**CS 6301-004: Special Topics in Computer Science (R for Data Scientists)**

**Assignment 2: Text** **Classification**

**Team Members:**

**Teja Kiran Chunduri (txc163430)**

**Prathyusha Kanmanth Reddy(pxk166230)**

**Dataset Used:**

We used the SMS Spam Collection dataset which is a public set of SMS labeled messages that have been collected for mobile phone spam research.

Dataset link: <https://archive.ics.uci.edu/ml/machine-learning-databases/00228/>

Number of instances: 5574

Number of columns: 2

File format: TSV

**Methodology:**

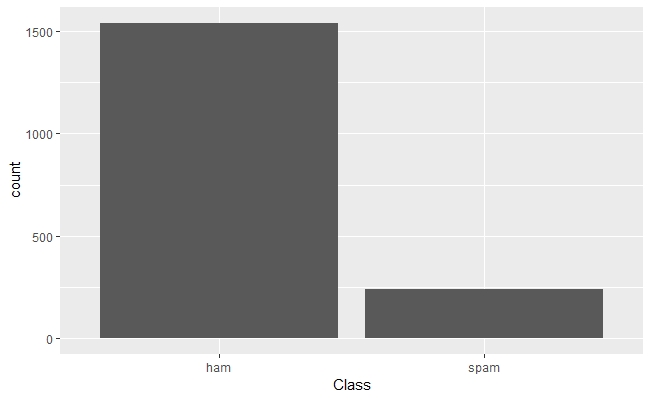
The dataset consists of a large chunk of messages each labelled as Spam or ham (Not Spam).

As with any neural nets, we need to convert our data into numeric format. Fortunately, keras comes with lot of functions that can achieve this. To clean the raw dataset, we have used the “tm” and “SnowballC” libraries.



Whether a message is Spam or not will be the response variable or classification label for the messages.

The distribution of the class variables is as follows:

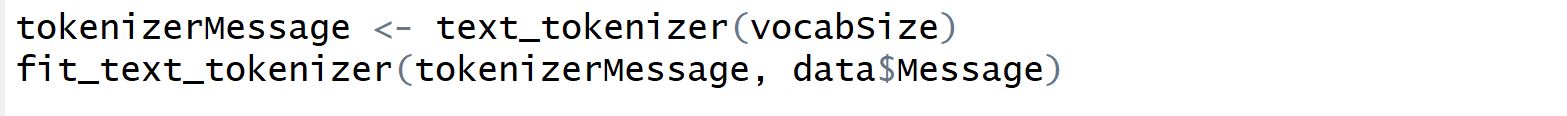


**Preprocessing the Dataset:**

1. The class variables (“Ham” and “Spam”) are converted into binary 0 or 1. 1 represents a spam message and 0 refers to ‘Not Spam’
2. The entire Corpus is then converted to UTF8 format. This would get rid of any non ascii characters. This would not only get rid of unnecessary data but is also a necessary step since the “tokenizer” which we will be using later cannot process non ascii information.
3. The following processing techniques are applied on Text Corpus:

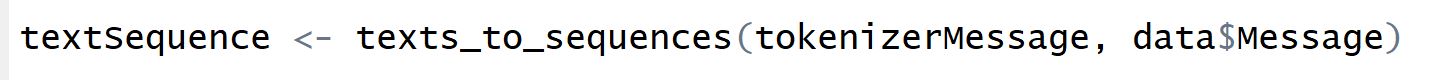
* Used tm library to handle all the preprocessing.
* Replaced hyphens, colons and single quotes with space
* Removed punctuations
* Converted the entire corpus to lowercase letters
* Removed all numbers and stop words
* Stripped extra white spaces
* Stemmed the entire document
* Converted class from categorical to numerical (binary) by writing a function
* Renamed the dataset columns with proper names for easy access

1. Now the preprocessed text gives a sequence of words. To feed it to the neural net, this needs to be converted to sequence of integers, where each word stands for a specific word in a dictionary
2. First, we’ve built a Tokenizer with a fixed number of words (1000) which is our feature space or dictionary (1000 most frequent words in our corpus) and fit it to the text data.

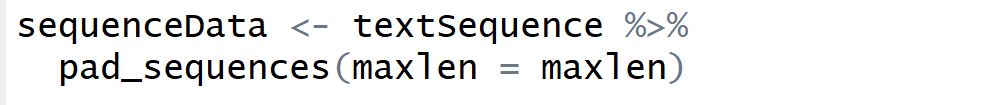


1. Now that we have the dictionary of integers and their corresponding word mappings, we have used texts\_to\_sequences function to convert each message into list of integers.

At this point, words that were not in the dictionary will be discarded



1. Now, we will have a list of integers of different lengths for each message. Since, neural net expects all the lists to be of equal length, we have padded the lists using pad\_sequences function which takes a maxlen(fixed size for all lists) argument.



**Model Building and Data Validation**

We’ve split the processed data into train(80%) and test(20%) datasets. Splitting the train data into train and validation is taken care in model.fit function with validation\_split attribute.

The following section lists the parameters used in each run and respective plot of history.

**Parameters Used:**

**RUN 1:**

Activation function used in input and hidden layer: Relu

Activation function used in output layer: Sigmoid

dropout rate in input layer: 0.2

dropout rate in hidden layer: 0.3

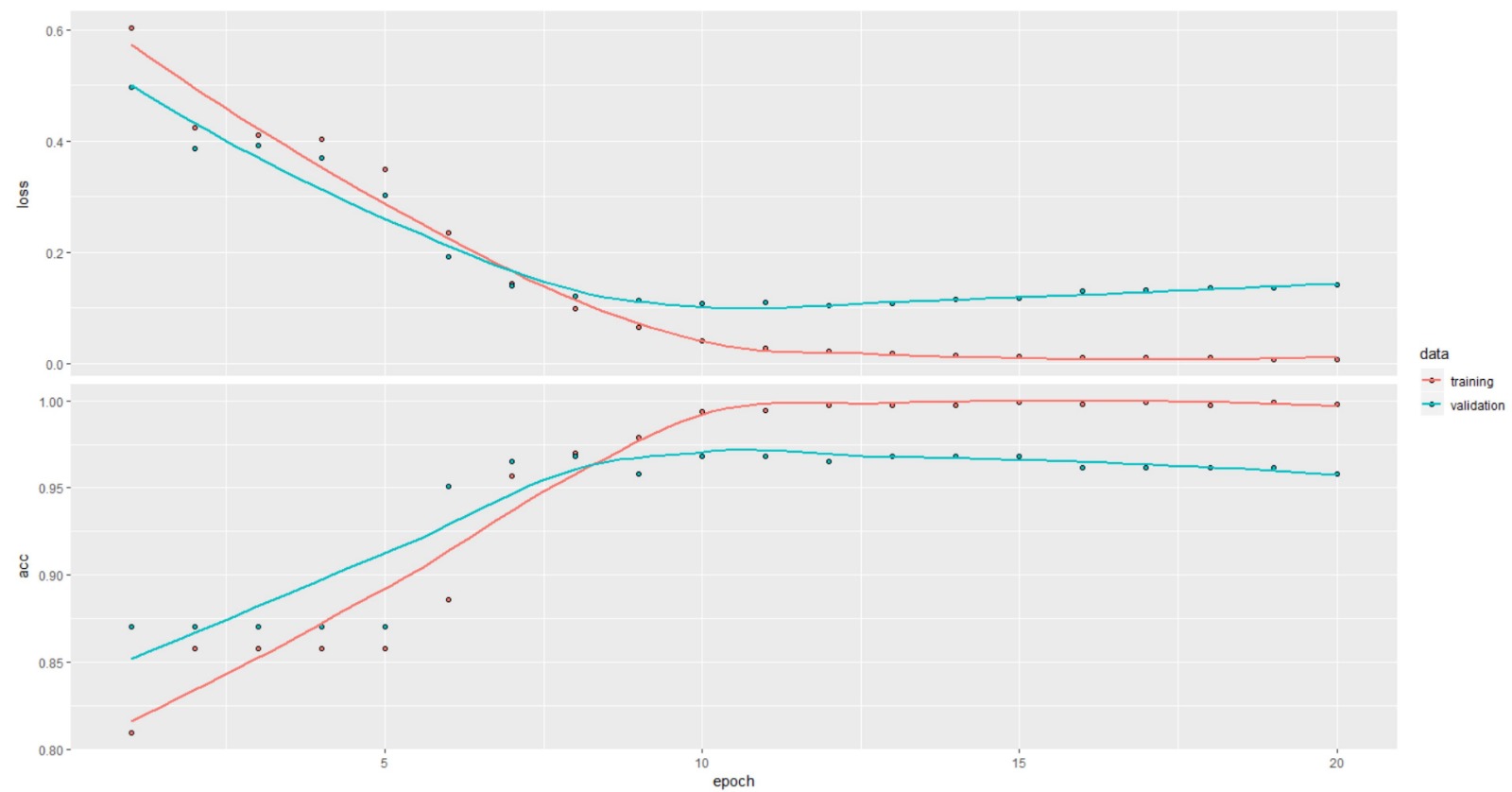
Batch size: 50

Epochs: 20

Validation split: 0.2

Accuracy Obtained: **97.85**

Plots: History plots



**RUN 2:**

Activation function used in input and hidden layer: Relu

Activation function used in output layer: Sigmoid

dropout rate in input layer: 0.1

dropout rate in hidden layer: 0.3

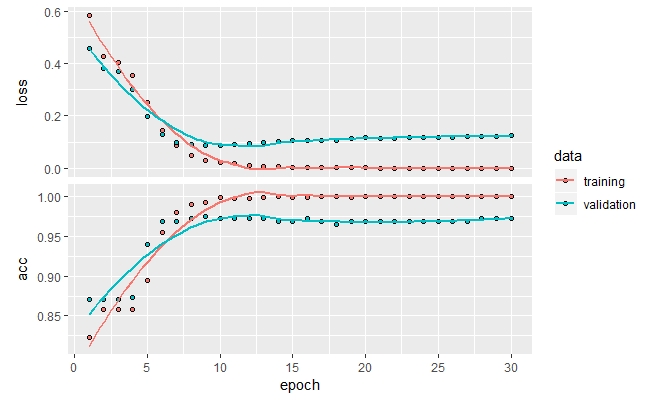
Batch size: 50

Epochs: 30

Validation split: 0.2

Accuracy Obtained: **96.01**

Plots:



**RUN 3:**

Activation function used in input and hidden layer: Tanh

Activation function used in output layer: Sigmoid

dropout rate in input layer: 0.3

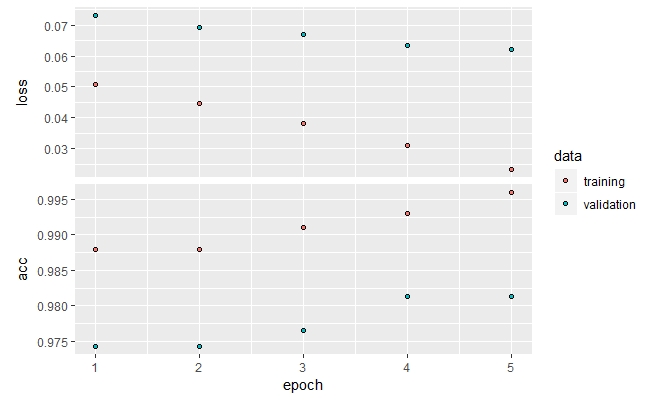
dropout rate in hidden layer: 0.4

Batch size: 100

Epochs: 5

Validation split: 0.3

Accuracy Obtained: **88.04**

Plots:

**RUN 4:**

Activation function used in input and hidden layer: Tanh

Activation function used in output layer: Sigmoid

dropout rate in input layer: 0.1

dropout rate in hidden layer: 0.3

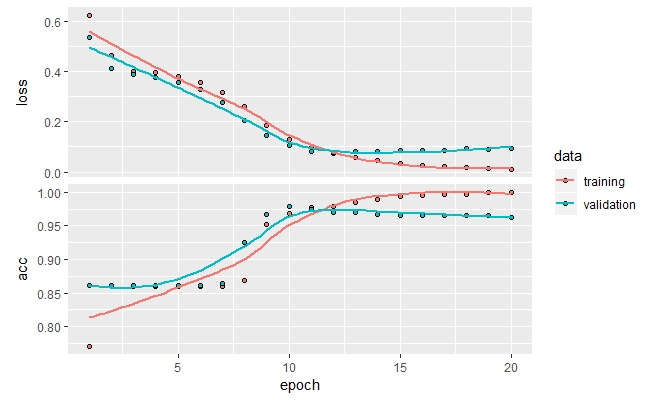
Batch size: 100

Epochs: 20

Validation split: 0.3

Accuracy Obtained: **97.24**

Plots:



**RUN 5:**

Activation function used in input and hidden layer: Relu

Activation function used in output layer: Sigmoid

dropout rate in input layer: 0 (Removed dropout)

dropout rate in hidden layer: 0 (Removed dropout)

hidden units: 100

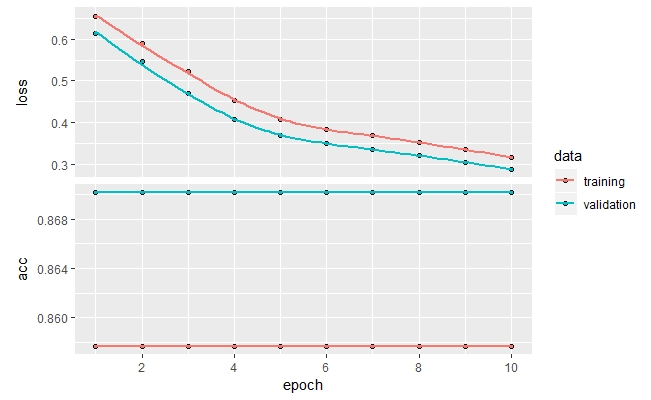
Batch size: 100

Epochs: 10

Validation split: 0.2

Accuracy Obtained: **88.04**

Plots:



**The results from running on the Google Cloud are included in the submission ZIP file. A html file is generated for each run which contains detailed information of the model parameters used, and respective metrics and plots.**