Stat Progr Packages Course Project Report

Tsering Dolkar, Jaeyoung Lee, Liang Shi

Dec 2020

Abstract

1 Introduction

This is an Presidential election year in the U.S. Here, you should create a dashboard of political interest. Ideally, the dashboard will contain a map and summary plots of the data. What data can you find that 1 might suggest one candidate over the other, locally, regionally, nationally? Is there a time component to the data showing enthusiams change? Is there social data available that might show trends? A few of the many possible data sources: Twitter:

1. https://towardsdatascience.com/understanding-political-twitter-ce3476a38377
2. https://web.archive.org/web/20160628093159/http://www.nsd.uib.no/ macrodataguide/country2.html?id= 840c=United%20States%20of%20America
3. https://www.cpds-data.org/

2 Model

2.1 Classification Tree model

Classification trees base splitting decisions on a metric, for example the trees presented in class use deviance to judge how to partition the tree. I used rpart tree classification package. Rpart uses gini impurity to select splits when performing classification. To quantitatively evaluate how good a split is,

1. We randomly pick a datapoint in a dataset, then
2. Randomly classify it according to the class distribution in the dataset.

If we have C total classes and $p(i)$ is the probability of picking a datapoint with class i, then the Gini Impurity is calculated as

$$G = \sum_{i=1}^{C} p(i) \times (1 - p(i))$$

Therefore, Gini impurity is the probability of incorrectly classifying a randomly chosen element in the dataset if it were randomly labeled according to the class distribution in the dataset. Rpart is trying to choose the best split by maximizing the Gini Gain. We do this by

original impurity – weighted impurities of the branch

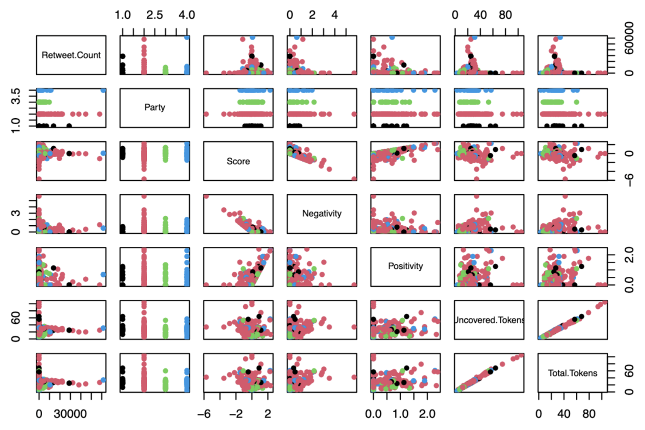


Figure 1: Scatterplot of the data

If we were to see clusters on the scatterplot, we will expect classification tree to be quite effective. However, this data doesn't seem very representative of all voters as we had initially hoped. However, we will proceed forward to see how well it works against other methods. Classification trees base splitting decisions on a metric, for example the trees presented in class use deviance to judge how to partition the tree. I Programmatically split the data into 75% and 25% as training and validation sets respectively.

2.2 Modified Generalized Linear model

The following equation is logistic regression for binary response.

$$\log \frac{\pi_i}{1-\pi_i} = \beta_0 + \sum_{j=1}^{p} \beta_p x_{ip}.$$

To obtain the estimator of regression coefficients, maximum likelihood approach is usually used. For this project, we used the maximum likelihood approach as well. However, we combined the method with Bootstrap-like approach instead of usual ML approach to speed up the computation.

Not using the whole observation, but we re-sample M times with sample size $n = 1000$. from the observations and fit logistic regression model for each bootstrap sample, and the sample mean of the model parameter estimators would be our model estimators.

The algorithm for the re-sampling method is as follows:

1. For j = 1, . . . , $M = 1000$,
   (a) Obtain sample of size $n = 1000$.
   (b) Fit logistic regression from the jth sample.
   (c) Store the coefficients of the model.

From the coefficients we stored, we compute the sample mean of the coefficients similar to the bootstrap approach, and take the sample means as our model estimators.

2.3 Deep learning model

This section is about the deep learning part of this course project. By analyzing the twitter comments data, we try to find the relationship between the PartyName(Republican or Democrats) and some features in twitter, such as Retweet-Count, Negative words, Positive words, total token and covered token.

2.3.1 Model Structure

Table 2 shows the model structure of the deep neural network. As the goal of the model is for classification, we use 4 Deep neural network layers. For the first three layers, the output shape is 50, with a linear activation function. The last layer is the binary classification layer, so we use Sigmoid activation function to make the classification. The number of total parameters is 5501.

Table 1: Data

Layer	Output shape	Parameters	Activation Function
Inputlayer	0	0	None
DNN 1	(,6)	350	Linear
DNN 2	(,50)	2550	Linear
DNN 3	(,50)	2550	Linear
DNN 4	(,1)	51	Sigmoid

2.3.2 Model Traning

3 Result

3.1 Classification Tree result

We see from the matrix that the best classification while choosing the minimum observation to split on and minimum observation at a terminal node required is at a varying range of minbucket and minsplit values. 'xerror' which is the cross validation error should not be increasing when we look down the column of a classification tree model result. Otherwise, we risk the chance of choosing parameters that will give us overfitted tree. Keeping in mind that minsplit is the minimum number of observations in a node for which the routine will even try to compute a split, we will choose a minsplit = 1.

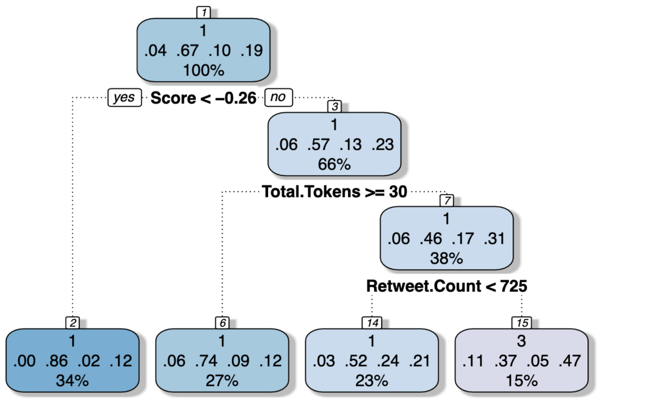


Figure 2: Classification tree Result from training set

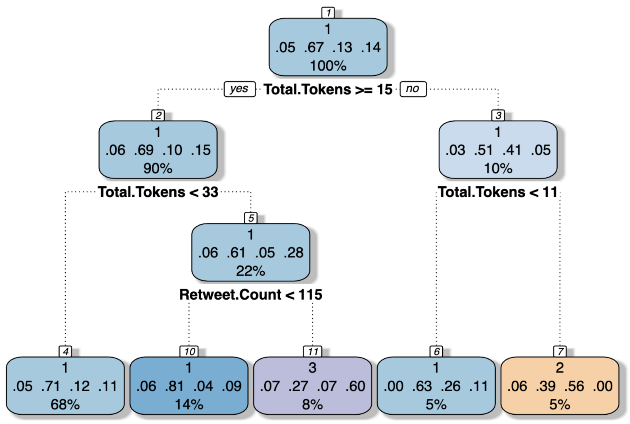


Figure 3: Classification tree Result from validation set

From this, it correctly classifies the data into a Republican majority twitter users. This could be because we don't have a data representative of the population on twitter, or that there was some kind of bias while collecting this data.

If it is the first reason, it could be that since the key role of Republicans, i.e. Trump, is especially active on twitter, more republicans are politically active on twitter than the other parties and therefore, in effect caused bias in this data from twitter.

3.2 Modified Generalized Linear model result

```
## [1] "Summary of data"

##  retweet_count       party               score
##  Min.   :     1   Length:12515164    Min.   :-28.30769
##  1st Qu.:    85   Class :character   1st Qu.: -0.30769
##  Median :   906   Mode  :character   Median :  0.00000
##  Mean   :  6840                      Mean   :  0.03608
##  3rd Qu.:  5431                      3rd Qu.:  0.41026
##  Max.   :625495                      Max.   : 33.38462

## [1] "Summary of logistic regression coefficients"

##   (Intercept)     retweet_count           score
##  Min.   :0.8465   Min.   :-1.634e-05   Min.   :-0.68466
##  1st Qu.:1.0491   1st Qu.:-6.950e-06   1st Qu.:-0.34276
##  Median :1.0990   Median :-4.887e-06   Median :-0.27249
##  Mean   :1.0995   Mean   :-4.684e-06   Mean   :-0.27241
##  3rd Qu.:1.1532   3rd Qu.:-3.029e-06   3rd Qu.:-0.20221
##  Max.   :1.3437   Max.   : 1.615e-05   Max.   : 0.03723

## [1] "Summary of exponenetiated coefficients"

##   (Intercept)    retweet_count     score
##  Min.   :2.331   Min.   :1     Min.   :0.5043
##  1st Qu.:2.855   1st Qu.:1     1st Qu.:0.7098
##  Median :3.001   Median :1     Median :0.7615
##  Mean   :3.012   Mean   :1     Mean   :0.7655
##  3rd Qu.:3.168   3rd Qu.:1     3rd Qu.:0.8169
##  Max.   :3.833   Max.   :1     Max.   :1.0379
```

The raw data has 24201654 observations. To reduce the size of data and to fix the right tail distribution of *Re-tweet counts*, the observations which have zero counts are excluded. Also, to control the multicollinearity, redundant variables are not selected for constructing a model.

From the raw data, 12515164 observations are used which *Re-tweet counts* are non-zero. Since the observations are still too large, it is necessary to use some methods handling the large data. To speed up computation and deal with large data, a re-sampling method is used which is similar to Bootstrap. Unlike Bootstrap, we sample without replacement and the sample size of each sample is smaller than the original.

From the 1000 samples, we can obtain a matrix of logistic regression coefficients.

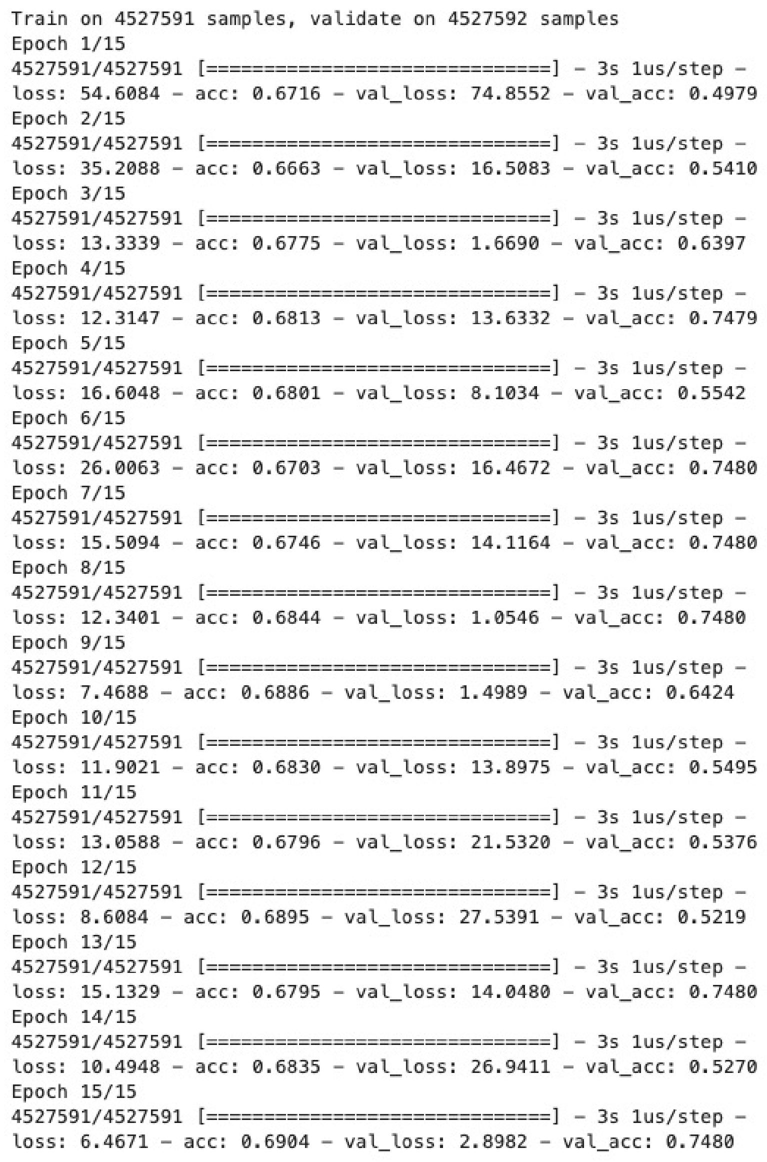
From the result, we can build a logistic regression model as follows:

$$\log \frac{P(Y = Republicans)}{P(Y = Democrats)} = 1.0995 - 4 \times 10^{-6} Retweet - 0.27241 Score \tag{1}$$

where the baseline is *Democrats*, and the odds is defined as the proportion of Republicans to Democrats.

From the model (1), we can notice that the input *Re-tweet counts* is not significant. In other words, there is quite little effect of *Re-tweet counts* to the model. In addition, under fixed *Re-tweet counts*, the odds of Republicans is $exp(1.0995 - 0.27241 \times Score) = 3.012 \times exp(-0.27241 \times Score)$. Therefore, as *Score* increases as one, the odds of Republicans decreases 23.84%. In other words, we can say that increase of the sentiment score of the tweets leads to the decrease of the odds of Republicans, and reversely, the increase of the odds of Democrats.

3.3 Deep learning result



```
Train on 4527591 samples, validate on 4527592 samples
Epoch 1/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 54.6084 - acc: 0.6716 - val_loss: 74.8552 - val_acc: 0.4979
Epoch 2/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 35.2088 - acc: 0.6663 - val_loss: 16.5083 - val_acc: 0.5410
Epoch 3/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 13.3339 - acc: 0.6775 - val_loss: 1.6690 - val_acc: 0.6397
Epoch 4/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 12.3147 - acc: 0.6813 - val_loss: 13.6332 - val_acc: 0.7479
Epoch 5/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 16.6048 - acc: 0.6801 - val_loss: 8.1034 - val_acc: 0.5542
Epoch 6/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 26.0063 - acc: 0.6703 - val_loss: 16.4672 - val_acc: 0.7480
Epoch 7/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 15.5094 - acc: 0.6746 - val_loss: 14.1164 - val_acc: 0.7480
Epoch 8/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 12.3401 - acc: 0.6844 - val_loss: 1.0546 - val_acc: 0.7480
Epoch 9/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 7.4688 - acc: 0.6886 - val_loss: 1.4989 - val_acc: 0.6424
Epoch 10/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 11.9021 - acc: 0.6830 - val_loss: 13.8975 - val_acc: 0.5495
Epoch 11/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 13.0588 - acc: 0.6796 - val_loss: 21.5320 - val_acc: 0.5376
Epoch 12/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 8.6084 - acc: 0.6895 - val_loss: 27.5391 - val_acc: 0.5219
Epoch 13/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 15.1329 - acc: 0.6795 - val_loss: 14.0480 - val_acc: 0.7480
Epoch 14/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 10.4948 - acc: 0.6835 - val_loss: 26.9411 - val_acc: 0.5270
Epoch 15/15
4527591/4527591 [==============================] - 3s 1us/step -
loss: 6.4671 - acc: 0.6904 - val_loss: 2.8982 - val_acc: 0.7480
```

Figure 4: Training result of deep learning model

We use cross-validation to find whether the model is over-fitted or not. The train set and the validation set are randomly split 1:1 for each epoch.

After 15 epoches training, the training loss and the validation loss do not decrease anymore. The training accuracy is stable but validation accuracy frustrate greatly. So this means the model cannot learn anymore from the data. So I think the deep learning model cannot catch the relationship between the response variable and the features.

4 Conclusion

From this, it correctly classifies the data into a Republican majority twitter users. This could be because we don't have a data representative of the population on twitter, or that there was some kind of bias while collecting this data. If it is the first reason, it could be that since the key role of Republicans, i.e. Trump, is especially active on twitter, more republicans are politically active on twitter than the other parties and therefore, in effect caused bias in this data from twitter.