MANIPAL UNIVERSITY
JAIPUR

MUJ/2021/CSE/2021

18 June

Jaipur

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# CERTIFICATE

This is to certify that the project titled **Rumor detection in twitter** is a record of the bona fide work done by **Harshit Kumar (189302122)** submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology(B.Tech) in **Computer Science and Engineering of Manipal University Jaipur, during the academic year 2020- 21.**

**Dr. Sandeep Chaurasia**

*Project guide, Dept. of Computer Science and Engineering*

*Manipal University Jaipur*

A Minor Project Report on

# RUMOUR DETECTION IN TWITTER

*carried out as part of the course CS1634 Submitted by*


**Harshit Kumar**

**(Reg: 189302122)**

**VI-CSE**

**Rakshith Alaham Gangeya**

**(Reg: 189302117)**

**VI-CSE**


in partial fulfilment for the Award of the Degree

of

**BACHELOR OF TECHNOLOGY**

In Computers Science and Engineering

2018-2022


MANIPAL UNIVERSITY
JAIPUR

Under the guidance of

## Dr. Sandeep Chaurasia


**Department of Computer Science and Engineering**

**School of Computing and Information Technology**

**Manipal University Jaipur. April 2021**

# ABSTRACT

Twitter is one of the most popular microblogging platforms. It serves as one of the foremost go-to media for research in natural language processing (NLP), where practitioners rely on deriving various sets of features leveraging content, network structure, and memes of users within these networks. However, the unprecedented existence of such massive data acts as a double-edged sword, one can easily get unreliable information from such sources, and it is a challenge to control the spread of false information either maliciously or even inadvertently. With the growth of social media such as Twitter in the community, it became easier for people to get data or spread rumors. But developing an automatic moderation system to check the credibility of the data is a challenging work, an alternate approach is with the help of users or third-party organization to flag those tweets which are suspected to be rumors, but this method is highly inefficient and redundant, hence making our social media an unreliable source for Information. As a part of this work, we aim to develop an algorithm to segregate such tweets and flag them as rumors. We used three publicly available twitter datasets to find such discriminative features which can classify a tweet as a rumor or non-rumor. Using two recurrent (bi-directional LSTM) neural models and two artificial neural models, the results in the work provide us evident information to correlate important features for classifying the tweets. Along with the result, we have also proposed methods for attributes extraction from user and tweets and pre-processing techniques to retain the key information.

# Contents

# LIST OF FIGURES LIST OF TABLES

# 1. INTRODUCTION

In layman's terms rumor is a statement whose veracity is not confirmed quickly or at all, but it is propagating among a bunch of people. Rumors give birth to the spread of misinformation (false information) or disinformation (deliberate false information) in public. Rumors have been a part of society from historical times. With the birth of new-age technologies and the development of social media sites, the grounds for connectivity have rapidly increased. Among these sites, Twitter has emerged as a leading source of information sharing, where, on average 500 million tweets are shared per day and each user, by design, can tweet up to 140 characters long messages or share files like pictures or videos. Twitter supports a follower and following design in which a user can subscribe to some other user's tweets to receive information posted by the latter. Such a system provides a fluent flow of data. These tweets sometimes contain resourceful information like public announcements by governing organizations, sharing of events, personal experiences, beliefs, thoughts, and many more. Camouflaging among this data, rumors are widely spreading and destroying the credibility of this information. This problem is elevated due to crowdsourcing where a rumor can be backed by many users due to a lack of knowledge regarding the matter and their beliefs. To tackle this issue a lot of work has been done in the research community to find a prominent solution. With this work, we try to understand the root cause of how a rumor is generated on Twitter, its varying nature, way of propagation, the targeted audience, and attributes through which it is easier to determine whether an online post is a rumor or not. Note: the work doesn't include finding the veracity of the rumor tweet, its main purpose is rumor classification only.

## 1.1. MOTIVATION

Upon exhaustive research, we have found that work in the research field basically include developing various methods to flag the post on early stage as rumor and understand their propagation, or mainly finding the veracity of these rumor tweets, i.e., either they are true, false, or un-verified.

With our work, we aim to obtain all such deterministic attributes/features of tweets through which we can determine the credibility of the posts.

# 2. LITERATURE REVIEW

## 2.1. PAPER 1

Psychologists studied the phenomenon of rumors from various angles. First studies were carried out in1902 by German psychologist and philosopher, William Stern, and later in 1947 by his student Gordon Allport, who studied how stories get affected in their lifecycle. In 1994, Robert Knapp published "A Psychology of Rumors", which comprised of a collection of more than a thousand rumors propagated during World War II.

## 2.2. PAPER 2

From an NLP perspective, researchers have studied numerous aspects of credibility of online information. For example, Ghaoui detects rumors within specialized domains, such as trustworthiness, credibility assessment and information verification in online communication.

## 2.3. RELATED WORKS

- Rosa Sicilia, Stella Lo Giudice, Yulong Pei, Mykola Pechenizkiy, Paolo Soda, "Twitter Rumour Detection in the Health Domain" in Expert Systems with Applications (2018).
- The paper explores the selection of different features using different classification methods for rumour detection which is beneficial in selecting effective combination of classifiers and features. It aims at analyzing which features are the most informative and whether the newly introduced ones are meaningful or not for the classification purpose.
- Dr. Dinesh B. Vaghela1, Divya M. Patel2, "Rumor Detection with Twitter and News Channel Data Using Sentiment Analysis and Classification" in International Journal of Advance Engineering and Research Development Volume 5, Issue 02, February -2018.

- In this paper, detection approach is based on the sentiment classification. The paper has results of comparison between different supervised learning techniques for detection of rumours. The limitation is that only one feature, sentiment polarity, is used to detect a rumour. The accuracy of their model ranges from 60-70% using different algorithms.

## 2.4. EARLY WORK

In layman's terms rumor is a statement whose veracity is not confirmed quickly or at all, but it is propagating among a bunch of people. Rumors give birth to the spread of misinformation (false information) or disinformation (deliberate false information) in public. Rumors have been a part of society from historical times. With the birth of new-age technologies and the development of social media sites, the grounds for connectivity have rapidly increased. Among these sites, Twitter has emerged as a leading source of information sharing, where, on average 500 million tweets are shared per day and each user, by design, can tweet up to 140 characters long messages or share files like pictures or videos. Twitter supports a follower and following design in which a user can subscribe to some other user's tweets to receive information posted by the latter. Such a system provides a fluent flow of data. These tweets sometimes contain resourceful information like public announcements by governing organizations, sharing of events, personal experiences, beliefs, thoughts, and many more. Camouflaging among this data, rumors are widely spreading and destroying the credibility of this information. This problem is elevated due to crowdsourcing where a rumor can be backed by many users due to a lack of knowledge regarding the matter and their beliefs. To tackle this issue a lot of work has been done in the research community to find a prominent solution. With this work, we try to understand the root cause of how a rumor is generated on Twitter, its varying nature, way of propagation, the targeted audience, and attributes through which it is easier to determine whether an online post is a rumor or not. Note: the research paper doesn't include finding the veracity of the rumor tweet, its main purpose is rumor classification only.

## 2.5. PROBLEM STATEMENT

Finding, extracting, and engineering important features from a tweet post to develop a supervised classifier able to flag whether the given tweet is rumor or not.

## 2.6. RESEARCH OBJECTIVES

- Understanding and optimizing feature selection from all possible parameters available for extraction from the twitter API.
- The Work also include finding the correlation between derived attributes.
- Training different models on those features and combining them using ensemble methods to get maximum accuracy.

# 3. METHODOLOGY AND FRAMEWORK

## 3.1. SYSTEM ARCHITECTURE

### 3.1.1. Platform

- Google Colab

### 3.1.2. Software

- Windows 8/8.1/10
- Language: Python 3.8.7
- IDE: Colab Notebook
- Modules Used:
- Numpy 1.19.5
- Pandas 1.1.5
- Matplotlib 3.2.2
- Sklearn 0.0
- Tensorflow 2.4.1
- Keras 0.4.1
- Nltk 3.2.5
- Spacy 2.2.4
- Genism 3.6.0
- Selenium 3.141.0

### 3.1.3. Hardware

- Nvidia K80 / T4 GPU model
- 12 GB GPU Memory
- 0.82GHz GPU Memory Clock
- 4.1 TFLOPS performance

## 3.2. ALGORITHMS AND TECHNIQUES USED

### 3.2.1. Logistic Regression

- Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

### 3.2.2. Random Forest

- Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees. Random decision forests correct for decision trees habit of overfitting to their training set. decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

### 3.2.3. LSTM

- Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video).

### 3.2.4. SVM

- In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that evaluate data for classification and regression analysis.

### 3.2.5. XGBoost

- XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

### 3.2.6. Bernoulli Naive Bayes Model

- Bernoulli Naive Bayes is a variant of Naive Bayes. ... Naive Bayes classifier is a probabilistic classifier which means that given an input, it predicts the probability of the input being classified for all the classes. It is also called conditional probability.

## 3.3. DETAILED DESIGN METHODOLOGIES

### 3.3.1. Getting Data

- Retrieved PHEME dataset, Twitter 15-16 dataset and combining them.
- Extracting tweet ids from the dataset.
- Collecting tweets in json format from the twitter using twitter API and tweepy module.
- Reading JSON file of each individual tweets and collecting important key, value attributes in user and tweet csv files.
- Total dataset contains 5900 tweet and user data.
- Overview of tweet data csv:
    - TweetId
    - Text
    - UserId
    - Like count
    - Retweet Count
    - Number of hashtags
    - Number of URLs
    - Number of user mentions
    - Post creation datetime stamp
    - Post age
    - Text length
    - Label

- Overview of user data csv:
    - UserId
    - Account creation datetime stamp
    - Account age
    - Follower's count
    - Verified
    - Statuses count
    - Favorite's count
    - Listed Count
    - Screen name
    - Screen name length

### 3.3.2. Cleaning and preprocessing data

- Converting json files to csv files
- Applying NLP methods on source tweet to remove links, hashtags etc.
- Converted datasets into homogenous formats.
- Applied Veracity factor given by twitter on target values available in the dataset.

### 3.3.3. Feature correlation and selection

- Applying Anova, Kendall and chi-squared analysis on various parameters for feature selection.
- Applying point-biserial correlation and finding relationship between continuous variables and targeted binary label.
- Analyzing collected features for developing various approaches to solve the problem.

### 3.3.4. Various approaches based on collected features.

4. Natural Language Processing on source tweet and LSTM to train a rumor classifier.
5. Natural Language Processing on source tweet and LSTM to train a fake tweet classifier.
6. Model development using Tweet and user meta data.

7. Model development using tweet replies and reactions and other related meta data.

### 3.3.5. Working

In this section, the overall working is divided into 5 parts. The parts are divided based on the hypotheses. Each part will contain details regarding the database used, pre-processing, selected features, the model used.

*A. TWEET TEXT ANALYSIS MODEL*

*1) DATABASE*

In this hypothesis, the combination of PHEME, Twitter 15 and Twitter 16 is used.

*2) PRE_PROCESSING*

Initially, we used the tweet-pre-processor [1] library of python to remove links, emojis, numbers. Next, we used the contractions [2] library of Python to replace the contracted words with their expanded forms. Next, we pre-compiled a list of abbreviation which are commonly used in tweets and then we replaced the abbreviated words in Tweet text with their true forms. Next, we used the wordsegment [3] library of Python to remove mentions, removing the hashtags and splitting the text in the hashtags. Next, we removed the stop words using a pre-defined list. Next, we used the Tokenizer [4] library of Python to fit on texts and then convert them to sequences. We further padded the size of the sequences to a size that is equal to the sum of the mean of cleaned tweets length and standard deviation of the length of cleaned tweets. We have used fast Text word embeddings to create the embedding matrix which is further passed into the neural network.

*3)TRAINING AND MODEL ARCHITECTURE*

We have used a sequential TensorFlow model. The first layer in the model is an Embedding layer which takes the embedding matrix, embeddings dimension, input length as inputs. Next, we have used a dropout layer with a value of 0.5. Next, we have used 4 layers of Bidirectional LSTM with 64 nodes in each with return sequences turned on. We further have 1 layer of Bidirectional LSTM with 32 nodes.

Then, we have a dense layer with 32 nodes and a relu activation function. Next, we have a layer of Batch Normalization which is processed with a dropout layer with a value of 0.5. Further, we have the final layer which is a dense layer with 1 node with the activation function of the sigmoid. This model is compiled with Adam optimizer and the loss used is binary cross-entropy. The accuracy metrics used is accuracy. This model summary is as follows.

Tabel 1

| Layer (Type) | Output Shape | Param# |
|---|---|---|
| embedding (Embedding) | (None,93,300) | 2724600 |
| dropout (Dropout) | (None,93,300) | 0 |
| bidirectional (Bidirectional LSTM) | (None,93,128) | 186880 |
| Bidirectional_1(Bidirectional LSTM) | (None,93,128) | 98816 |
| bidirectional_2(Bidirectional LSTM) | (None,93,128) | 98816 |
| bidirectional_3(Bidirectional LSTM) | (None,93,128) | 98816 |
| bidirectional_4(Bidirectional LSTM) | (None,64) | 41216 |
| dense (Dense) | (None,32) | 2080 |
| batch normalization(Batch Normalization) | (None,32) | 128 |
| dropout_1(Dropout) | (None,32) | 0 |
| dense_1(Dense) | (None,1) | 33 |

| |
|---|
| Total Params: 6,317,985 |
| Trainable params: 526,721 |
| Non-trainable params: 5,791,264 |

This data was split into test and train with a test size of 0.2, random state of 42 and stratified for the target labels.

The model was trained with the X_train padded sequences and y_train and with validation data ( X_test padded sequences, y_test). The batch size taken is 128 and trained for 150 epochs with an early stopping monitoring for minimum validation loss with patience of 5.

*B. FAKE TWEET CLASSIFICATION*

*1) DATABASE*

In this hypothesis, GossipCon dataset is used.

*2)PRE_PROCESSING*

The pre-processing is the same as the Tweet text analysis model.

*3)TRAINING AND MODEL ARCHITECTURE*

The model used here is the same as the tweet text analysis model. Only the number of word embeddings is changed due to different sizes of databases. This model summary is as follows.

Table 2

| Layer (Type) | Output Shape | Param# |
|---|---|---|
| embedding (Embedding) | (None,84,300) | 5791200 |
| dropout (Dropout) | (None,84,300) | 0 |
| bidirectional (Bidirectional LSTM) | (None,84,128) | 186880 |
| Bidirectional_1(Bidirectional LSTM) | (None,84,128) | 98816 |

| | | |
|---|---|---|
| bidirectional_2(Bidirectional LSTM) | (None,84,128) | 98816 |
| bidirectional_3(Bidirectional LSTM) | (None,84,128) | 98816 |
| bidirectional_4(Bidirectional LSTM) | (None,64) | 41216 |
| dense (Dense) | (None,32) | 2080 |
| batch normalization (Batch Normalization) | (None,32) | 128 |
| dropout_1(Dropout) | (None,32) | 0 |
| dense_1(Dense) | (None,1) | 33 |
| Total Params: 6,317,985 | | |
| Trainable params: 526,721 | | |
| Non-trainable params: 5,791,264 | | |

This data was split into test and train with a test size of 0.2, random state of 42 and stratified for the target labels.

The model was trained with the X_train padded sequences and y_train and with validation data ( X_test padded sequences, y_test). The batch size taken is 256 and trained for 150 epochs with an early stopping monitoring for minimum validation loss with patience of 5.

### *C. USER AND TWEET META-DATA MODEL*

#### *1) DATABASE*

In this hypothesis, the combination of PHEME, Twitter 15 and Twitter 16 is used

#### *2)PRE_PROCESSING*

The data was analysed and the features with skewed data were log-transformed to get a normal distribution. Next, the pre-processing of text is done like the tweet text analysis model. Further, the text was also lemmatized. We used TextBlob [5] to find the sentiment polarity of the text. Next, we calculated the text length and tweet-age features. Next, we calculated the Posted in feature which is the time gap in which the user created the profile and posted the tweet. The final feature list is.

Table 3

| Numerical Features | Categorical Features |
|---|---|
| Follower's count | Is_reply |
| Retweet count | Verified |
| Favourite count | Is_quote_status |
| No: of Symbols | Profile_image_url |
| No: of User mentions | profile_background_image_url |
| No: of Hashtags | Default profile image |
| No: of URL's | Default profile |
| Polarity | Profile_use_background_image |
| Text length | Has_location |
| Post age | Has_url |
| Statuses count | |
| Friends count | |
| Favourites count of user | |
| Listed count | |
| Account age | |
| Screen name length | |
| The time gap between user-created time and tweeted time(Posted_in_time) | |

In total, we have 27 features. We performed feature selection on these features and arrived at the final features.

The final list of selected features after feature selection is.

Table 4

| Numerical Features | Categorical Features |
|---|---|
| Text Length | Profile_use_background_image |
| No: of URL's | Profile_background_image_url |
| Post age | Default_profile_image |
| Statuses count | Default_profile |
| Listed count | Verified |
| No: of symbols | |
| Posted_in | |
| No: of user mentions | |
| Polarity | |
| Favourites count of user | |
| Favourite count of a tweet | |
| Screen name length | |
| No: of hashtags | |

*3)TRAINING AND MODEL ARCHITECTURE*

First, we applied a Standard scaler to the selected features. We have used a sequential TensorFlow model. The first layer is a dense layer with 64 nodes, relu activation function and L2 regularization with a value of 0.01. Next, we have a dropout layer with a value of 0.3. Further, we have a dense layer with 64 nodes, relu activation function and L2 regularization with a value of 0.01. Next, we have a dropout layer with a value of 0.3. The final layer is a dense layer with a sigmoid activation function and glorot_uniform weights initializer. The model is compiled with Adam optimizer,

having binary cross-entropy loss and metrics as accuracy. This model summary is as follows.

Table 5

| Layer(type) | Output Shape | Param # |
|---|---|---|
| dense_1(Dense) | (None,64) | 1216 |
| dropout_1(Dropout) | (None,64) | 0 |
| dense_2(Dense) | (None,64) | 4160 |
| dropout_2(Dropout) | (None,64) | 0 |
| dense_3(Dense) | (None,1) | 65 |
| Total params: 5,441 | | |
| Trainable params: 5,441 | | |
| Non-trainable params: 0 | | |

This data was split into test and train with a test size of 0.2, random state of 42 and stratified for the target labels.

The model was trained with scaled X_train values and y_train. The batch size taken was 32 and validation data was (X_test_scaled,y_test).

*D. SENTIMENTAL ANALYSIS ON REACTION OF TWEETS MODEL*

*1) DATABASE*

In this hypothesis, the PHEME dataset is used.

*2)PRE_PROCESSING*

The text comments on the tweets are cleaned and pre-processed like the tweet text analysis model and further, it is lemmatized. Text Blob Python library was used to find the polarity of the comments on the tweets. We multiplied these polarities with favourite count, retweet count, sum of favourite count and retweet count to get 3 weighted averages of polarities. Retweet count, Favourite count, no: of hashtags in the text of comments are also taken in the total features list. The total list of features is.

Table 6

| Retweet Count |
|---|

| Favourite count |
|---|
| Sentiments |
| No: of hashtags |
| Favourite weighted polarity |
| Retweet weighted polarity |
| Favourite retweet weighted polarity |

After performing feature selection, it was observed that Favourite weighted polarity, retweet weighted polarity and favourite retweet weighted polarity had a similar negative correlation to the label.

So, we only took the main polarity(sentiments) feature during feature selection.

The final list of selected features is.

Table 7

| Favourite count |
|---|
| Sentiments (Polarity) |
| No: of hashtags |

This data is scaled using Standard Scaler after splitting it into test and train.

*3)TRAINING AND MODEL ARCHITECTURE*

We have used a sequential TensorFlow model. The first layer is a dense layer with 32 nodes and a relu activation function. Second, we have a dropout layer with a value of 0.3. Third, we have a dense layer with 16 nodes and a relu activation function. Fourth, we have a dropout layer with a value of 0.2. Fifth, we have a dense layer with 16 nodes and a relu activation function. Sixth, we have the final layer which is a dense layer with a sigmoid activation function.

This model is compiled with Adam optimizer and binary cross-entropy is taken as loss function. The metric used is accuracy. The model summary is as follows.

Table 7

| Layer(type) | Output Shape | Param # |
|---|---|---|
| dense_1(Dense) | (4598,32) | 256 |

| | | |
|---|---|---|
| drouput_1(Dropout) | (4598,32) | 0 |
| dense_2(Dense) | (4598,16) | 528 |
| dropout_2(Dropout) | (4598,16) | 0 |
| dense_3(Dense) | (4598,16) | 272 |
| dense_4(Dense) | (4598,1) | 17 |
| Total params: 1,073 | | |
| Trainable params: 1,073 | | |
| Non-trainable params: 0 | | |

This data was split into test and train with a test size of 0.2, random state of 42 and stratified for the target labels.

The model was trained with scaled X_train values and y_train. The batch size taken was 32 and validation data was (X_test_scaled,y_test).

*E. ENSEMBLE MODEL COMBINING THE ABOVE MODELS*

*1) DATABASE*

In this hypothesis, the combination of PHEME, Twitter 15 and Twitter 16 is used

*2)TRAINING AND MODEL ARCHITECTURE*

Due to the lack of data of reactions on tweets, we have ensembled the other 3 models on the complete data and ensembled all 4 models only for PHEME data.

We have extracted predict probabilities of each model and treated them as separate features to train a model. We have used a sequential TensorFlow model. The first layer is a dense layer with 64 nodes and a relu activation function. The second is a dropout layer with a value of 0.3. Third, is a Dense layer with 64 nodes and a relu activation function. Fourth, is a dropout layer with a value of 0.3. The fifth and final layer is a dense layer with 1 node, sigmoid activation function and glorot_uniform weight initializer.

The model is compiled using Adam optimizer and binary cross-entropy as loss function. The metrics used are accuracy. The summary of the model is as follows.

Table 8

| Layer (Type) | Output Shape | Param # |
|---|---|---|

| | | |
|---|---|---|
| dense_1(Dense) | (None,64) | 256 |
| droupout_1(Dropout) | (None,64) | 0 |
| dense_2(Dense) | (None,64) | 4160 |
| dropout_2(Dropout) | (None,64) | 0 |
| dense_8(Dense) | (None,1) | 65 |
| Total params: 4,481 | | |
| Trainable params: 4,481 | | |
| Non- trainable params: 0 | | |

# 4. WORK DONE

## 4.1. Details as required.

- Collected Data from various sources.
- Cleaned and pre-processed data
- Feature selection of parameters
- Completed models of 4 different approaches mentioned above.

## 4.2. Results

### 4.2.1 LSTM on source text tweet after applying NLP methods.

- Train accuracy:88.7%
- Test accuracy: 80.1%
- Train loss: 0.274
- Test loss: 0.449

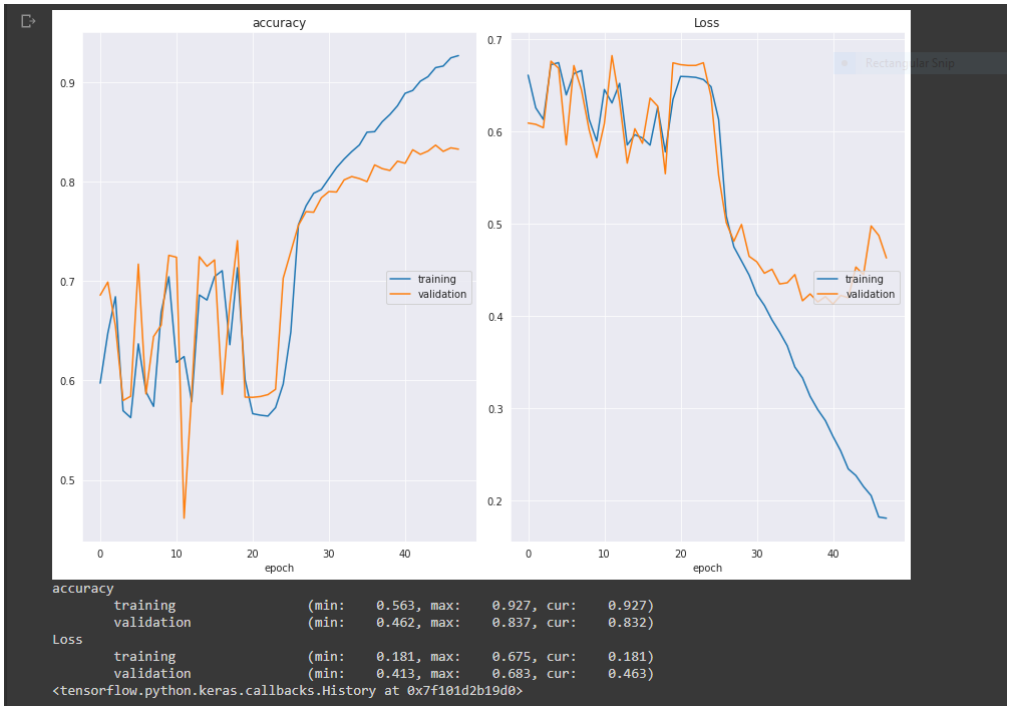


Figure 1

### 4.2.2 Model on tweet and user meta data

- Train accuracy:78.1%
- Test accuracy: 71.9%
- Train loss: 0.469
- Test loss: 0.550

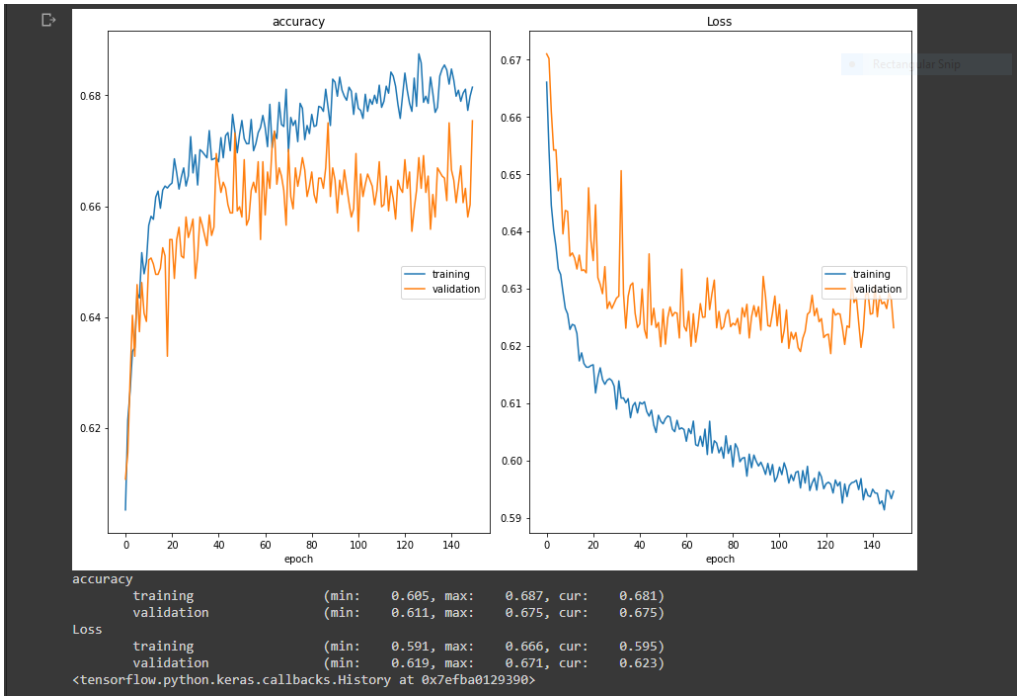

Figure 2

### 4.2.3 Fake Tweet classification Model.

- Train accuracy:97.4%
- Test accuracy: 97.3%
- Train loss: 0.074
- Test loss: 0.084

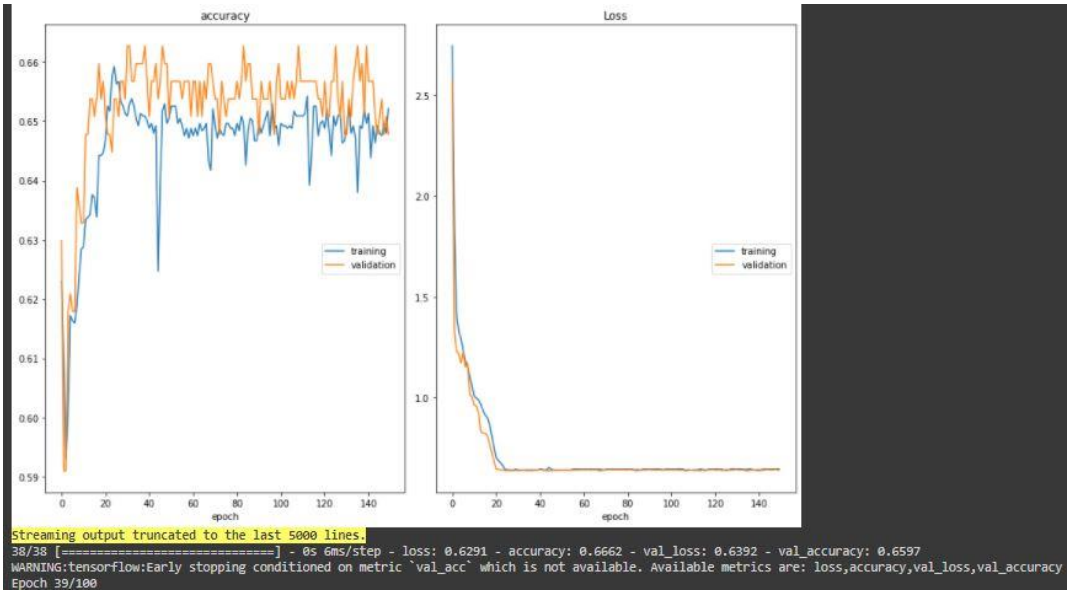


Figure 3

### 4.2.4 Model on sentimental analyser on tweets comments

- Train accuracy:63.5%
- Test accuracy: 63.3%
- Train loss: 0.644

- Test loss: 0.647

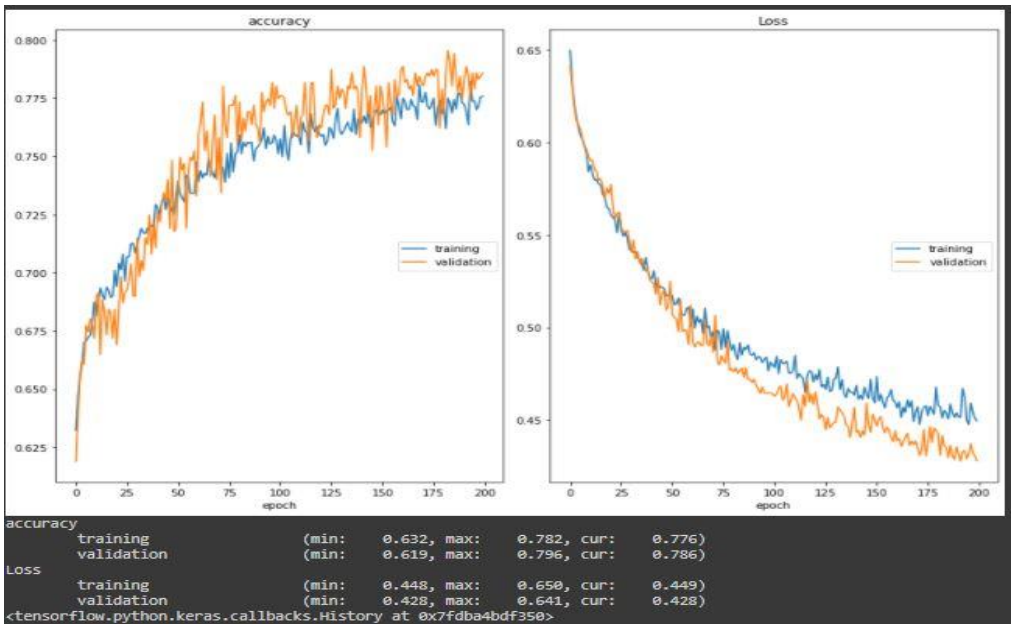


Figure 4

## 4.3. Individual contribution

### Harshit Kumar

- Data gathering and Data cleaning.
- Analysing features and developing various approaches to solve the problem.
- Models on Reactions data using sentimental analyzer.
- Models on user's credibility using user's past tweet data.

### Rakshith Alaham Gangeya

- Feature Selection of parameters
- Analysing features and developing various approaches to solve the problem.
- LSTM model on source text data after applying NLP methods.
- Models on tweet, user raw meta data

# 5. CONCLUSION AND FUTURE WORK

1. Developing and proving all the suggested hypothesis
2. Ensemble all models to get a final model.
3. Fine tuning of model's Hyperparameters
4. Final Model Weights estimation to conclude Feature's importance.

# References


[1] Keen, P., Honnibal, M., & Yankovsky, R. (n.d.). *TextBlob: Simplified Text Processing*. Retrieved from https://textblob.readthedocs.io: https://textblob.readthedocs.io/en/dev/

[2] Google. (n.d.). *Tensorlfow Core v2.5.0*. Retrieved from tensorflow.org: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer

[3] Jenks, G. (n.d.). *Python Word Segmentation*. Retrieved from grantjenkis.com: http://www.grantjenks.com/docs/wordsegment/

[4] Özcan, S. (n.d.). *Tweet-preprocessor*. Retrieved from pypi.org: https://pypi.org/project/tweet-preprocessor/

[5] van, P. (n.d.). *contractions.0.0.52*. Retrieved from pypi.org: https://pypi.org/project/contractions/