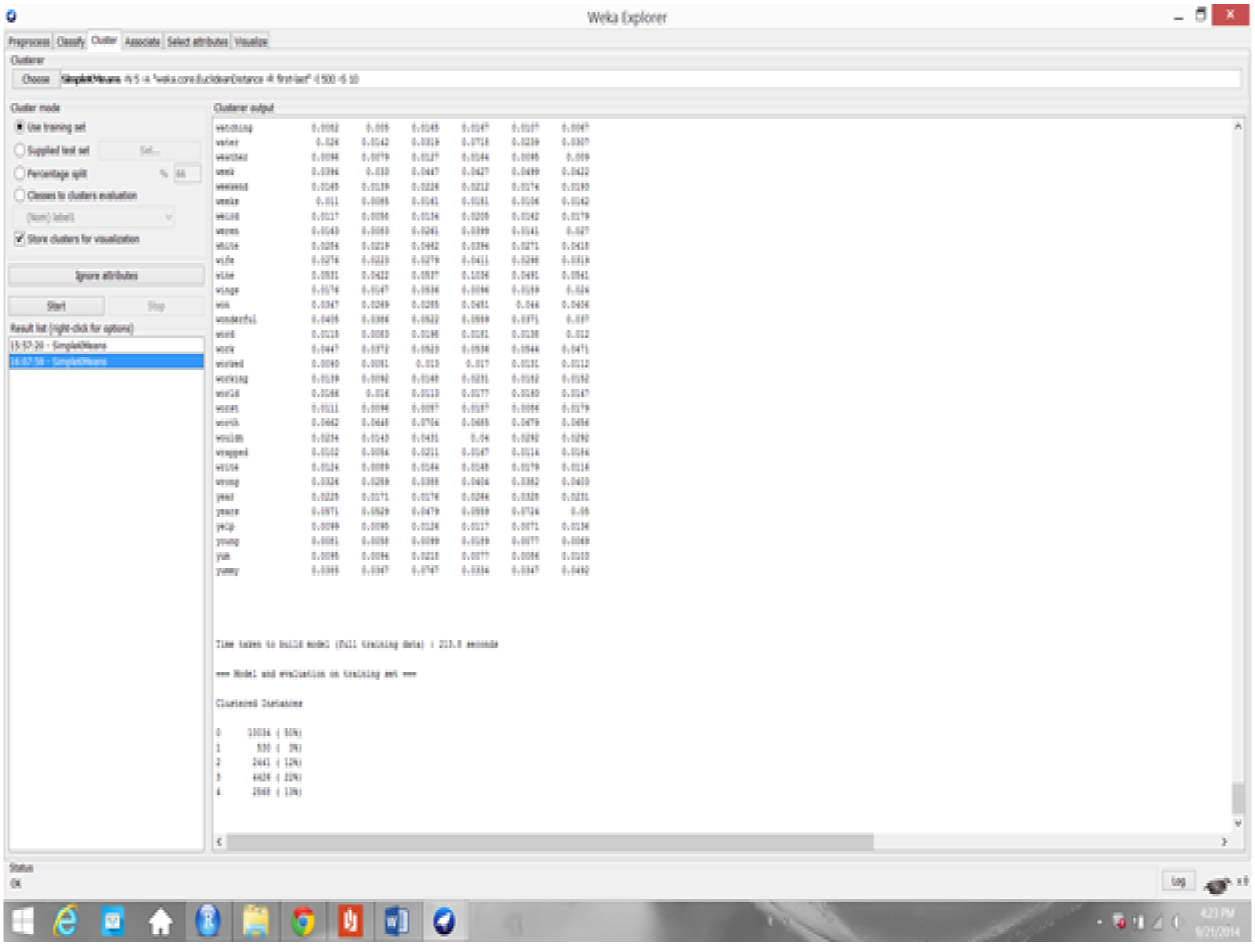


Distribution of 'low' and 'high' in each cluster:

Ratings	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Total
high	7989	386	2870	2308	13553 (68%)
low	2941	137	2339	1029	6446 (32%)
Total	**10930 (54%)**	**523 (3%)**	**5209 (26%)**	**3337 (17%)**	**19999**

4. K = 5:

Cluster Output:



Distribution of 'low' and 'high' in each cluster:

Ratings	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Total
high	7330	391	1330	3031	1471	13553 (68%)
low	2704	139	1111	1395	1097	6446 (32%)
Total	**10034 (50%)**	**530 (3%)**	**2441 (12%)**	**4426 (22%)**	**2568 (13%)**	**19999**

Clustering with almost all values of K, suggested that 'high' is dominant in almost all the clusters. Therefore, clustering doesn't separate rated restaurants between 'high' and 'low' satisfactorily. This also makes sense because we are only considering comments from the review and ignoring other numeric parameters of the review.