Kaggle Assignment Two

-- See Click Predict Fix

Liuxun ZHANG

MSIT student
Rensselaer Polytechnic Institute

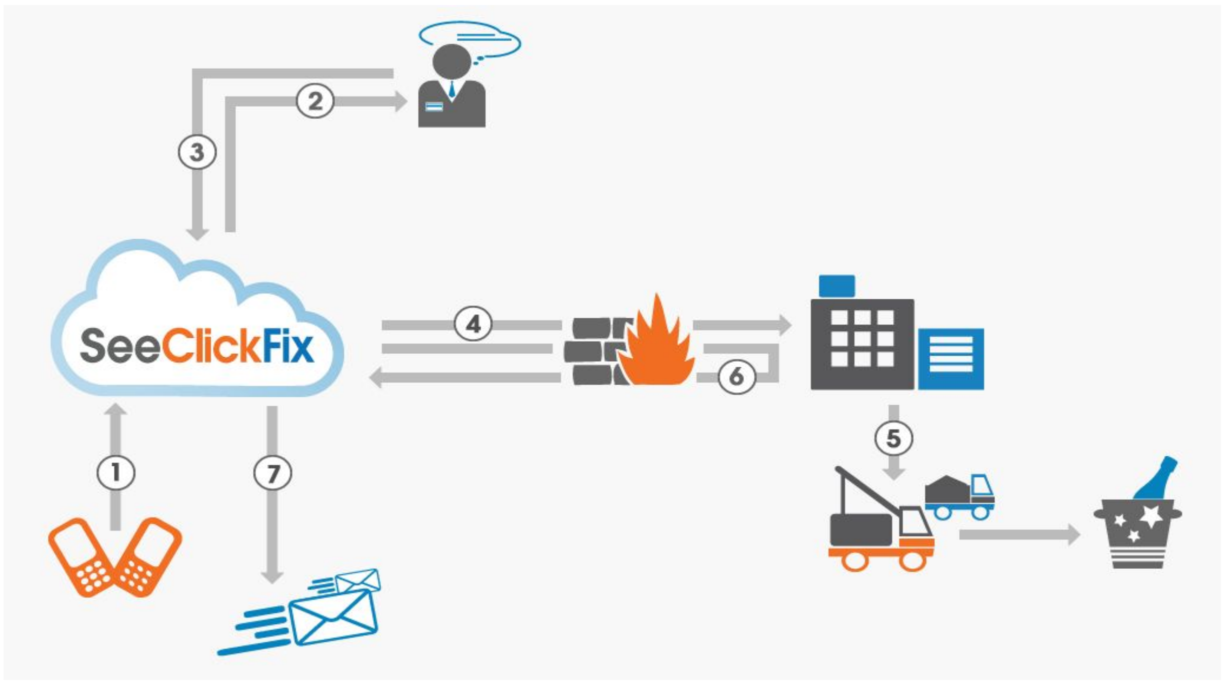
Overview of Analytics Problem

Predict which 311 issues are most important to citizens

This competition is the successor to the See Click Predict Fix Hackathon. The purpose of both competitions is to quantify and predict how people will react to a specific 311 issue. What makes an issue urgent? What do citizens really care about? How much does location matter? Being able to predict the most pressing 311 topics will allow governments to focus their efforts on fixing the most important problems. The data set for the competitions contains several hundred thousand 311 issues from four cities.

About 311

311 is a mechanism by which citizens can express their desire to solve a problem the city or government by submitting a description of what needs to be done, fixed, or changed. In effect, this provides a high degree of transparency between government and its constituents. Once an issue has been established, citizens can vote and make comments on the issue so that government officials have some degree of awareness about what is the most important issue to address.



Evaluation

The model should predict, for each issue in the test set, the number of views, votes, and comments. The Root Mean Squared Logarithmic Error (RMSLE) will be used to measure the accuracy.

The RMSLE is calculated as

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2}$$

Where:

n is three times the total number of issues in the test set (summing over each of views, votes, and comments for each issue)

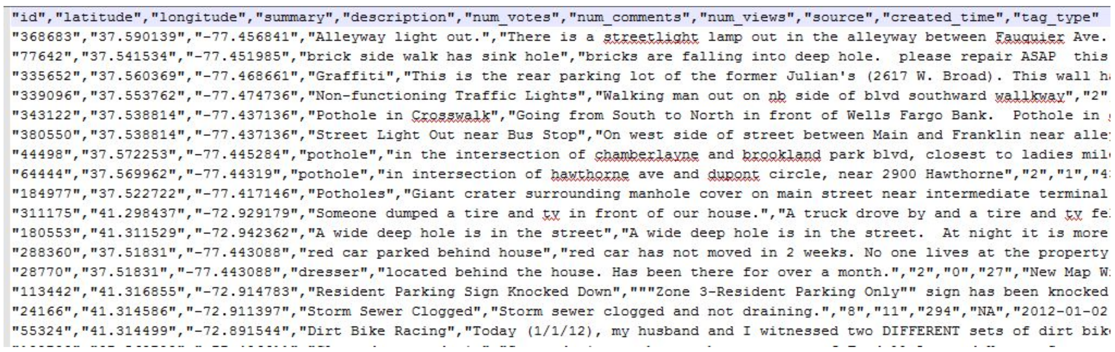
pi is your predicted value

ai is the actual value

log(x) is the natural logarithm

Analytic Approach

The snapshot of training data is shown below,



```
"id","latitude","longitude","summary","description","num_votes","num_comments","num_views","source","created_time","tag_type"
"368683","37.590139","-77.456841","Alleyway light out.","There is a streetlight lamp out in the alleyway between Fauquier Ave.
"77642","37.541534","-77.451985","brick side walk has sink hole","bricks are falling into deep hole.  please repair ASAP  this
"335652","37.560369","-77.468661","Graffiti","This is the rear parking lot of the former Julian's (2617 W. Broad). This wall h
"339096","37.553762","-77.474736","Non-functioning Traffic Lights","Walking man out on nb side of blvd southward wallkway","2"
"343122","37.538814","-77.437136","Pothole in Crosswalk","Going from South to North in front of Wells Fargo Bank.  Pothole in
"380550","37.538814","-77.437136","Street Light Out near Bus Stop","On west side of street between Main and Franklin near alle
"44498","37.572253","-77.445284","pothole","in the intersection of chamberlayne and brookland park blvd, closest to ladies mil
"64444","37.569962","-77.44319","pothole","in intersection of hawthorne ave and dupont circle, near 2900 Hawthorne","2","1","4
"184977","37.522722","-77.417146","Potholes","Giant crater surrounding manhole cover on main street near intermediate terminal
"311175","41.298437","-72.929179","Someone dumped a tire and tv in front of our house.","A truck drove by and a tire and tv fe
"180553","41.311529","-72.942362","A wide deep hole is in the street","A wide deep hole is in the street.  At night it is more
"288360","37.51831","-77.443088","red car parked behind house","red car has not moved in 2 weeks. No one lives at the property
"28770","37.51831","-77.443088","dresser","located behind the house. Has been there for over a month.","2","0","27","New Map W
"113442","41.316855","-72.914783","Resident Parking Sign Knocked Down","""Zone 3-Resident Parking Only"" sign has been knocked
"24166","41.314586","-72.911397","Storm Sewer Clogged","Storm sewer clogged and not draining.","8","11","294","NA","2012-01-02
"55324","41.314499","-72.891544","Dirt Bike Racing","Today (1/1/12), my husband and I witnessed two DIFFERENT sets of dirt bik
```

Basic analysis of the given features in training data:

Feature Name	Feature Type	Feature Description	Null-able
"id"	Integer	Randomly generated id numbers for all the records. Just for identification, non-relevant with prediction.	No
"latitude"	Float	Latitude of the 311 issue. Serving as location information along with longitude.	No
"longitude"	Float	Longitude of the 311 issue. Serving as location information along with latitude.	No
"summary"	Text	Summary/title of the 311 issue. Usually 3-8 words.	No
"description"	Text	Detailed description of the 311 issue.	No

		Hundreds of words.	
"num_votes"	Integer	Number of votes of the 311 issue.	No
"num_comments"	Integer	Number of comments of the 311 issue.	No
"num_views"	Integer	Number of views of the 311 issue.	No
"source"	Text (factor)	Collection source of the 311 issue.	Yes
"created_time"	Data and time	Created time for the 311 issue.	No
"tag_type"	Text (factor)	Tag for the 311 issue. Word or phrase.	Yes

After taking a look at the training data, it is clear that several text data are provided as main features for the prediction. Summary and description of the 311 issue play important roles for predicting the number of votes, comments and views due to the natural reasons.

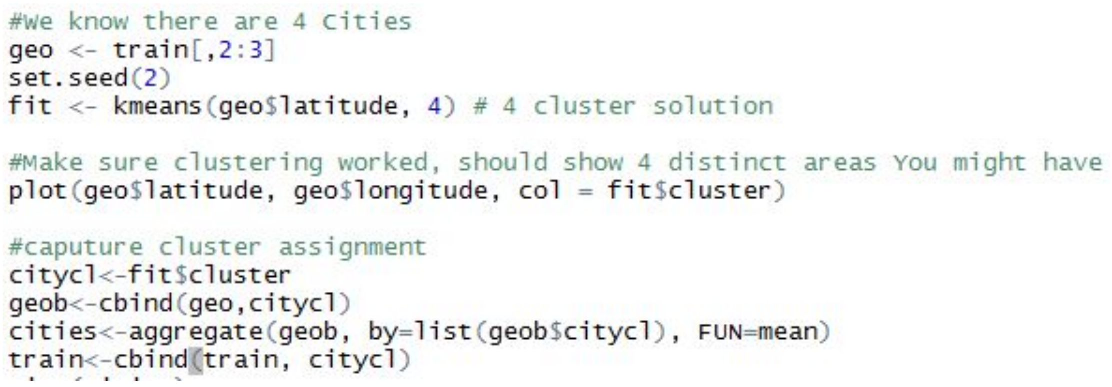
So the approach to solve the prediction problem contains two main parts:

Feature conversion and Text-mining.

Feature Conversion

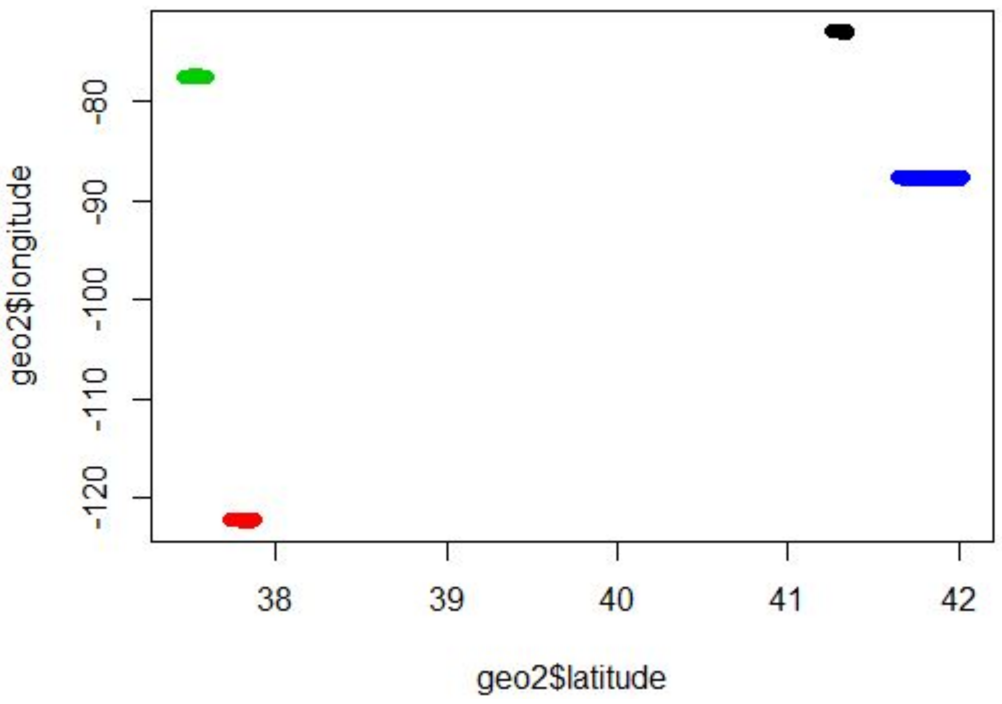
Location

Location data is generated for all records by using latitude and longitude data to do K-means clustering.
Code shown as below,



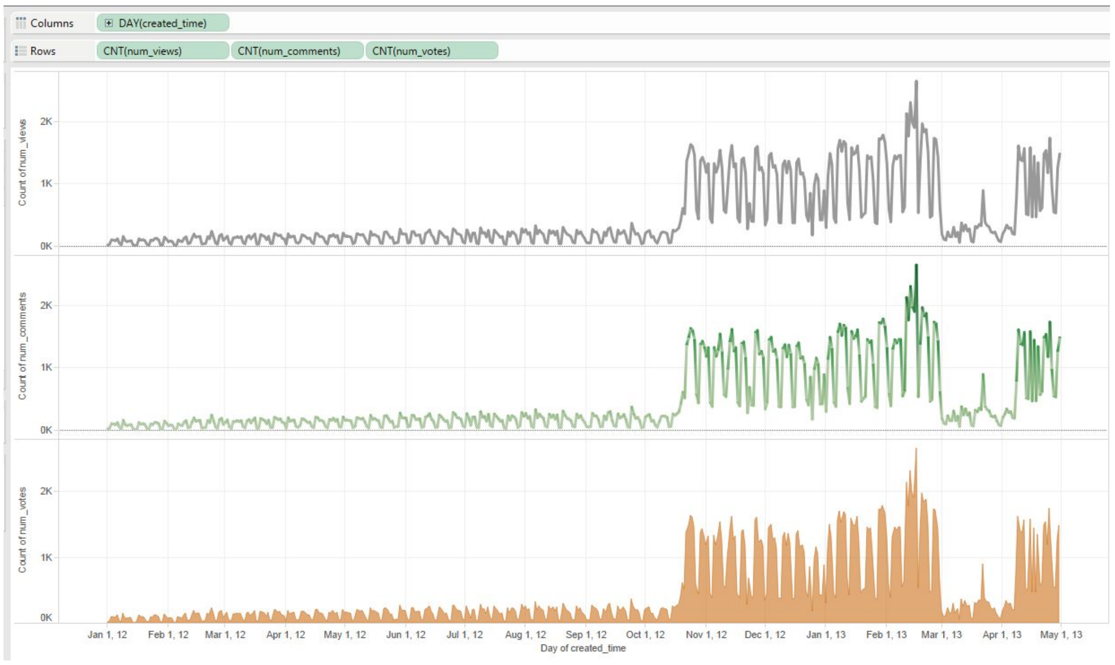
```
#We know there are 4 cities
geo <- train[,2:3]
set.seed(2)
fit <- kmeans(geo$latitude, 4) # 4 cluster solution

#Make sure clustering worked, should show 4 distinct areas You might have
plot(geo$latitude, geo$longitude, col = fit$cluster)

#caputure cluster assignment
citycl<-fit$cluster
geob<-cbind(geo,citycl)
cities<-aggregate(geob, by=list(geob$citycl), FUN=mean)
train<-cbind(train, citycl)
```

Clustering result is shown as below,

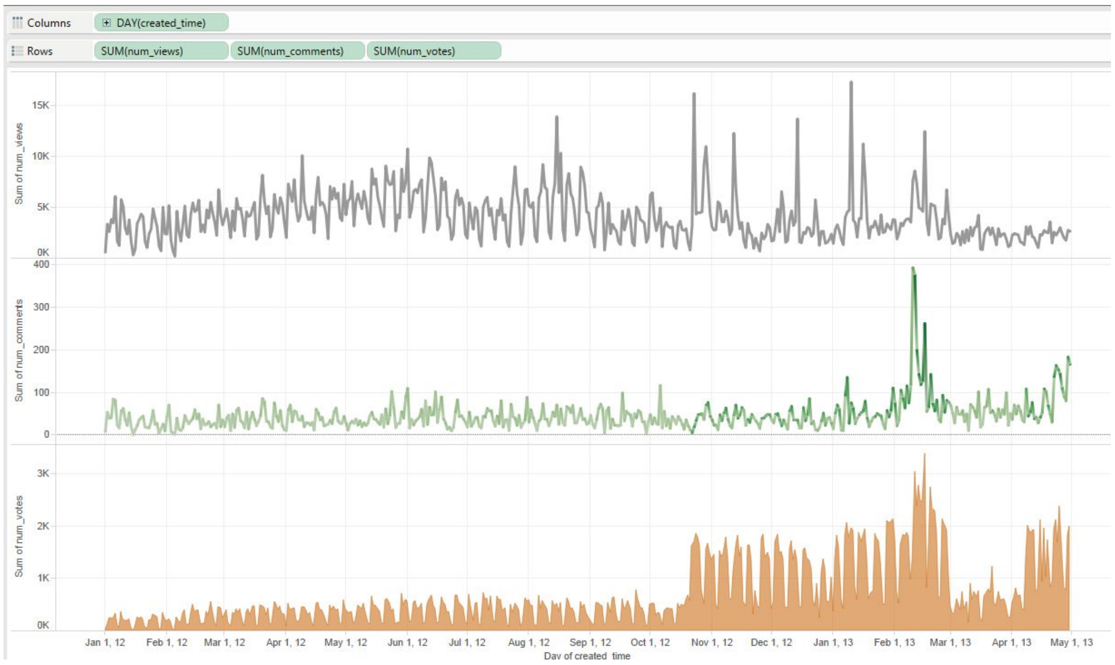


Data and Time

By using Tableau to plot the count of num_votes, num_comments and num_views against date time data, it is easy to recognise that numbers before Nov 01, 2012 was much lower than later. So it is concluded that the training data should be divided to at least 2 parts in terms of created_time.



Plotting below is sum of num_votes, num_comments and num_views against date time data. The num_votes performed the same as previous, however the num_views and comments performed differently and the clusters were not obvious before and after Nov 01, 2012.



Comparison between the data after Nov 01,2012 and the entire data were used separately to train the models.

Source

The source of the 311 issue. Since they are text format, it is better to convert them into numeric values.

```
train$source_numeric<-ifelse(train$source=="android",1,train$source)
train$missing.source <- is.na(train$source_numeric)

#train$source_numeric<-ifelse(train$source=="android",1,train$source)
train$source_numeric[is.na(train$source_numeric)] <- 0
```

As the code shown above, a new feature named source_numeric were created in the training data. Similar code for the testing data.

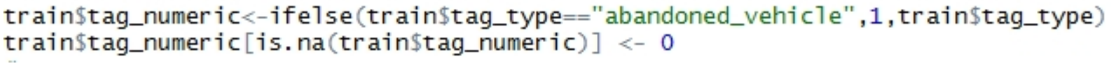
```
test$source_numeric<-ifelse(test$source=="android",1,test$source)
test$missing.source <- is.na(test$source_numeric)

#test$source_numeric<-ifelse(test$source=="android",1,test$source)
test$source_numeric[is.na(test$source_numeric)] <- 0
```

Tag_Type

The tagging for each record, as shown below, there are totally 43 tags for training data.

```
structure(c(126L, 117L, 54L, 22L, 54L, 157L, 1024L, 169L, 110L,
393L, 106L, 134L, 4066L, 26L, 139L, 2209L, 34L, 3L, 46L, 47L,
1L, 1594L, 58L, 15L, 6066L, 26L, 1L, 1L, 184L, 23L, 72L, 30L,
1585L, 1912L, 676L, 3768L, 32L, 41L, 705L, 19791L, 7788L, 10L,
169714L), .Names = c("abandoned_vehicle", "abandoned_vehicles",
"animal_problem", "bad_driving", "bench", "bike_concern", "blighted_property",
"bridge", "crosswalk", "drain_problem", "drug_dealing", "flood",
"graffiti", "heat", "homeless", "hydrant", "illegal_idling",
"lost_and_found", "noise_complaint", "odor", "other", "overgrowth",
"parking_meter", "pedestrian_light", "pothole", "prostitution",
"public_art", "public_concern", "road_safety", "roadkill", "robbery",
"rodents", "sidewalk", "signs", "snow", "street_light", "street_signal",
"test", "traffic", "trash", "tree", "zoning", "NA's"))
```

There are more than half records in the training data set that don't have a Tag_type, which means NA is listed in the cell.
It would be better to use numeric values and put values for the NAs.
The following code converts tag_type to tag_numeric by using ifelse function and then NA is replaced with "0" by using is.na function.

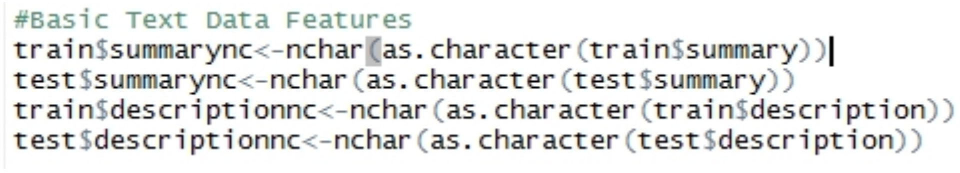


```
train$tag_numeric<-ifelse(train$tag_type=="abandoned_vehicle",1,train$tag_type)
train$tag_numeric[is.na(train$tag_numeric)] <- 0
```

Text-Mining

Number of characters in summary

Count of characters in the summary column then store into a new column named summarync. Code shown below,



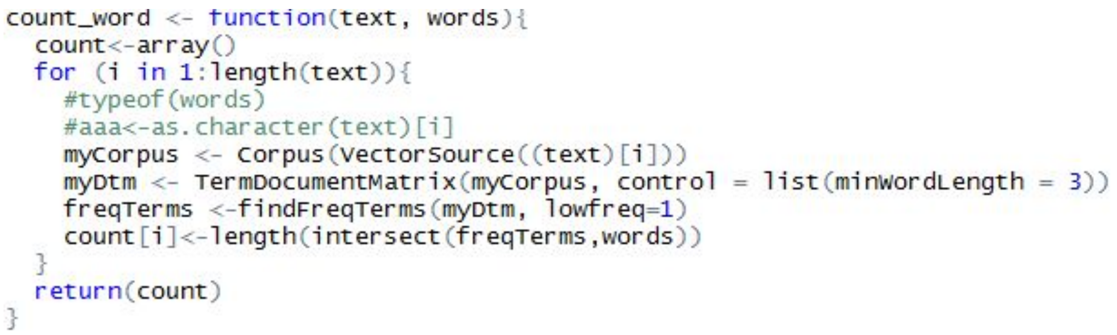
```
#Basic Text Data Features
train$summarync<-nchar(as.character(train$summary))
test$summarync<-nchar(as.character(test$summary))
train$descriptionnc<-nchar(as.character(train$description))
test$descriptionnc<-nchar(as.character(test$description))
```

Number of characters in description

Count of characters in the description column then store into a new column named descriptionnc. As shown above.

Creation of count_word function

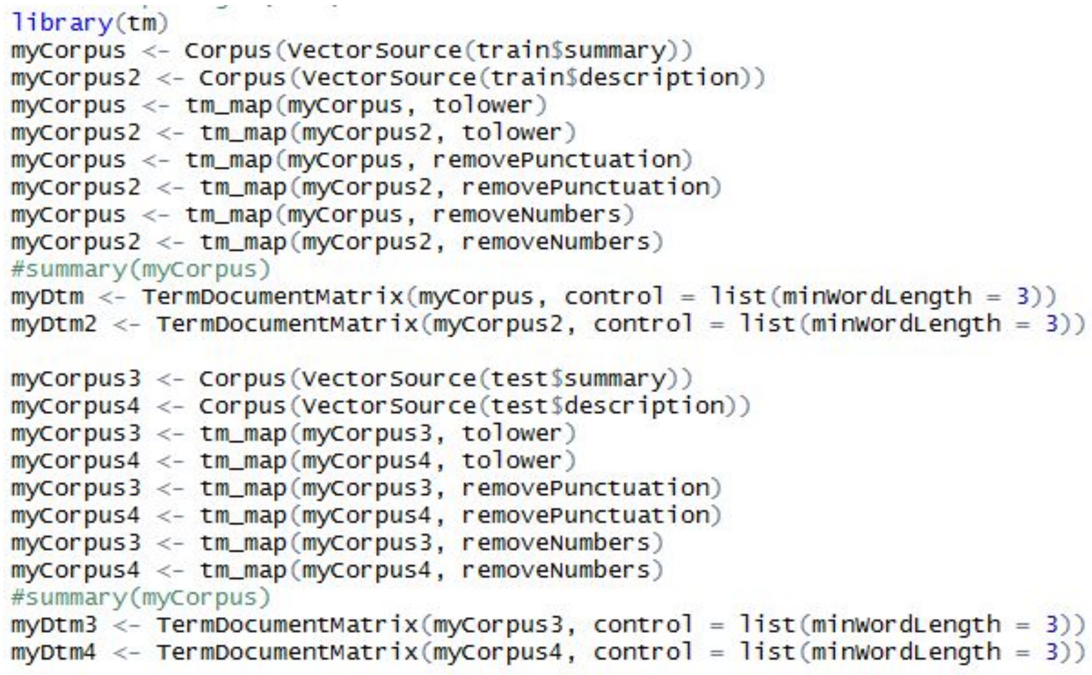
A personal function is created. It takes two input arguments: text, which is a column of text (like train$summary and train$description) and words, which is a list of words.
For each item in text, the function first converts text into text document using methods from the package "tm". Then the function computes the intersection of two sets of strings, text document and words and stores the result to the array named count.
After the full iteration of given text, the same length column count is returned for further use.



```
count_word <- function(text, words){
  count<-array()
  for (i in 1:length(text)){
    #typeof(words)
    #aaa<-as.character(text)[i]
    myCorpus <- Corpus(VectorSource((text)[i]))
    myDtm <- TermDocumentMatrix(myCorpus, control = list(minWordLength = 3))
    freqTerms <-findFreqTerms(myDtm, lowfreq=1)
    count[i]<-length(intersect(freqTerms,words))
  }
  return(count)
}
```

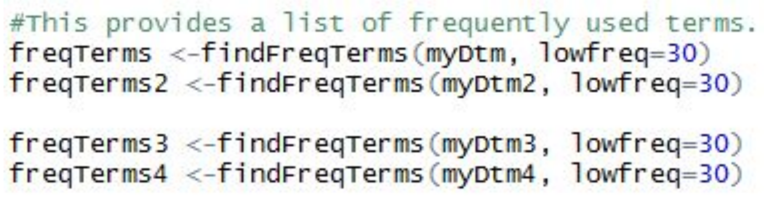
Conversion of summary and description into corpus and further analysis

Using the same package “tm”, Corpus of summary and description are created for both training and testing data. Then the TermDocumentMatrix are created for summary and description of both training and testing data.



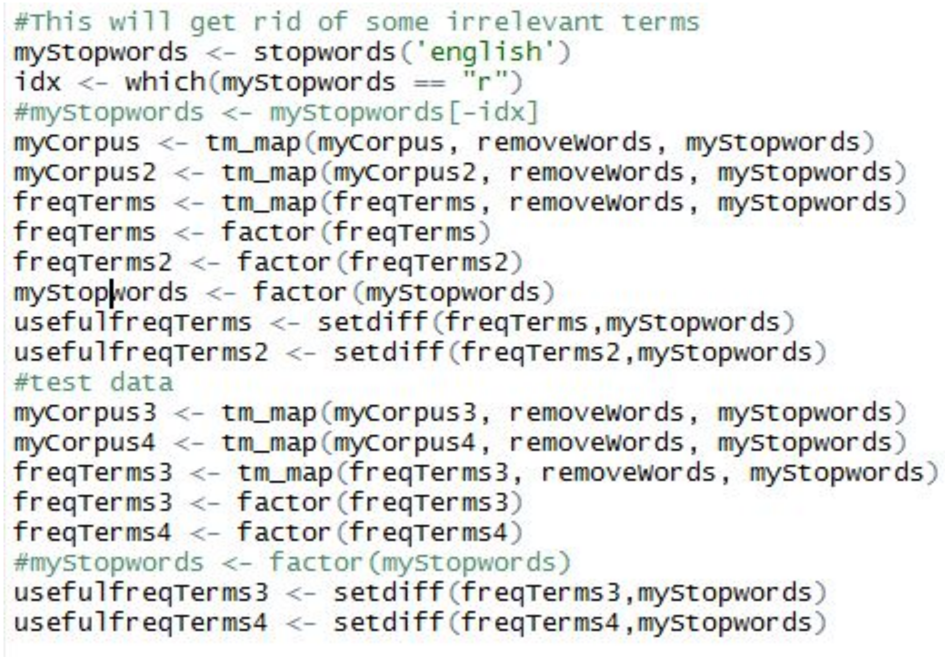
```
library(tm)
myCorpus <- Corpus(VectorSource(train$summary))
myCorpus2 <- Corpus(VectorSource(train$description))
myCorpus <- tm_map(myCorpus, tolower)
myCorpus2 <- tm_map(myCorpus2, tolower)
myCorpus <- tm_map(myCorpus, removePunctuation)
myCorpus2 <- tm_map(myCorpus2, removePunctuation)
myCorpus <- tm_map(myCorpus, removeNumbers)
myCorpus2 <- tm_map(myCorpus2, removeNumbers)
#summary(myCorpus)
myDtm <- TermDocumentMatrix(myCorpus, control = list(minWordLength = 3))
myDtm2 <- TermDocumentMatrix(myCorpus2, control = list(minWordLength = 3))

myCorpus3 <- Corpus(VectorSource(test$summary))
myCorpus4 <- Corpus(VectorSource(test$description))
myCorpus3 <- tm_map(myCorpus3, tolower)
myCorpus4 <- tm_map(myCorpus4, tolower)
myCorpus3 <- tm_map(myCorpus3, removePunctuation)
myCorpus4 <- tm_map(myCorpus4, removePunctuation)
myCorpus3 <- tm_map(myCorpus3, removeNumbers)
myCorpus4 <- tm_map(myCorpus4, removeNumbers)
#summary(myCorpus)
myDtm3 <- TermDocumentMatrix(myCorpus3, control = list(minWordLength = 3))
myDtm4 <- TermDocumentMatrix(myCorpus4, control = list(minWordLength = 3))
```

Then the frequent word are generated for the TermDocumentMatrixs, with the lowest frequency 30 as lower limit. Code as below,



```
#This provides a list of frequently used terms.
freqTerms <-findFreqTerms(myDtm, lowfreq=30)
freqTerms2 <-findFreqTerms(myDtm2, lowfreq=30)

freqTerms3 <-findFreqTerms(myDtm3, lowfreq=30)
freqTerms4 <-findFreqTerms(myDtm4, lowfreq=30)
```

Stopwords are removed from the frequent terms found previously, code as below,



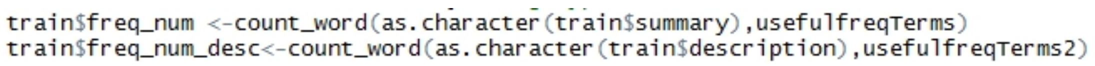
```
#This will get rid of some irrelevant terms
myStopwords <- stopwords('english')
idx <- which(myStopwords == "r")
#myStopwords <- myStopwords[-idx]
myCorpus <- tm_map(myCorpus, removeWords, myStopwords)
myCorpus2 <- tm_map(myCorpus2, removeWords, myStopwords)
freqTerms <- tm_map(freqTerms, removeWords, myStopwords)
freqTerms <- factor(freqTerms)
freqTerms2 <- factor(freqTerms2)
myStopwords <- factor(myStopwords)
usefulfreqTerms <- setdiff(freqTerms,myStopwords)
usefulfreqTerms2 <- setdiff(freqTerms2,myStopwords)
#test data
myCorpus3 <- tm_map(myCorpus3, removeWords, myStopwords)
myCorpus4 <- tm_map(myCorpus4, removeWords, myStopwords)
freqTerms3 <- tm_map(freqTerms3, removeWords, myStopwords)
freqTerms3 <- factor(freqTerms3)
freqTerms4 <- factor(freqTerms4)
#myStopwords <- factor(myStopwords)
usefulfreqTerms3 <- setdiff(freqTerms3,myStopwords)
usefulfreqTerms4 <- setdiff(freqTerms4,myStopwords)
```

Now usefulfreqTerms are generated and we can count the frequent words in summary and description.

Number of frequent words in summary

Number of frequent words in description

Code to achieve the counting is shown below, simply calling the function and storing the results.

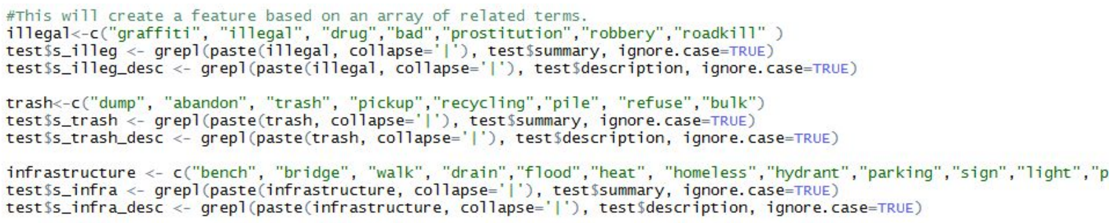


```
train$freq_num <-count_word(as.character(train$summary),usefulfreqTerms)
train$freq_num_desc<-count_word(as.character(train$description),usefulfreqTerms2)
```

Number of illegal/trash/infrastructure words in summary

Number of illegal/trash/infrastructure words in description

Word sets of illegal/trash/infrastructure are created in advance.
Code to achieve the counting for testing data is shown below, simply calling the function with different inputs and storing the results.



```
#This will create a feature based on an array of related terms.
illegal<-c("graffiti", "illegal", "drug","bad","prostitution","robbery","roadkill" )
test$s_illeg <- grepl(paste(illegal, collapse='|'), test$summary, ignore.case=TRUE)
test$s_illeg_desc <- grepl(paste(illegal, collapse='|'), test$description, ignore.case=TRUE)

trash<-c("dump", "abandon", "trash", "pickup","recycling","pile", "refuse","bulk")
test$s_trash <- grepl(paste(trash, collapse='|'), test$summary, ignore.case=TRUE)
test$s_trash_desc <- grepl(paste(trash, collapse='|'), test$description, ignore.case=TRUE)

infrastructure <- c("bench", "bridge", "walk", "drain","flood","heat", "homeless","hydrant","parking","sign","light","p
test$s_infra <- grepl(paste(infrastructure, collapse='|'), test$summary, ignore.case=TRUE)
test$s_infra_desc <- grepl(paste(infrastructure, collapse='|'), test$description, ignore.case=TRUE)
```

Prediction

Most of the classification algorithms can’t predict multiple dependent variables at the same time, thus the order of my prediction is:

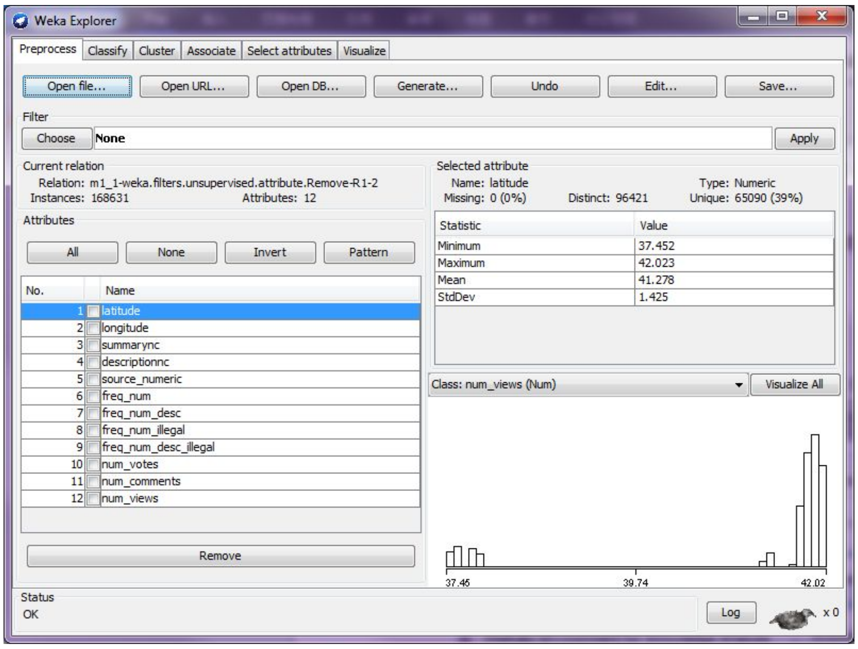
$$\begin{tikzcd}\text{Num\_votes} \arrow[r] & \text{num\_comments} \arrow[r] & \text{num\_views}\end{tikzcd}$$

Cross-validation

Cross-validation is done by using Weka 3.6.9.

Waikato Environment for Knowledge Analysis
Version 3.6.9
(c) 1999 - 2013
The University of Waikato
Hamilton, New Zealand

After exporting the training and testing data to CSV files, Weka is used because it has GUI and the arguments of algorithms are easy to be set. Also since Weka is written in Java the prediction in Weka cost less time.
Data imported to Weka as shown below,



Filter: None

Current relation
Relation: m1_1-weka.filters.unsupervised.attribute.Remove-R1-2
Instances: 168631 Attributes: 12

Attributes

No.	Name
1	latitude
2	longitude
3	summarync
4	descriptionnc
5	source_numeric
6	freq_num
7	freq_num_desc
8	freq_num_illegal
9	freq_num_desc_illegal
10	num_votes
11	num_comments
12	num_views

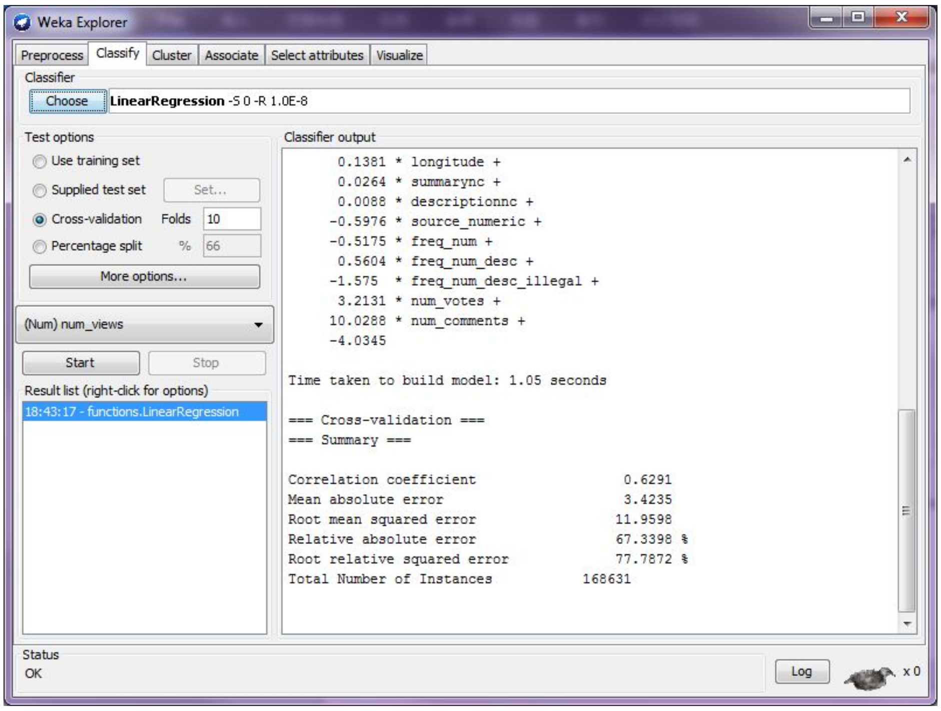
Selected attribute
Name: latitude Type: Numeric
Missing: 0 (0%) Distinct: 96421 Unique: 65090 (39%)

Statistic	Value
Minimum	37.452
Maximum	42.023
Mean	41.278
StdDev	1.425

Class: num_views (Num)

Status
OK

Cross-validation snapshot is shown as below,



LinearRegression -S 0 -R 1.0E-8

Use training set
Supplied test set
[✓ Cross-validation] Folds 10
Percentage split % 66

(Num) num_views

18:43:17 - functions.LinearRegression

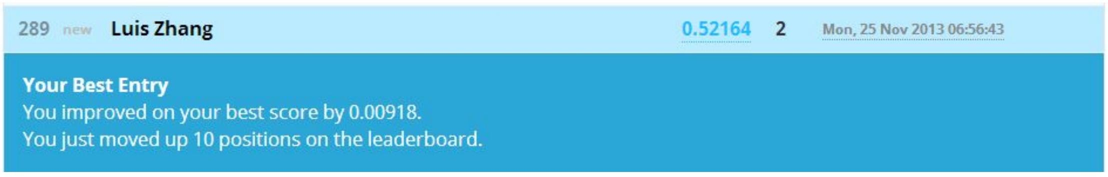
```
      0.1381 * longitude +
      0.0264 * summarync +
      0.0088 * descriptionnc +
     -0.5976 * source_numeric +
     -0.5175 * freq_num +
      0.5604 * freq_num_desc +
     -1.575  * freq_num_desc_illegal +
      3.2131 * num_votes +
     10.0288 * num_comments +
     -4.0345

Time taken to build model: 1.05 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.6291
Mean absolute error                      3.4235
Root mean squared error                 11.9598
Relative absolute error                 67.3398 %
Root relative squared error             77.7872 %
Total Number of Instances           168631
```

Status
OK

Several different algorithms are applied for prediction on testing data.
Training data are taken from different columns of train data.
Arguments are changed and attempted several times for prediction.
The main methods applied are shown as below,

```
library("nnet")
train.nn2 <- nnet(num_votes~citycl+summarync+descriptionnc+s_illegTRUE+s_illeg_descTRUE+s_trash
                  decay = 5e-4, maxit = 500)
predict(train.nn2, n )

library("e1071")
model<- svm(num_votes~citycl+summarync+descriptionnc+s_illegTRUE+s_illeg_descTRUE+s_trashTRUE+s

library("neuralnet")
train.nn2 <- neuralnet(num_votes~citycl+summarync+descriptionnc+s_illegTRUE+s_illeg_descTRUE+s_

result<-compute(train.nn2, n)
write.csv(result, 'predict.csv')
table(test$num_votes, )


library("randomForest")
train.rf <- randomForest(num_votes~citycl+summarync+descriptionnc+s_illeg+s_illeg_desc+s_trash+

pred <- predict(train.rf, test)
#pred_b <- ifelse(pred > 0.5, 1, 0)
```

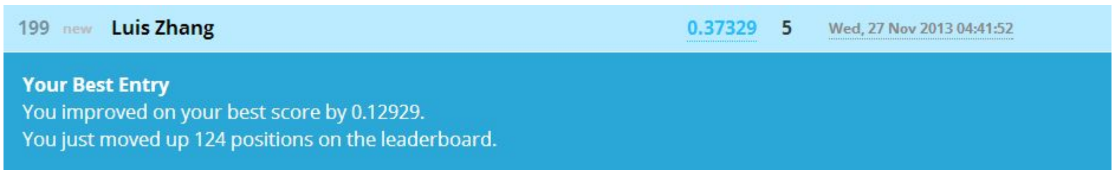
Result of using data from the entire training set:



289 new Luis Zhang 0.52164 2 Mon, 25 Nov 2013 06:56:43

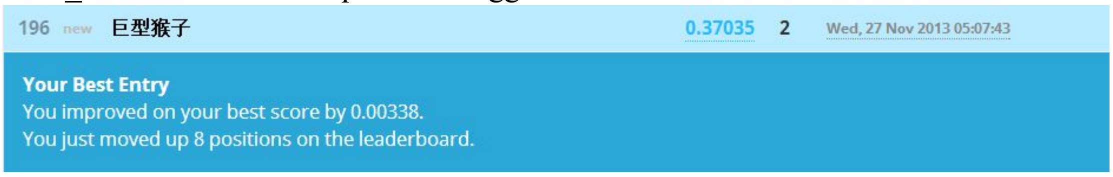
Your Best Entry
You improved on your best score by 0.00918.
You just moved up 10 positions on the leaderboard.

Result of using data from the training set from Nov 01/2012 to the end.


199 new Luis Zhang 0.37329 5 Wed, 27 Nov 2013 04:41:52

Your Best Entry
You improved on your best score by 0.12929.
You just moved up 124 positions on the leaderboard.

Modification applied to the testing result data using excel.
Num_votes are set to be equal to or bigger than one.


196 new 巨型猴子 0.37035 2 Wed, 27 Nov 2013 05:07:43

Your Best Entry
You improved on your best score by 0.00338.
You just moved up 8 positions on the leaderboard.

Excel formula: =IF(AND(B2>0,B2<=1),1,B2)

Conclusion

- Analyze the data type/format first.
- Plotting the data is extremely helpful.
- If time and location is provided, use them wisely.
- Text mining is helpful for the results, but time-consuming, so testing on smaller sets first.
- Cross validating on training data then apply the model on testing data.
- Some algorithms perform better with continuous numbers (SVM,Neural Networks), while some perform better with discrete numbers(trees and random forests). Linear Regression is perfect because it’s super fast.