Data Science for Industry Assignment 2

Tiffany Woodley

September 2019

1 Introduction

1.1 Problem Overview

This assignment looks into two methods of text classification, both using feed-forward neural networks and 1-D convolutional neural networks. Both of these methods are applied to a public dataset from the US Consumer Financial Bureau, which contains data about complaints regarding financial products and services.

The two methods as mentioned above will only be given features from the text of the complaints, and have been used in order to determine whether the complaint received compensation or not. Once the two methods have been fit to the data they will be compared on their ability to generalize to new data, and the better of the two will be chosen.

Once the best method has been chosen, it will be used in a R shiny Application that will be created for the purpose of new users being able to input their complaint, and see the likelihood that they would receive compensation or not.

1.2 Data Set

The dataset comprises of the variables:

- The date the complaint was received,
- The type of product the complaint refers to,
- The text explaining the complaint itself,
- Whether the consumer's complaint was compensated.

Since this assignment is only focusing on using text to classify, the variables that will be used are the text narrative of the complaint for input features, and whether the consumer was compensated for that complaint as labels.

The dataset being used is a subset of the original dataset, containing 20 000 complaints. Figure 1 below highlights how unbalanced the data is, by only having 20 percent of the observations refer to complaints that have been compensated. This means the model will need to obtain over 80 percent in order for it to perform better than just classing all of the test observations as false.

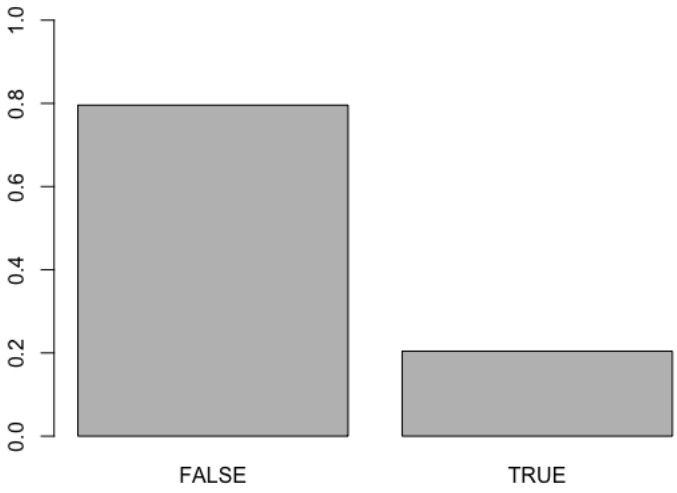

Figure 1: Percentages of true and false observations

There are two ways around this imbalance,

- Under-sampling - removing observations from the false observations until the dataset is balanced,
- Over-sampling - adding duplicates of the true observations until the dataset is balanced.

Since with under-sampling, a lot of the data and information would be lost, over-sampling was used. Although over-sampling does not lose any of the information in the dataset, it does not add any extra information about the true observations.

The data was split 70% training, and 30% test, this gave the test set a fair amount of data to approximate how well the model will generalize. After the data was split, the training set was over sampled to make it balanced for the model to learn from.

2 Feed-Forward Neural Network

2.1 Bag of Words Feature Extraction

2.1.1 Uni-grams

A word list containing all the words that appeared in any of the training complaints was compiled. From this list, all stop words were removed, and filters with regular expressions were applied to the remaining words in order to remove any nonsensical words.

Figure 2 below shows a plot of the frequencies of the words in the word list, from the histogram it can be seen that a large majority of the words only appear once within all the complaint's text. All of the words that appeared less than 10 times in all the complaint's text were removed from the word list, as these words were not very likely to give insights into the general trends over the complaints. This left us with 5936 words.

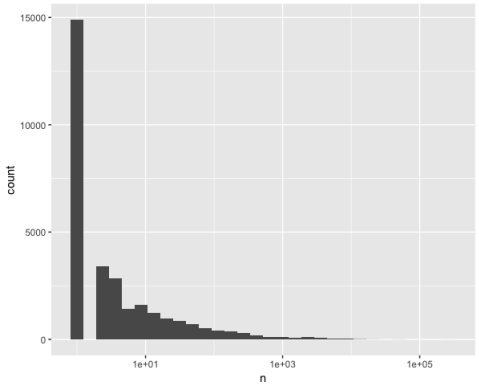

Figure 2: Frequencies of words

Once the word list was filtered, the words were spread across columns in order to create a data frame in the shape shown below.

Compensated	average	abide	abilities	...	zero
TRUE	0	0	2	...	0
FALSE	1	0	1	...	0
FALSE	0	0	0	...	1

2.1.2 bi-grams

A bi-gram list of all the word pairs that appeared in any of the training complaints was compiled, from this list any bi-grams containing stop words or nonsensical words were removed.

Figure 3 below shows a plot of the frequencies of the bi-grams in the bi-gram list, from the histogram it can be seen that a large majority of the bi-grams only appear once within all the complaint's text. All bi-grams that appeared less than 10 times in all the complaint's text were removed from the bi-gram list, as these word are not very likely to give insights into general trends over the complaints.

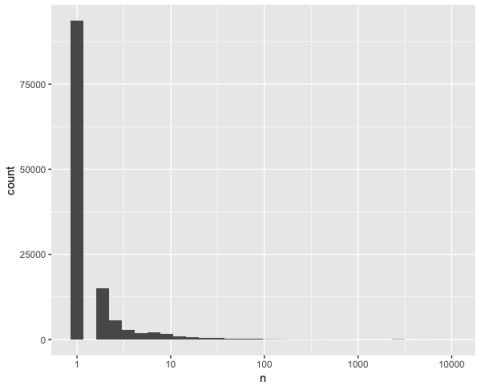

Figure 3: Frequencies of bi-grams

This left us with 3776 bi-grams. From this, each bi-gram was looked at in order to find the log-likelihood that the first word preceding the second word gave any more meaning to the second word. The log likelihood was then transformed into a X^2 distribution by applying chi-squared distribution function to the log likelihood. The p value was then found and any bi-grams with a p-value of over 0.05 were removed from the b-gram list leaving 697 bi-grams.

These bi-grams were then added on as columns to the complied data frame containing the uni-grams. This gave the feed-forward network 6633 inputs.

2.2 Fitting the model

When fitting the model the following hyper-parameters were tweaked

Hyper- parameter	Values
Number of layers	1,2,3
Number of neurons	5,10,15
Batch size	16,50,100
L2 regularization(lambda)	0.001,0.001

A grid search approach was applied to narrow down these hyper-parameters, the results were compared on the best validation accuracy over 10 epochs. The metric used to compare on the the validation set was classification accuracy as the training set given to the model was oversampled to be balanced. The Figure

below shows the results of the grid search.

The figures show how the grid search performed with different number of layers in the network, the point of this was to find the general area of where the model architecture should be before fine tuning. The best model according to the validation accuracy found was 85.28 %, which was using 3 layers, 10 neurons per layer, a batch size of 16 and a lambda of 0.001. Although this architecture produced the highest validation accuracy a model with 2 layers of 5 neurons, a batch size of 16, 10 epochs and lambda of 0.001 with a validation accuracy of 83,68 % was chosen. The reason for this choice is that there is a slight decline in validation accuracy for a big decrease in complexity of the model, this means the model will have a better chance at generalizing.

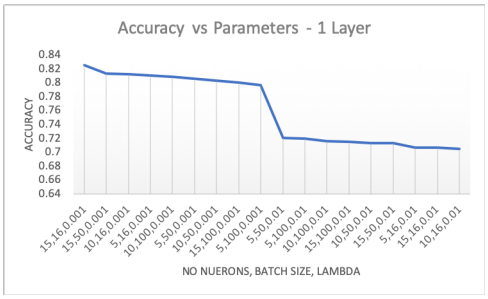

Figure 4: Grid search on 1 layer feed forward network

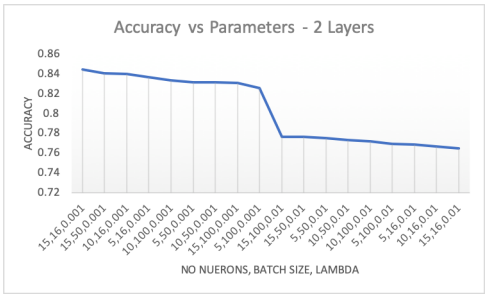

Figure 5: Grid search on 2 layer feed forward network

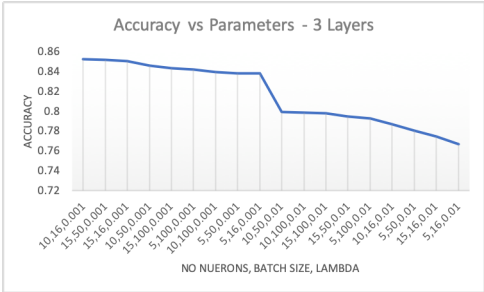

Figure 6: Grid search on 3 layer feed forward network

The hyper-parameters chosen above were from quite a coarse grid search, and so another grid search was run on a finer scale around these parameters chosen above. This grid-search's results can be viewed in Figure 7, and the since it performed best the previous chosen architecture of 2 layers of 5 neurons, a batch size of 16, 10 epochs and lambda of 0.001 was kept.

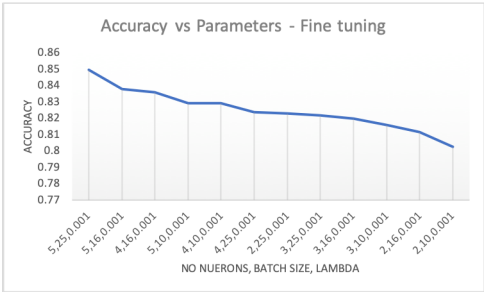

Figure 7: Grid search on fine-tuning feed forward network

The figure below shows the history of the final fitted model, showing how the model fits as the epochs increase. From Figure 8, 5 epochs were chosen as it is at this point were the accuracy starts evening out and could be an indication of the model starting to fit to noise.

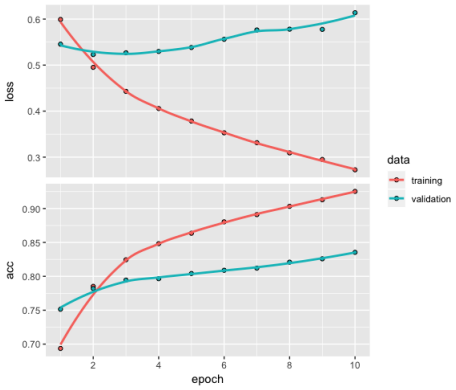

Figure 8: Feed Forward network- accuracy vs epoch

3 1-D Convolutional Neural Network

3.1 Tokenization and Embedding

A word list containing all the words that appeared in any of the training complaints was compiled. From this list all stop words were removed, and filters with regular expressions were applied to the words to removed any nonsensical words.

Sequences of numbers that represent the words in the complaints in terms of the tokenized words in the word list were then created.

For example, if the tokenized word list looked something like this

1	hello
2	are
3	you
4	how

The sequence of a complaint "Hello, how are you?" would look like

[1 4 2 3]

Since all complaints are of different lengths, they are then all padded with zeros until they are all of length 250. They need to be the same length as each will be an input into the network.

These padded sequences are then fed into an embedding layer, which in turn creates a vector for each word which increases the width of the vectors given to the convolutional layer. The size of vector used for this exercise was 20. Which would result in 250 vectors of 20 as the input - 5000 inputs.

The use of bi-grams, tri-grams etc is not necessary in this case as because consecutive words are below or above each other, when the filters move down them they will pick up on relationships between consecutive words.

3.2 Fitting the model

When fitting the model the following hyper-parameters were tweaked

Hyper- parameter	Values
Number of layers - dense	2
Number of layers - convolution	1,2
Number of neurons	5,10,15
Batch size	16,50,100
Kernel size	2,3
No of filters	16,32,48
L2 regularization(lambda)	0.001,0.01

Figure 9 below shows the performance of a model, with 1 1D convolutional layer, followed by 2 dense layers. The model which performed best in this grid search (on the validation set 91.9%) had 5 neurons in the dense layers, a batch size of 16, a kernel of 3 3, lambda of 0.001 and 16 filters.

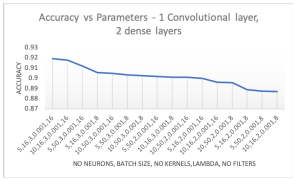

Figure 9: Grid search on 1 1D layer convolutional network

The above mentioned architecture was then extended to have an extra 1D convolutional layer, with the number of filters as seen in the table below. Adding in a second convolutional layer did not improve the accuracy as seen in the table, and so an extra 1D convolutional layer was not added.

Filters in layer 1	Filters in layer 2	Validation Accuracy
16	8	0.8857143
8	4	0.866567

Since this architecture was on the lower end of the grid search, smaller architectures were looked into in the table below.The table shows the results of the previous mentioned model with the number of neurons in the dense layers changed. The smaller architectures performed closely to the architectures with more neurons, the small amount of reduced validation accuracy was worth the simpler model. And so the final model chosen had 3 neurons in two dense layered, preceded by a convolutional layer with 16 filters, a kernel size of 3 and lambda of 0.001.

Number of neurons	Validation Accuracy
2	0.8896672
3	0.8989416
4	0.9062715
5	0.9190726

Looking at Figure 10 the validation accuracy starts evening out at 4 epochs, afterwards chances are the model is starting to fit to noise, and so 4 epochs was chosen.

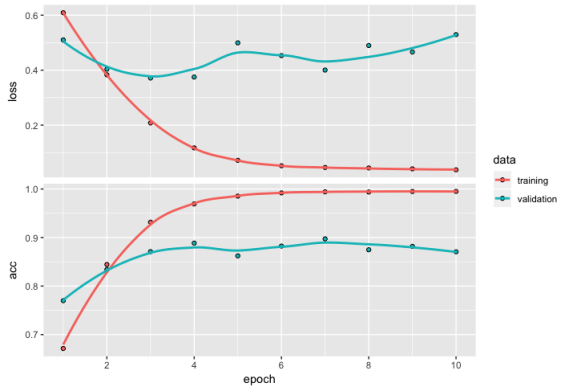

Figure 10: 1D Convolutional network- accuracy vs epoch

4 Out of Sample performance

The table below shows the classification accuracy of the two models on the test set, since the test set had never been seen by the model before it evaluated on it, it is the best indication to how well the model performs out of sample.

Model	Test Acc
Feed-Forward	0.7238793
Convolutional	0.7539154

Since the test set was not over sampled and therefore still being unbalanced, classification accuracy is a biased metric. And so the results were looked into further by using a confusion matrix.

The table below shows the confusion matrix for the 1D convolutional neural network. Initially it is clear to see that the model performs much better on the false observations. The specificity (actual false observations which were predicted correctly) was 36.13%, the sensitivity (actual true observations which were predicted correctly) was 85.47%. The specificity and sensitivity further show how the model performs better on the false observations.

	0	1
0	4082	694
1	783	443

The table below shows the confusion matrix for the feed-forward neural network. The specificity (actual false observations which were predicted correctly) was 44.70%, the sensitivity (actual true observations which were predicted correctly) was 81.34%. The feed-forward neural network performs better than the 1D convolutional network on predicting true observations, but worse at predicting false observations.

If the data that the model would be performing on would be balanced, this would be the better model overall.

	0	1
0	3796	979
1	678	548

Both models perform poorly on the true observations, and so the number of true observations fed into the model were not enough to give the model enough information on general trends of the compensated claims. This could be improved by sampling more true observations from the original main dataset, rather than oversampling the true observations in the subset we were working with.

Since the subset was randomly sampled from the main dataset, which contained all the claims, the unbalanced distribution of the subset is likely to follow the distribution of all the complaints. And so based on the ratio of complaints coming in, the best chance of getting the prediction of all complaints coming in would be to predict them all to be false. Since the models can predict true observations to some extent they will perform better on balanced complaints coming in.

Between the two models, the 1D convoltional model was chosen, despite it not being as accurate on the true observations, it has a higher accuracy on false observations which have a higher probability of being entered into the application.

5 Shiny Application

The figures below show the shiny application created using the chosen model. The figures below show it's functionality when there is a false complaint, no complaint and true complaint entered.

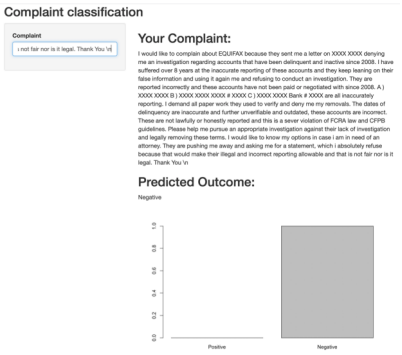

Figure 11: Shiny App- False Complaint

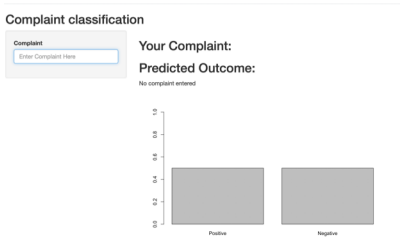

Figure 12: Shiny app- No Complaint

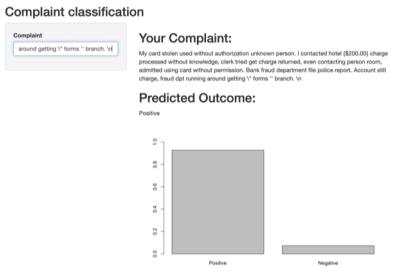

Figure 13: Shiny App- True Complaint