UIS UNIVERSITY OF ILLINOIS SPRINGFIELD

TOPIC:

MULTICLASS CLASSIFICATION FOR QUERY TOPIC PREDICTION

BY

DISHA SHETIYA (dshet4@uis.edu)

GAURAV PATIL (gpati4@uis.edu)

ABSTRACT

The challenge of categorizing examples into one of three or more classes is known as multiclass classification or multinomial classification in machine learning and statistical classification, whereas categorizing instances into one of two groups is known as binary classification. The goal of the multiclass classification, a common machine learning challenge, is to categorize every occurrence into one of a preset set of classes. A classification model generates scores for each class based on an instance and uses those scores to order the classes. Top-K Accuracy (K=1 or 5) is a common metric used to assess a classification model's performance. Because there are several possible classes for shades of color that each image might fall under, using a model to identify various hues in a painting is an example of a multiclass categorization.[2] A sample can only have one class in multiclass categorization. Predicting the subject of a certain query for the yahoo answers dataset is the project's principal objective. There are many methods to solve the problem of multiclass classification i.e., by Neural Networks, Random Forest, Naïve Bayes, k-nearest neighbors, Support Vector Machine and many more.[3] But for this project we would be using the neural network model.

CONTENTS

LIST OF FIGURES

CHAPTER 1

PROBLEM DEFINITION AND PROJECT GOAL

Given a question asked on Yahoo Answers we are trying to predict the topic of a certain query. The dataset which is used for this project was taken from KAGGLE. Various levels such as Society & Culture, Science & Mathematics, Health, Education &References, Computers & Internet, Sports, Business & Finance, Entertainment & Music, Family & Relationship, Politics & Government . All these levels are numbered from index 1 – 10. Various features such as Class Index, Question Title, Question Content and Best Answers are in the dataset. Then we are cleaning the data using Corpus and removing the stop words, punctuations, blank spaces. Then we are vectorizing the data and then training our model.

CHAPTER 2

RELATED WORK

- Examining Multiclass Classification through the Lens of Learning to Rank Nan Wang, Zhen Qin, Le Yan, Honglei Zhuang, Xuanhui Wang, Michael Bendersky, Marc Najork

  This study suggests a way for improving accuracy in multiclass classification issues. The issue is that in cases when there are many classes, standard approaches struggle to appropriately distinguish fresh observations due to the large number of options. The concept is to create specialized classifiers for classes that exhibit more common errors, i.e., to create a chain of specialized classifiers for easier issues. As a result, the approach is based on class hierarchy inference. The similarity matrix is derived using the confusion matrix generated from the train data, and the hierarchy of classifiers is derived using an agglomerative clustering algorithm. Given a confusion matrix, we present a semi-metric for calculating distances between classes.

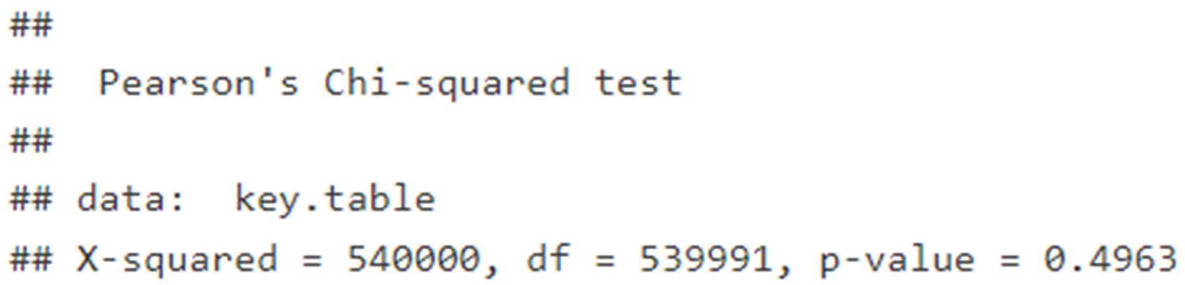
- A Review of Multi-Class Classification for Imbalanced Data Mahendra Sahare, Hitesh Gupta

  This paper presents a multi-class categorization survey for data imbalancing. In this paper a few issues regarding multiclass classification are discussed. Critical issues discovered in the review include issues with the feature selection procedure of the classifier and an endless population of data. Such an issue arises from the implementation of binary classifiers in the form of liner classifiers. The initial strategy was directly handling the multiclass scenario by extending binary classification problems. This comprised k-nearest neighbors, naive bayes, decision trees, support vector machines, and neural networks. The second method divides the issue into various binary classification jobs. One-Versus-All (OVA), All-Versus-All(AVA) are some of the techniques utilized for this breakdown. The third method included placing the classes in a tree—typically a binary tree—and applying several binary classifiers to the tree's nodes until a leaf node was reached. In the future, we reduced the issue with binary classifier feature reduction and error-correcting code.

CHAPTER 3

DATA EXPLORATION AND PREPROCESSING

The dataset is primarily composed of four attributes: Class Index, Question Title, Question Content, and Best Answers. The dataset has 60000 rows, and the property Class Index is of integer type, whereas Question Title, Question Content, and Best Answers are of categorical type. We discovered that there are no missing values in the dataset after searching for them. The Class Index is our outcome variable. The properties 'Question Title,' 'Question Content,' and 'Best Answers' have been consolidated into a single textual column named 'Q&A.' After merging the characteristics, we were left with just two variables to deal with: the 'Class Index,' which is our result variable, and the 'Q&A,' which is the combined attribute. We used the Chi-square test to look for a link between the two attributes, and the p-value was 0.49.



```
##
##  Pearson's Chi-squared test
##
## data:  key.table
## X-squared = 540000, df = 539991, p-value = 0.4963
```

Fig. 3.1 Chi-Square Test

We used the Corpus function to clean and preprocess the data. We removed all the stop words, punctuations, and white strips from the dataset. After that we formed the word cloud. We also formed word cloud for all the class types from 1 to 10. We used DTM (Document Term Matrix) to find the most frequent words from the Corpus. We did split the data into 3 parts: for training (1:30000), validation (30001:40000) and testing (40001:60000). After looking for data imbalance it was found out that was pretty much balanced as seen from the below figure. The data balancing was found out using the caret library.

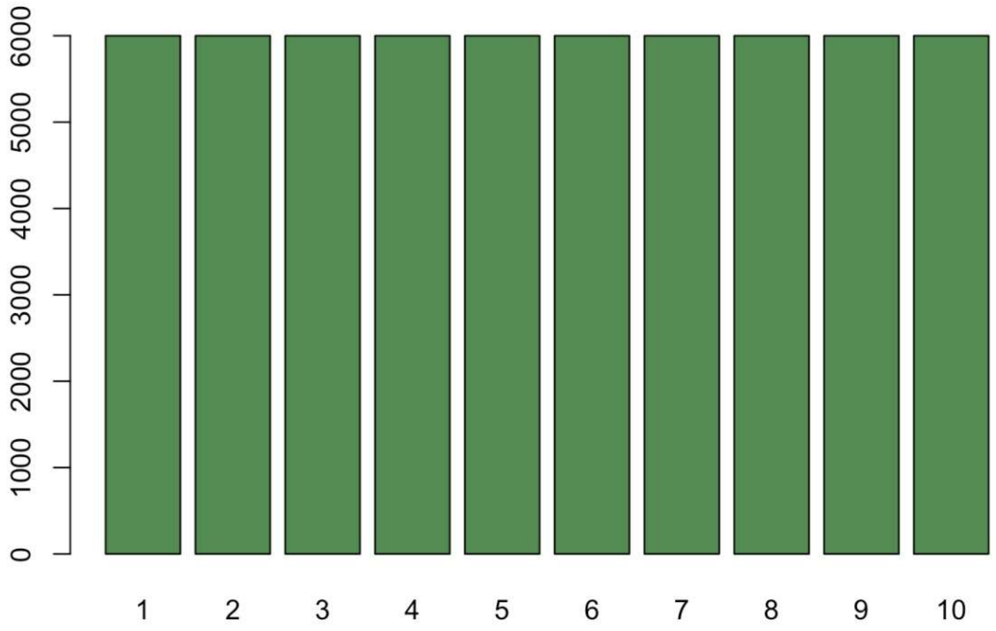


Fig. 3.2 Data Balancing

Word clouds of the types from 1 to 10 of the Class Index attribute were created to visualize the text data in the types. Following are the word cloud of all the types:

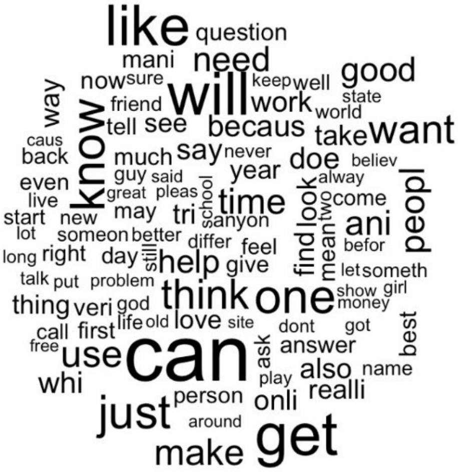


Fig. 3.3 Word Cloud type = 1

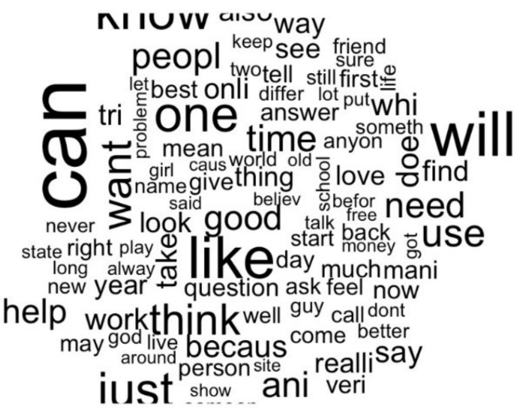


Fig. 3.4 Word Cloud type = 2

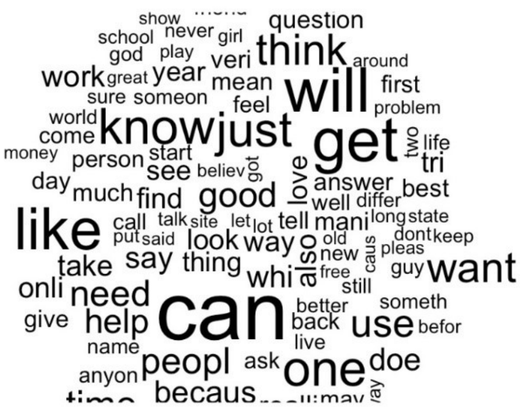


Fig. 3.5 Word Cloud type = 3

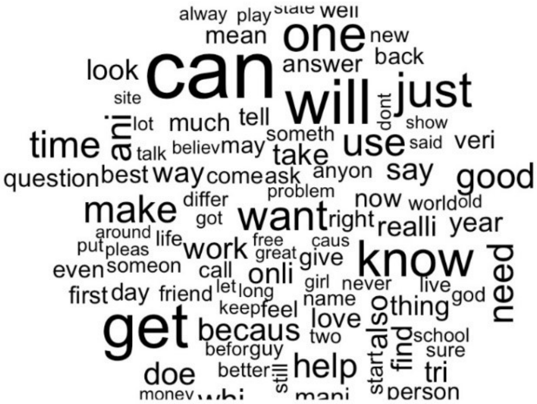


Fig. 3.6 Word Cloud type = 4

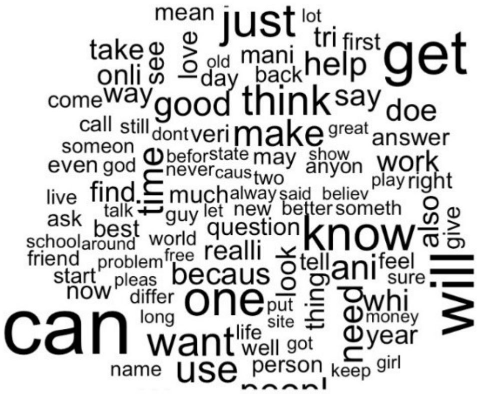


Fig. 3.7 Word Cloud type = 5

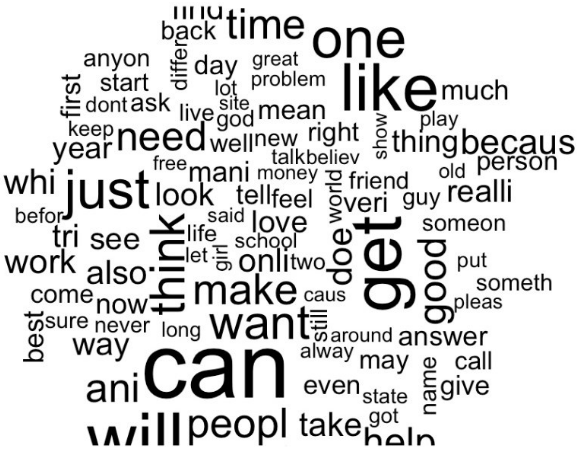


Fig. 3.8 Word Cloud type = 6

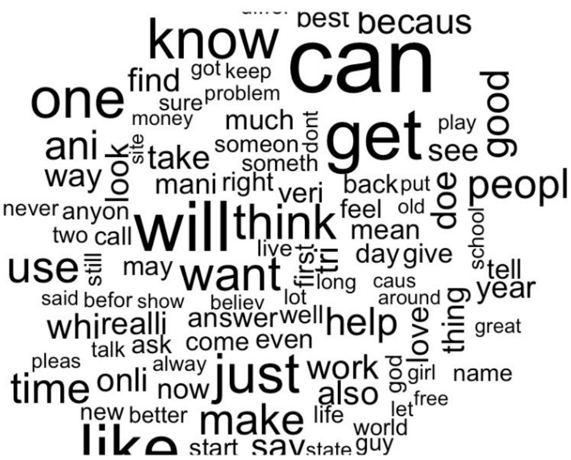


Fig. 3.9 Word Cloud type = 7

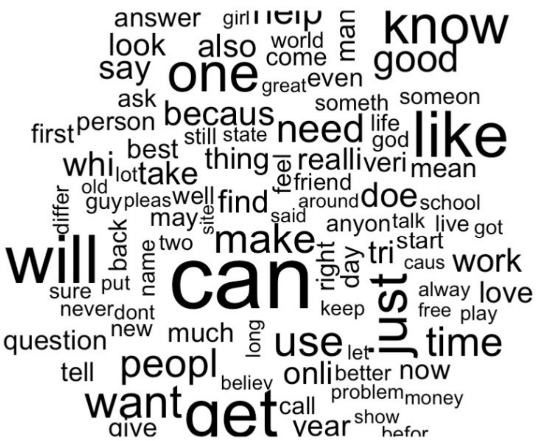


Fig. 3.10 Word Cloud type = 8

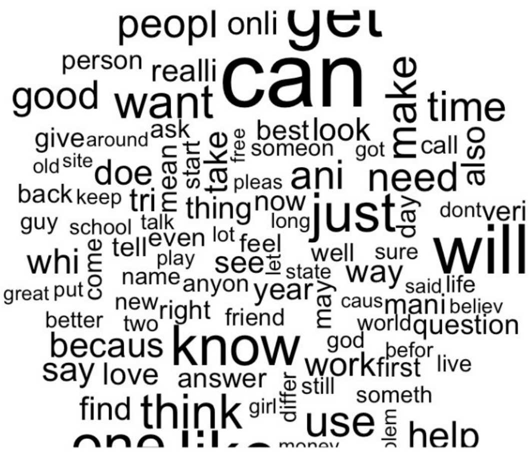


Fig. 3.11 Word Cloud type = 9

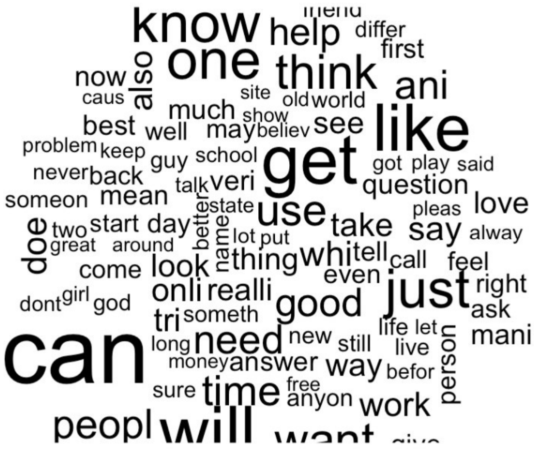


Fig. 3.12 Word Cloud type = 10

Thus, the visualized data can be observed from the word cloud. We can see from the word cloud that terms like can, will, get, like, know, one, good, and people are frequent across all sorts.

CHAPTER 4

DATA ANALYSIS AND EXPERIMENTAL RESULTS

We are training a neural network model for this multiclass classification, and we are using two layers of the neural network. We used text vectorization to transform text data to numerical vectors to assist create the machine learning model after dividing the data into three sections (training, validation, and testing). The primary goal of this is to create the processing model for the dataset. This allows the function to operate on all items of a vector without having to loop over and act on each element individually. Later, we train our neural network model with the 'adam' optimizer and the loss function sparse_categorical crossentropy' for our dataset. The ‘sparse_categorical_crossentropy’ helps us to work on the categorical data with more than 2 levels without the need to one-hot-encode the data. After we fit the data following plot is observed:

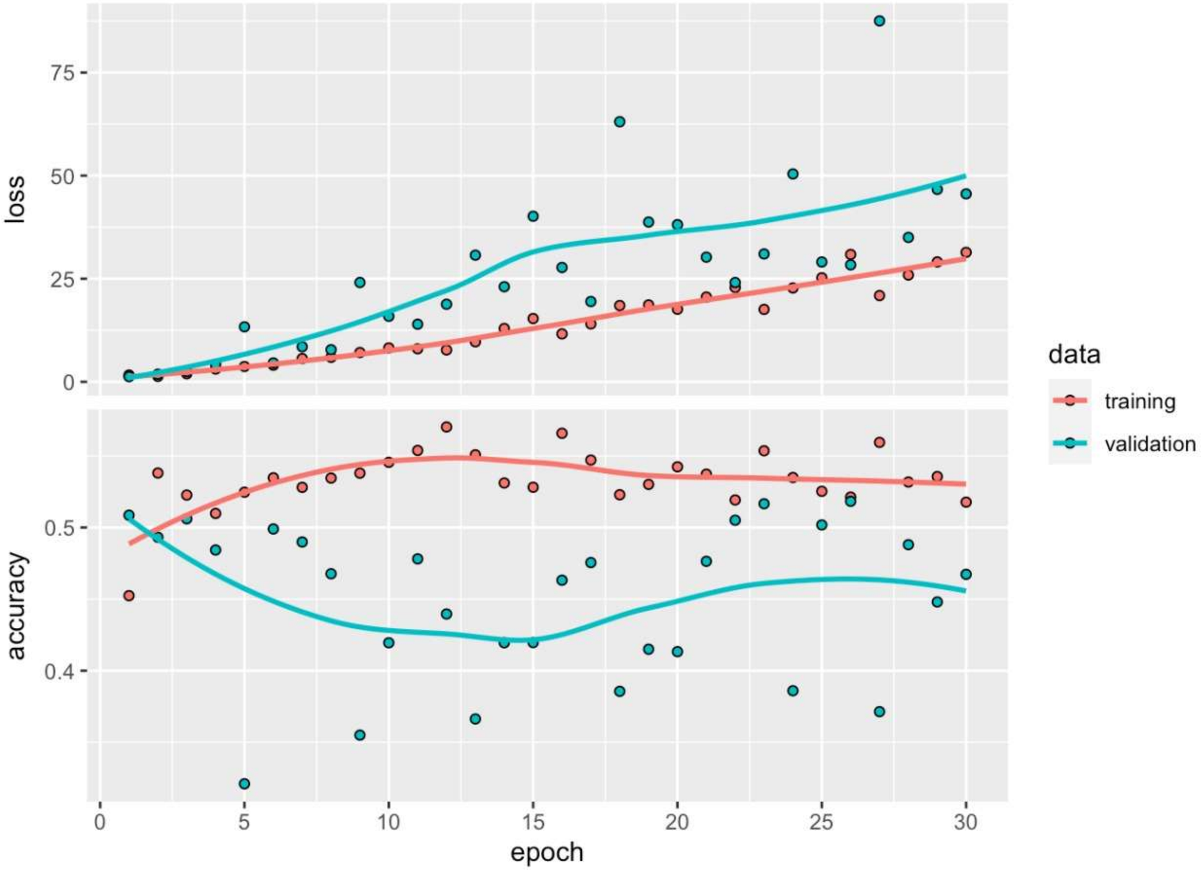
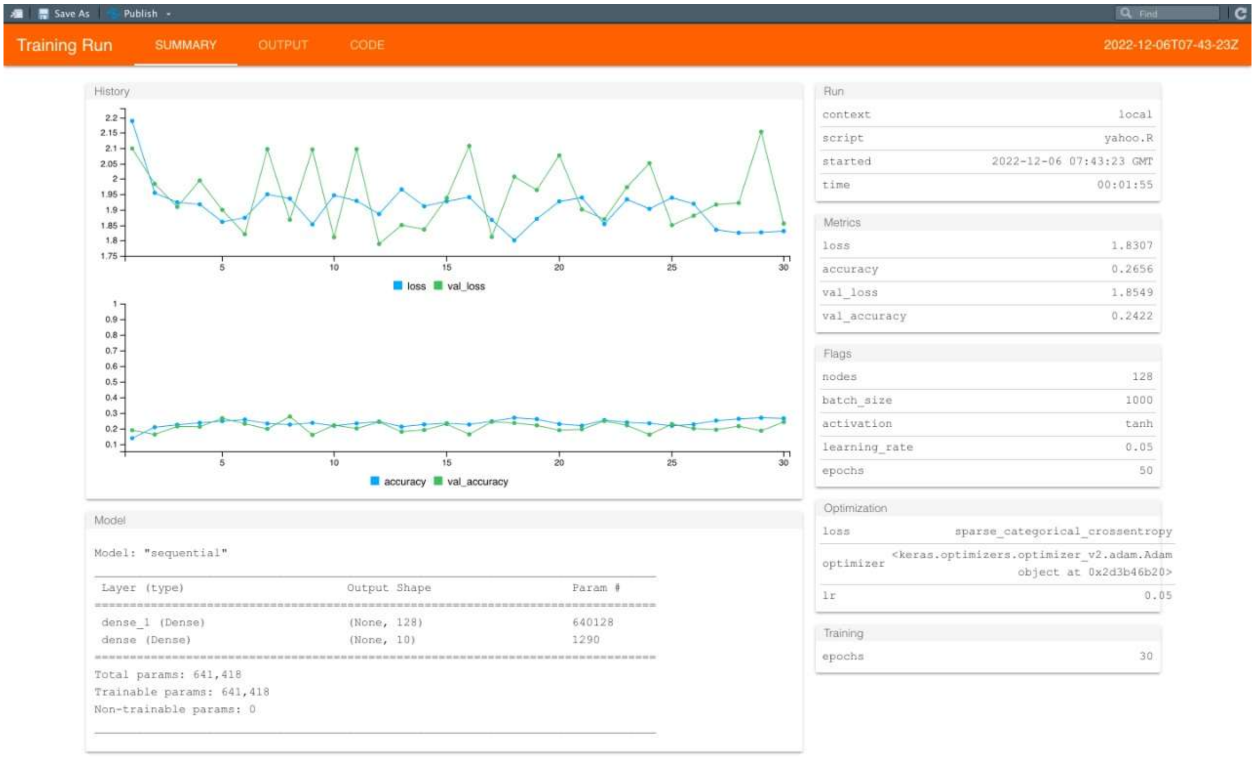


Fig. 4.1 plot history 1

We get the following results after computing the accuracy and loss percent: loss = 32.99747 and accuracy = 0.5077. As we can see from the results, after fitting our model, we achieve an accuracy of about 50%. We will later forecast the Class Index of the data based on the preceding results. We utilized a R script to fine-tune our model using learning rates of 0.01, 0.05, 0.001, and 0.0001. The tuning batch sizes were 100, 200, 500, and 1000. The activation functions employed we are using 'relu,' 'sigmoid', and 'tanh.' The model executed 288 flag combinations in total, and the data was sampled up to 6 combinations. The directory with the best results from the six runs was dir[4].



Run

context	local
script	yahoo.R
started	2022-12-06 07:43:23 GMT
time	00:01:55

Metrics

loss	1.8307
accuracy	0.2656
val_loss	1.8549
val_accuracy	0.2422

Flags

nodes	128
batch_size	1000
activation	tanh
learning_rate	0.05
epochs	50

Optimization

loss	sparse_categorical_crossentropy
optimizer	<keras.optimizers.optimizer_v2.adam.Adam object at 0x2d3b46b20>
lr	0.05

Training

epochs	30

Model

```
Model: "sequential"
________________________________________________________________________________
 Layer (type)                        Output Shape                     Param #
================================================================================
 dense_1 (Dense)                     (None, 128)                      640128
 dense (Dense)                       (None, 10)                       1290
================================================================================
Total params: 641,418
Trainable params: 641,418
Non-trainable params: 0
________________________________________________________________________________
```

Fig 4.2 Training Run

Later, we further train our model using ‘tanh’ and ‘softmax’ activation function. Here the learning rate was set to 0.05 which we got from the above result. After fitting the model, we plotted the following graph:

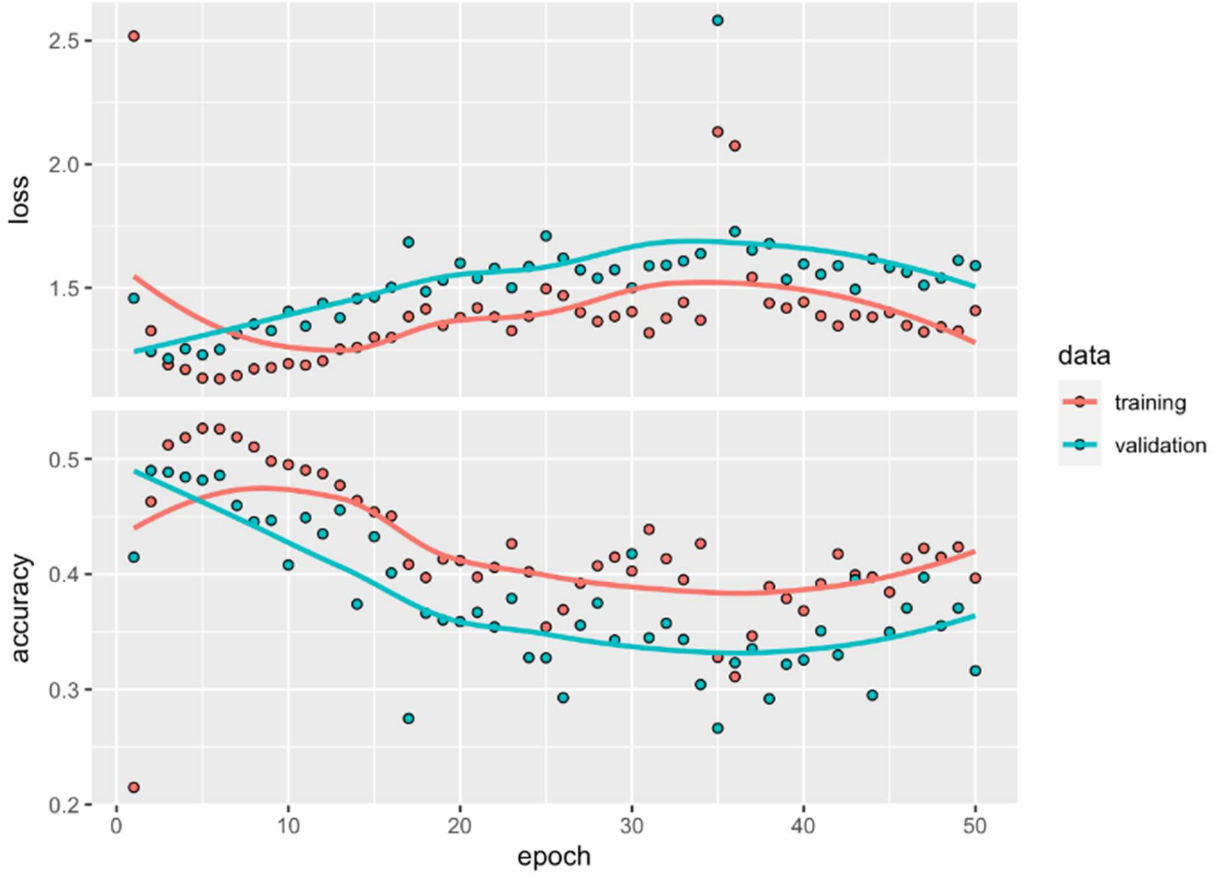
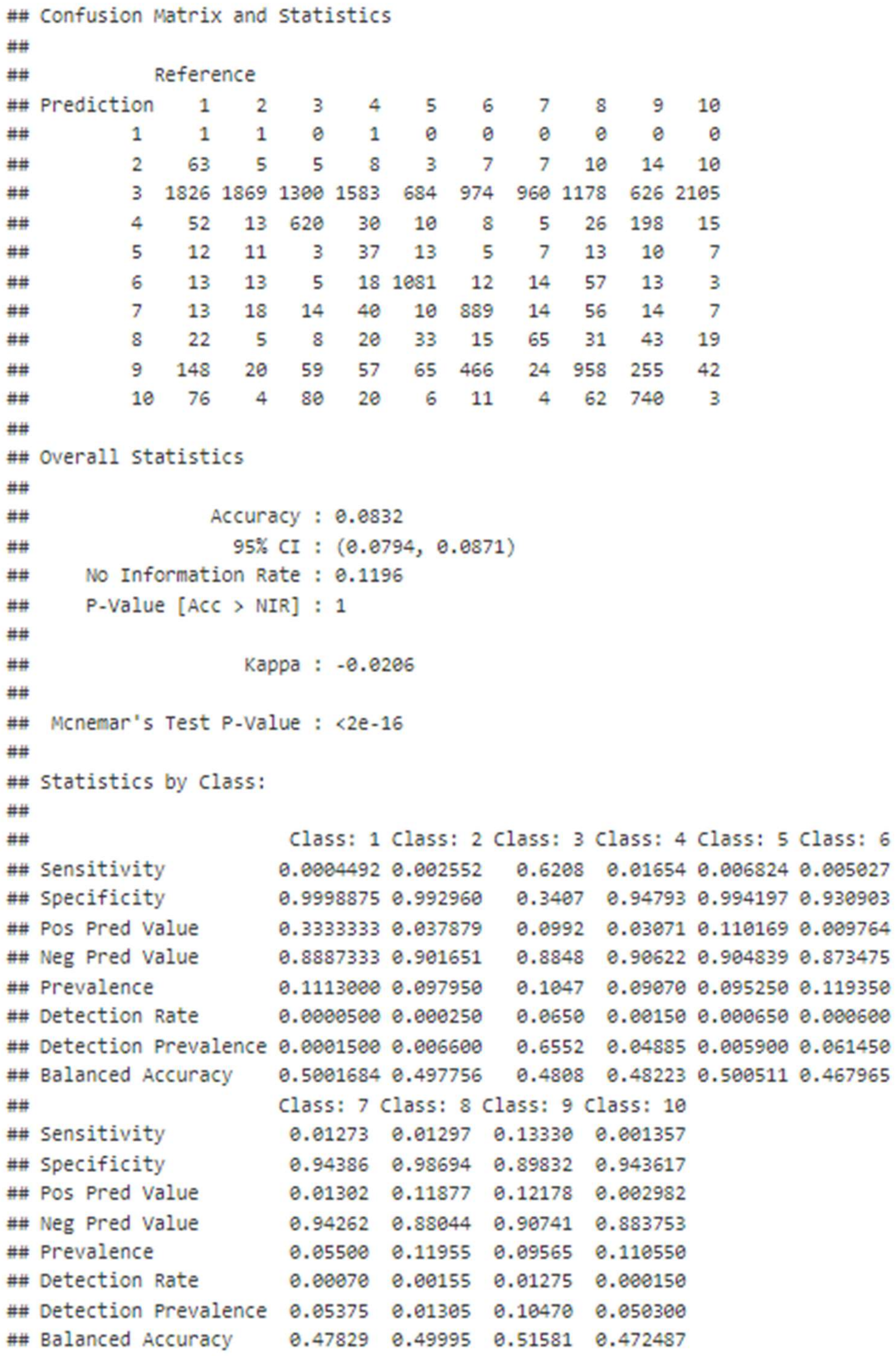


Fig. 4.3 plot history 2

For the evaluation we used the confusion matrix to find out the accuracy of the model and the kappa value.



```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    1    2    3    4    5    6    7    8    9   10
##         1     1    1    0    1    0    0    0    0    0    0
##         2    63    5    5    8    3    7    7   10   14   10
##         3  1826 1869 1300 1583  684  974  960 1178  626 2105
##         4    52   13  620   30   10    8    5   26  198   15
##         5    12   11    3   37   13    5    7   13   10    7
##         6    13   13    5   18 1081   12   14   57   13    3
##         7    13   18   14   40   10  889   14   56   14    7
##         8    22    5    8   20   33   15   65   31   43   19
##         9   148   20   59   57   65  466   24  958  255   42
##         10   76    4   80   20    6   11    4   62  740    3
## 
## Overall Statistics
## 
##                Accuracy : 0.0832          
##                  95% CI : (0.0794, 0.0871)
##     No Information Rate : 0.1196          
##     P-Value [Acc > NIR] : 1               
## 
##                   Kappa : -0.0206         
## 
##  Mcnemar's Test P-Value : <2e-16          
## 
## Statistics by Class:
## 
##                       Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity          0.0004492 0.002552   0.6208  0.01654 0.006824 0.005027
## Specificity          0.9998875 0.992960   0.3407  0.94793 0.994197 0.930903
## Pos Pred Value       0.3333333 0.037879   0.0992  0.03071 0.110169 0.009764
## Neg Pred Value       0.8887333 0.901651   0.8848  0.90622 0.904839 0.873475
## Prevalence           0.1113000 0.097950   0.1047  0.09070 0.095250 0.119350
## Detection Rate       0.0000500 0.000250   0.0650  0.00150 0.000650 0.000600
## Detection Prevalence 0.0001500 0.006600   0.6552  0.04885 0.005900 0.061450
## Balanced Accuracy    0.5001684 0.497756   0.4808  0.48223 0.500511 0.467965
##                      Class: 7 Class: 8 Class: 9 Class: 10
## Sensitivity           0.01273  0.01297  0.13330  0.001357
## Specificity           0.94386  0.98694  0.89832  0.943617
## Pos Pred Value        0.01302  0.11877  0.12178  0.002982
## Neg Pred Value        0.94262  0.88044  0.90741  0.883753
## Prevalence            0.05500  0.11955  0.09565  0.110550
## Detection Rate        0.00070  0.00155  0.01275  0.000150
## Detection Prevalence  0.05375  0.01305  0.10470  0.050300
## Balanced Accuracy     0.47829  0.49995  0.51581  0.472487
```

Fig. 4.4 Confusion matrix

Thus the final accuracy we got after the entire training and evaluating process was 0.316100 with a loss of 1.589522.

CONCLUSION

Multiclass Classification is a challenging work since we had to classify 10 themes and get a high accuracy, but the dataset at hand was not very helpful in reaching a better result. To attain a better outcome, the dataset should be relatively large but also include more detail on how the data is classified into distinct subjects.

The model employed in the research did not function as predicted, although it nevertheless achieved a reasonable level of accuracy. It was discovered that it predicted the type '7' class index, i.e., Business and Finance, far better than the other classes. In the future we can involve much more attributes to help us classify the class indices to help our model to predict the topic more accurately.

REFERENCES

1. https://www.kaggle.com/datasets/bhavikardeshna/yahoo-email-classification?select=test.csv (FOR DATASET)
2. https://www.datarobot.com/blog/multiclass-classification-in-machine-learning/
3. https://medium.com/dataman-in-ai/a-wide-choice-for-modeling-multi-class-classifications-d97073ff4ec8