Project 3: Movie Review Sentiment Analysis

Team

1. Aman Arora – amana4@illinois.edu
2. Amartya Roy Chowdhury -- amartya4@illinois.edu
3. Kumar Gaurav Dubey -- kgdubey2@illinois.edu

Introduction

This is the project where we did sentiment analysis of IMDB reviews. In this project set-up the data has been divided into five different splits each containing a training and testing set, each containing 25000 data points. Our aim is to achieve an AUC level of 0.96 or better while not exceeding the vocabulary size of 1000. Output of this project is a file “myvocab.txt” containing vocabulary words less than or equal to 1000 and “mymain.r” which takes “myvocab.txt” as input. Each row contains 3 columns: 1) new_id being the ID of the review, 2) sentiment value of 1 (positive) and 0 (negative) and 3) review contains the review text.

Technical Details

We experimented with following approaches:

1. Logistic regression model with L2 (Ridge) penalty. DTM generated from the vocabulary file is used as the explanatory variables and sentiment as response variable. We use cross validation to pick the minimum lambda. The AUC score for the 5 spilts is as below:
2. Preprocessing data using td-IDF transformation followed by L2 (Ridge) penalty. DTM generated from the vocabulary file is used as the explanatory variables and sentiment as response variable. Logistic regression with L2 penalty was used for model filtting. This approach did not result in significant improvement in AUC from 1st approach and was not pursued further.

Implementation

We have used text2vec packages to create a document-term matrix. The glmnet package was used to fit a logistic regression with L1 (lasso: alpha=1) penalty to build a custom bag of n-grams after cleaning by considering the stop words. The steps are as follows.

We start by reading in the entire **alldata.tsv** and the **splits_F21.csv** and use the input R code to create all the 5 folders splits on which we are to test our model.

Data cleaning:

- We use train$review <- gsub('<. *?>', ' ', train$review) to remove punctuations and brackets.
- Stopwords removal: stopwords are the most common words.

- We use the ‘itoken’ function in text2vec to convert all reviews into lower cases and perform tokenization by specifying tokenizer = word_tokenizer.

Processing:

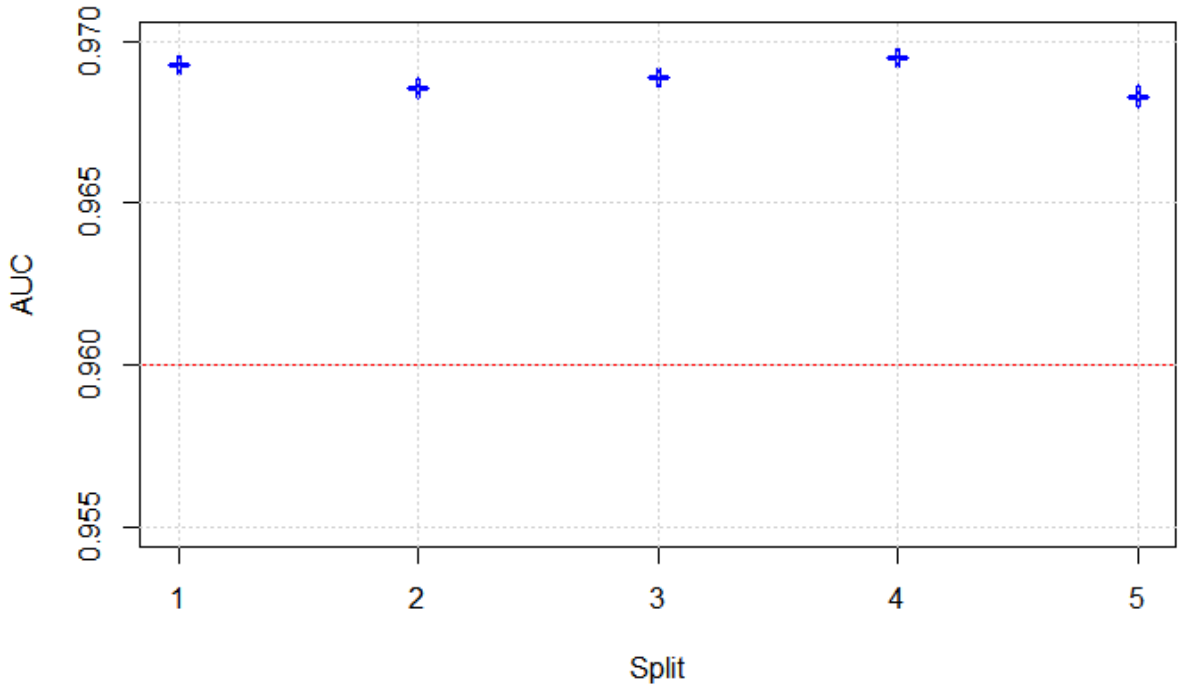
- Created a document-term matrix (DTM) fed with our customized.
- Used glmnet() with argument family='binomial' and logixtic regression Lasso.

Model Evaluation:

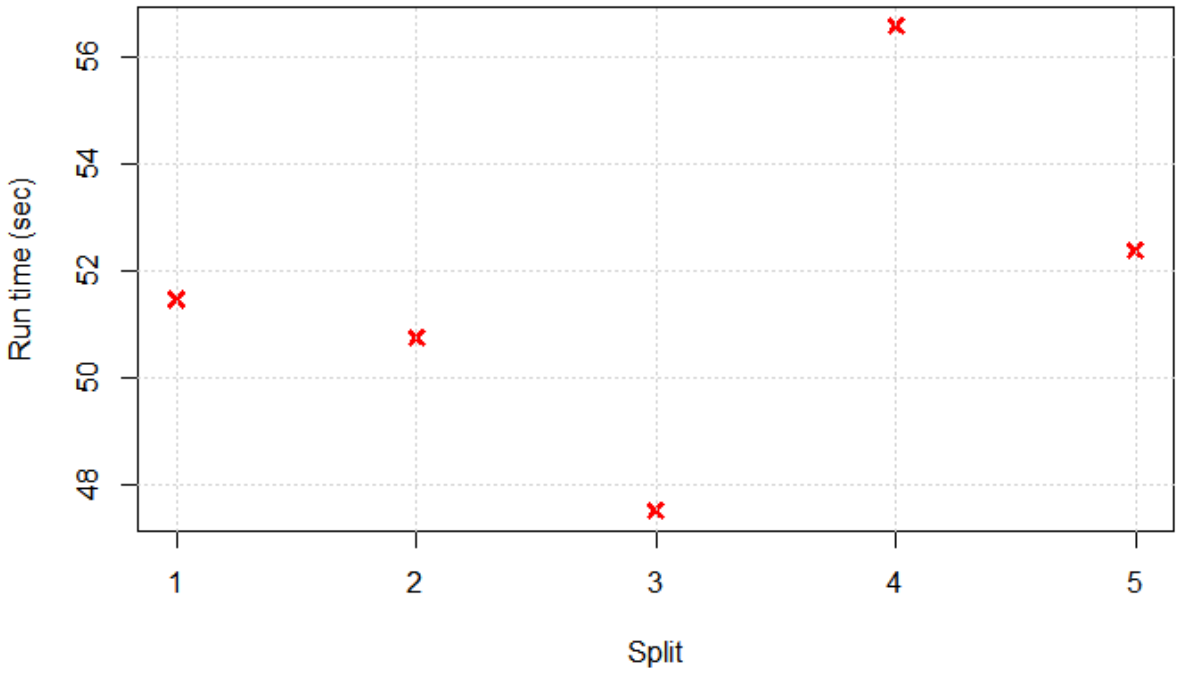
- The mymain.r will output submission.txt which will be evaluated against the test_y.tsv for
- Predicted sentiments in the respective fold and AUC (area under curve is calculated) is calculated as the evaluation metric.

Performance

The model uses a vocabulary list of 982 words – it performs relatively well on the test data. For all splits the AUC was greater than 0.96 and each split ran less than 57 secs



AUC vs Split



Runtime (sec) vs Split

Model Evaluation Results:

Split 1 AUC: 0.9693

Split 2 AUC: 0.9686

Split 3 AUC: 0.9689

Split 4 AUC: 0.9695

Split 5 AUC: 0.9683

Interesting Finds

As part of vocabulary selection we also ordered words by the magnitude of their t-statistics and picked the top words ordered by t-statistc value, which are then divided into two lists: positive words and negative words. We did not end up using this vocabulary as explanatory variables but its interesting to see the negative and positive sentiment words align with human understanding of negative and positive sentiment .

```{r}
pos.list[1:50]
```

```
 [1] "great"              "excellent"          "wonderful"          "best"               "of_best"            "one_of_best"        "love"
 [8] "perfect"            "loved"              "amazing"            "beautiful"          "superb"             "well"               "favorite"
[15] "brilliant"          "highly"             "life"               "must_see"           "also"               "fantastic"          "very"
[22] "beautifully"        "one_of"             "both"               "today"              "always"             "very_well"          "performance
[29] "enjoyed"            "performances"       "well_worth"         "highly_recommend"   "years"              "this_great"         "8_10"
[36] "touching"           "10_10"              "wonderfully"        "outstanding"        "young"              "world"              "perfectly"
[43] "highly_recommended" "strong"             "7_10"               "powerful"           "well_as"            "8"                  "as_well"
[50] "definitely"
```

```{r}
neg.list[1:50]
```

```
 [1] "bad"           "worst"          "waste"          "awful"          "terrible"       "worse"          "boring"         "no"
 [9] "stupid"        "nothing"        "waste_of"       "poor"           "horrible"       "of_worst"       "minutes"        "even"
[17] "so_bad"        "just"           "crap"           "supposed"       "one_of_worst"   "acting"         "poorly"         "supposed_to"
[25] "at_all"        "plot"           "why"            "script"         "ridiculous"     "waste_of_time"  "lame"           "worst_movie"
[33] "avoid"         "wasted"         "not_even"       "annoying"       "don't"          "oh"             "waste_time"     "dull"
[41] "mess"          "pointless"      "money"          "supposed_to_be" "laughable"      "any"            "cheap"          "pathetic"
[49] "thing"         "least"
```

Tech Specs

We used a Windows machine with following specs. Windows 10 – Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz 2.30 GHz: RAM:16.0GB

What we can do better

We can try other model like developing fast and high-performance gradient boosting tree models using XGBoost. We can also analyze the data using random forest. For penalty function we can also use Ridge regression. Ridge is computationally less intensive than Lasso. So we might get some better computational details.

Also, we consider generating the vocab from the main file which is having all data which we can improve by considering the split train file and then summing up all the split file vocab to generate the vocab.txt and then predict with the test data and do the analysis.

Acknowledgements:

- Piazza post and OH
- Kaggle Page "Bag of Words Meets Bags of Popcorn" https://www.kaggle.com/c/word2vec-nlp-tutorial
- [text2vec] R vignettes: https://cran.r-project.org/web/packages/text2vec/vignettes/text-vectorization.html