# Covid - 19 Misinformation Detection in Social Media

Lekha Shanthini R - 2018103558

Shankar Kumar S - 2018103063

## INTRODUCTION

Every second, a massive amount of data is generated from microblogs, content sharing via social media sites, and social networking. Twitter is a popular microblog where people express their views on current events. The act of collecting user-generated information from social media platforms and obtaining meaningful inferences is known as data mining in social media. Social media data mining aims to extract valuable data from consumers, identify patterns and trends, and form business conclusions. It is an effective tool in marketing and business strategies and will help better understand how people react to various topics.

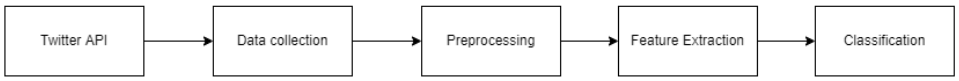
Fake news is a type of propaganda where misinformation is intentionally spread through news outlets and/or social media outlets. The gradual spread of fake news and misleading information during these hard times can have dire consequences, causing widespread panic and exacerbating the apparent threat of a pandemic that we cannot ignore.

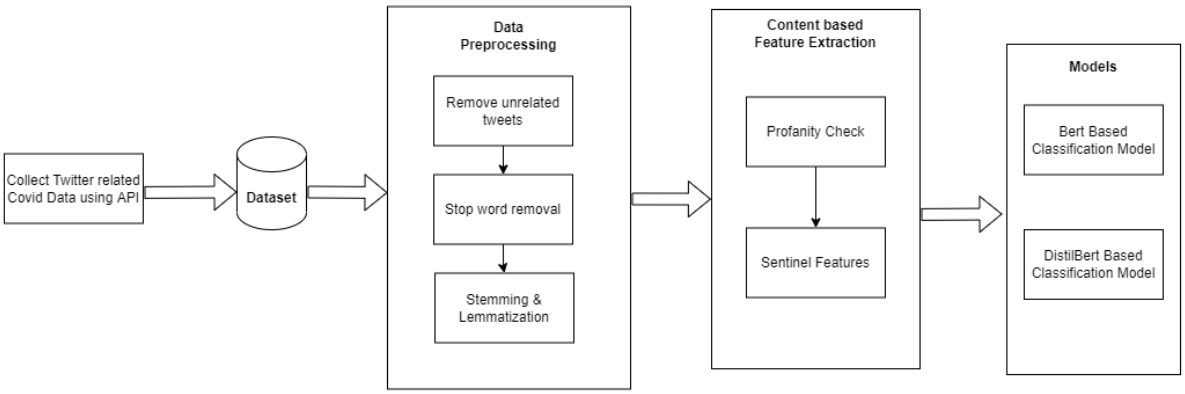
To deal with these challenges, in this project, we propose a robust approach to classify misinformation about Covid-19 using social media data.

## OBJECTIVES

- To check the truthfulness of major claims regarding Covid-19 in a social media post to decide the news veracity.
- To build a robust classification model to identify fake news regarding Covid-19.
- To extract sentinel textual features and abusive language scores.
- To implement an effective approach for combining the extracted features with the text embeddings.
- To compare the performance of various models using our approach.

## BLOCK DIAGRAM




## NOVELTY

The existing approach in the base paper uses a BERT based model to predict Covid-19 related fake news. We propose a method to incorporate BERT based text embeddings along with additional features obtained from the Feature Extraction module. This will enable us to identify and extract linguistic features like abusive language and sentinel features, from a given unstructured raw tweet. We hypothesize that giving the extracted supplementary features to the classification model will boost the accuracy of detecting the fake information.

## DATASET DESCRIPTION

Constraint.AI - 2021 is a shared task on hostile post detection which contains data collected from sources including various social-media platforms such as Twitter, Facebook, Instagram. Each record comprises a tweet related to Covid-19 and a label to denote its credibility. There are around 6000 records for training, 2000 for validation and 2000 for testing.

## MODULE DESCRIPTION

❖ **Install packages and dependencies**

The necessary packages and dependencies like NumPy, pandas, TensorFlow, Matplotlib, nltk, etc are installed and imported.

```
!pip install tensorflow-text==2.4.2
!pip install tf-models-official==2.4
!pip install tensorflow-gpu==2.4.2
!pip install delayed
!pip install alt-profanity-check
import numpy as np
import pandas as pd
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_text as text
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import classification_report
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import joblib
```

- ***Feature Extraction***

Extraction of linguistic features from the raw tweets is executed in this module. We write two functions namely:

➔ get_profanity_scores() - Checks for abusive language in the tweet and assigns a score accordingly. Generally, fake news can have hate speech and profane abusive words. If the profanity score is high, the tweet contains offensive words.

➔ get_text_ft() - Extracts all the sentinel features present in the tweet and returns the output as a list. These sentinel features include text length, number of uppercase characters, number of question marks, number of exclamation marks, etc.

Below are the top 5 tweets with the highest profanity scores. As we can observe, these tweets are 'fake' news, thus supporting our hypothesis for novelty.

| | id | tweet | label | profanity_scores | length | num_of_upper_char | num_of_qmarks | num_of_exclmmarks |
|---|---|---|---|---|---|---|---|---|
| 2160 | 2161 | Local Man Sick To Fucking Death Hearing About ... | fake | 0.999747 | 85 | 16 | 0 | 0 |
| 4895 | 4896 | Stoner Strung Out To Fuck Thanks To Lockdown R... | fake | 0.999376 | 81 | 11 | 0 | 0 |
| 8557 | 2138 | Nothing screams "I am sat around doing fuck al... | fake | 0.997964 | 184 | 6 | 0 | 0 |
| 4927 | 4928 | Ellie Goulding says she's amazed there were 20... | fake | 0.993335 | 174 | 13 | 1 | 0 |
| 2677 | 2678 | Coronavirus Cancels Billions of Man-Hours of P... | fake | 0.967980 | 96 | 12 | 0 | 0 |

The next table shows the top tweets with the highest number of sentinel features, which are also labeled as 'fake'.

| | id | tweet | label | profanity_scores | length | num_of_upper_char | num_of_qmarks | num_of_exclmmarks |
|---|---|---|---|---|---|---|---|---|
| 5464 | 5465 | Says Sarah Huckabee Sanders tweeted, ???It???s... | fake | 0.015965 | 338 | 28 | 34 | 0 |
| 3045 | 3046 | ???President Trump just announced that the ???... | fake | 0.008184 | 219 | 13 | 17 | 1 |
| 3291 | 3292 | Quotes Anthony Fauci as writing, ???I reject t... | fake | 0.082221 | 165 | 5 | 17 | 0 |
| 4420 | 4421 | Trey Gowdy said, ???I???m not saying COVID-19 ... | fake | 0.109564 | 149 | 9 | 17 | 0 |
| 5902 | 5903 | ???The CDC has removed the ???Covid-19 Pandemi... | fake | 0.010282 | 141 | 9 | 17 | 0 |

- ***Data Preprocessing***

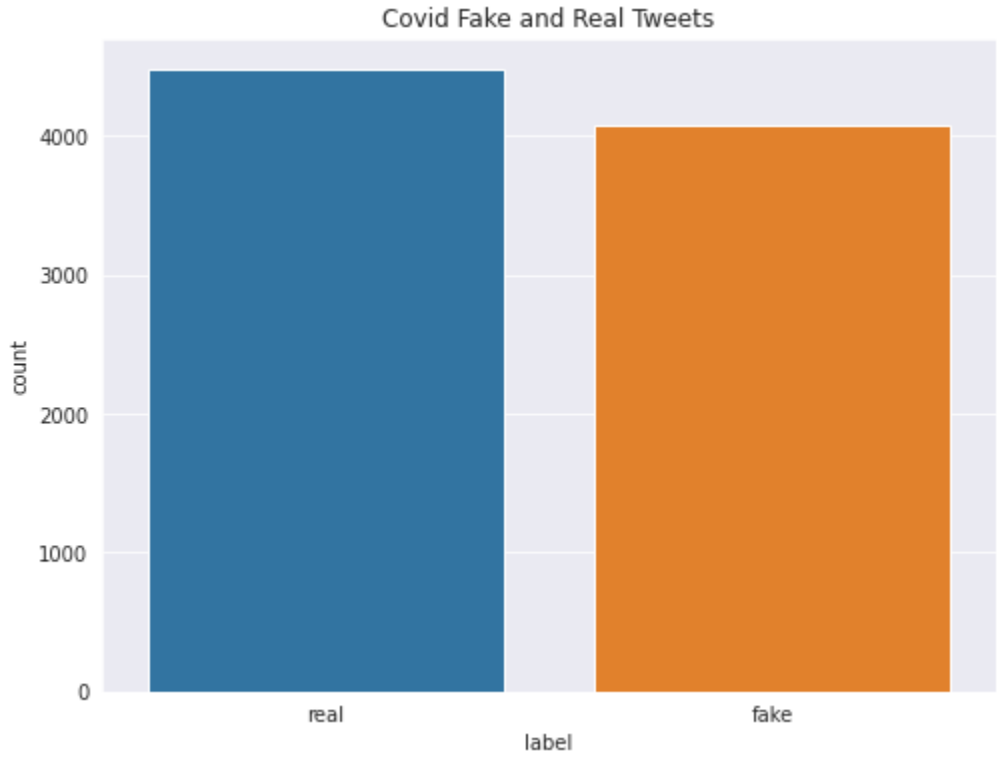
❖ **Reading training data**

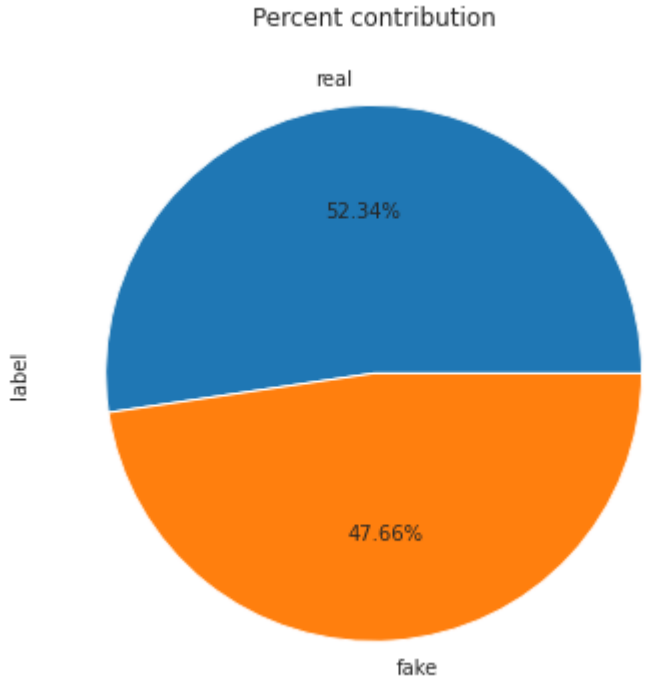
The dataset which is in the form of csv is loaded, and the sample is shown below. It consists of raw tweets extracted from Twitter and labels to denote real or fake.

| | id | tweet | label |
|---|---|---|---|
| 0 | 1 | The CDC currently reports 99031 deaths. In gen... | real |
| 1 | 2 | States reported 1121 deaths a small rise from ... | real |
| 2 | 3 | Politically Correct Woman (Almost) Uses Pandem... | fake |
| 3 | 4 | #IndiaFightsCorona: We have 1524 #COVID testin... | real |
| 4 | 5 | Populous states can generate large case counts... | real |

❖ **Training data insight**

The real and fake tweets are plotted in the form of a bar chart and a pie chart to analyze their corresponding contribution.



## ❖ Data Cleaning

In this module, we remove the unwanted details and inconsistencies in the data. First, all the links present in the tweet are replaced with a single blank space. Then the hashtags, extra whitespace and emojis are removed. Alphanumeric characters are replaced with a blank. Further, all the words are reduced to their stem words using Porter Stemmer, a python NLTK(Natural Language Toolkit) package.

In the next step, the stopwords are filtered out from the raw tweets. Stop words are a set of commonly used words in a language. Examples of stop words in English are “a”, “the”, “is”, “are” and so on. NLTK in python has a list of stopwords stored in 16 different languages. We download and store the stopwords for English and remove the words that match from our tweets. Finally, all the characters are converted to lowercase and produced as output of the preprocessing module.

| | id | tweet | label |
|---|---|---|---|
| 0 | 1 | cdc currently reports deaths general discrepan... | real |
| 1 | 2 | states reported deaths small rise last tuesday... | real |
| 2 | 3 | politically correct woman almost uses pandemic... | fake |
| 3 | 4 | testing laboratories india august tests done | real |
| 4 | 5 | populous states generate large case counts loo... | real |

## ❖ Generating word clouds

Word cloud is a data visualization technique for representing text data in which the size of each word indicates the frequency or importance of the word. A word cloud can highlight significant textual data points. Word clouds are commonly used to analyze data from social networking sites. Here, we separate the dataset based on its label as ‘real’ and ‘fake’ and generate two word clouds.

***Word Cloud for fake news:***


From the above word cloud, it is observed that most fake news is related to the ‘lockdown’, ‘vaccine’, ‘hospital’ as they are highlighted more than others.

***Word Cloud for real news:***


The words ‘number’, ‘total’, ‘new’, ‘average’ are some of the highlighted ones. It is evident from the above word cloud that real news contains more statistical data.

## ❖ Downsampling data

The dataset has an unequal distribution of classes where it has 4480 samples for ‘real’ and 4080 for ‘fake’. Training any model on an unbalanced dataset will lead to bias and impact the correlation between the features. To overcome this issue, we

downsample the dataset where some samples from the majority class(real) are randomly removed. After downsampling, we have 4080 samples for each class.

```
df_balanced = pd.concat([df_fake, df_real_downsampled])
df_balanced['label'].value_counts()
```

```
fake    4080
real    4080
Name: label, dtype: int64
```

- ***Data Normalization***

Following the feature extraction, the extracted textual features contain continuous values. These values are normalized using the MinMax Scaler. The main idea behind normalization/standardization is that variables that are measured at different scales do not contribute equally to the model fitting & model learned function. This might end up creating a bias. Thus, to deal with this potential problem feature-wise normalization such as MinMax Scaling is usually used prior to model fitting.The MinMax Scaler shrinks all the features within the given range, generally 0 to 1. Normalizing all the values of these extracted features we get,

| | id | tweet | label | profanity_scores | length | num_of_upper_char | num_of_qmarks | num_of_exclmmarks |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | The CDC currently reports 99031 deaths. In gen... | real | 0.009266 | 0.019483 | 0.013304 | 0.0 | 0.0 |
| 1 | 2 | States reported 1121 deaths a small rise from ... | real | 0.032141 | 0.012574 | 0.022173 | 0.0 | 0.0 |
| 2 | 3 | Politically Correct Woman (Almost) Uses Pandem... | fake | 0.061461 | 0.012800 | 0.035477 | 0.0 | 0.0 |
| 3 | 4 | #IndiaFightsCorona: We have 1524 #COVID testin... | real | 0.033270 | 0.021069 | 0.070953 | 0.0 | 0.0 |
| 4 | 5 | Populous states can generate large case counts... | real | 0.014837 | 0.024128 | 0.057650 | 0.0 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 8555 | 2136 | Donald Trump wrongly claimed that New Zealand ... | fake | 0.011558 | 0.025261 | 0.039911 | 0.0 | 0.0 |
| 8556 | 2137 | Current understanding is #COVID19 spreads most... | real | 0.102877 | 0.024014 | 0.037694 | 0.0 | 0.0 |
| 8557 | 2138 | Nothing screams "I am sat around doing fuck al... | fake | 0.997864 | 0.018804 | 0.013304 | 0.0 | 0.0 |
| 8558 | 2139 | Birx says COVID-19 outbreak not under control ... | fake | 0.029981 | 0.009515 | 0.019956 | 0.0 | 0.0 |
| 8559 | 2140 | Another 4422 new coronavirus cases have been c... | real | 0.011324 | 0.029225 | 0.031042 | 0.0 | 0.0 |

- ***Model Construction***

We have constructed two models BERT and DistilBERT and passed the output word embeddings to a neural network for the classification task. Later, we have also incorporated the extracted text features to the word embeddings to be passed as input to the neural network. Hence, four different models are executed here.

**A. BERT**

BERT helps us grasp the subtle nuances of language that computers don't quite understand the way humans do. BERT stands for Bidirectional Encoder Representations from Transformers. It is a transformer-based model pre-trained on 2500M words from Wikipedia and 800M words from books by Google. Traditional models were trained on the input based on a specific direction, either from left-to-right or from right-to-left. This led to errors due to loss in some contextual information. On the other hand, BERT is a contextual model, which processes a word based on all of its surroundings. Thus, it is deeply bidirectional.

We make use of these properties of BERT in this project. First, the BERT encoder and preprocessor is loaded. The input is first passed through a BERT preprocessor, which tokenizes the input into a sequence of ids. It is then passed into a BERT encoder. The output is passed as an input to the Neural Network which contains 1 Fully Connected layer. We have added dropout function which prevents the model from overfitting. The sigmoid activation function is applied here, as we are dealing with binary classification problem. The output of this will always be between 0 and 1. The model is compiled with Adam Optimizer to perform optimization and binary cross entropy loss. Binary cross entropy compares each of the predicted probabilities to actual class output which can be either 0 or 1. It then calculates the score that penalizes the probabilities based on the distance from the expected value.

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
text (InputLayer)               [(None,)]            0
__________________________________________________________________________________________________
keras_layer (KerasLayer)        {'input_word_ids': ( 0           text[0][0]
__________________________________________________________________________________________________
keras_layer_1 (KerasLayer)      {'default': (None, 7 109482241   keras_layer[0][0]
                                                                 keras_layer[0][1]
                                                                 keras_layer[0][2]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 768)          0           keras_layer_1[0][13]
__________________________________________________________________________________________________
output (Dense)                  (None, 1)            769         dropout[0][0]
==================================================================================================
Total params: 109,483,010
Trainable params: 769
Non-trainable params: 109,482,241
__________________________________________________________________________________________________
```

**B. BERT with text features**

The input is first passed through a BERT preprocessor which tokenizes the input into a sequence of ids. It is then passed into a BERT encoder, which produces BERT word embeddings. These word embeddings are then concatenated with the textual features extracted from the feature extraction module. The output is passed as an input to the Neural Network which contains 2 Fully Connected layers with 512 and 256 neurons respectively. Dropout functions are added to prevent the model from overfitting. The sigmoid activation function is applied here, as we are dealing with a binary classification problem. Regularization techniques like Early Stopping are also applied to further prevent any possibilities of overfitting. The model is compiled with Adam Optimizer to perform optimization and binary cross entropy loss.

```
Model: "model"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
text (InputLayer)            [(None,)]                 0
_________________________________________________________________________________________________
keras_layer (KerasLayer)     {'input_word_ids': ( 0           text[0][0]
_________________________________________________________________________________________________
keras_layer_1 (KerasLayer)   {'default': (None, 7 109482241   keras_layer[0][0]
                                                              keras_layer[0][1]
                                                              keras_layer[0][2]
_________________________________________________________________________________________________
fullyConnected1 (Dense)      (None, 512)               393728      keras_layer_1[0][13]
_________________________________________________________________________________________________
dropout (Dropout)            (None, 512)               0           fullyConnected1[0][0]
_________________________________________________________________________________________________
fullyConnected2 (Dense)      (None, 256)               131328      dropout[0][0]
_________________________________________________________________________________________________
input_1 (InputLayer)         [(None, 5)]               0
_________________________________________________________________________________________________
output (Dense)               (None, 1)                 257         fullyConnected2[0][0]
=================================================================================================
Total params: 110,007,554
Trainable params: 525,313
Non-trainable params: 109,482,241
_________________________________________________________________________________________________
```

### C. DistilBERT

DistilBERT is a small, fast, affordable, and lightweight BERT's distillation transformer model. It has

- ➔ 40% fewer parameters than Bert-based
- ➔ Works 60% faster

The general workflow of DistilBERT can be divided into:

- ❖ Tokenizing text
- ❖ Defining the model architecture
- ❖ Fine-tuning DistilBERT and training all weights

We write a function bert_encode() to tokenize the input passed. It adds special tokens like [CLS]

and [SEP] at the beginning and end of each sentence respectively. We then add paddings to the sentences that are shorter than the maximum length to make up the length. The output is passed as an input to the Neural Network which contains 1 fully connected layer. We have added a dropout function which prevents the model from overfitting. The sigmoid activation function is applied here, as we are dealing with a binary classification problem. The output of this will always be between 0 and 1. Regularization techniques are also applied to further prevent any possibilities of overfitting. The model is compiled with Adam Optimizer and binary cross entropy loss. Binary cross entropy compares each of the predicted probabilities to actual class output which can be either 0 or 1. It then calculates the score that penalizes the probabilities based on the distance from the expected value.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_word_ids (InputLayer)  [(None, 160)]             0
_________________________________________________________________
tf_distil_bert_model (TFDist TFBaseModelOutput(last_hi 66362880
_________________________________________________________________
tf.__operators__.getitem (Sl (None, 768)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 769
=================================================================
Total params: 66,363,649
Trainable params: 66,363,649
Non-trainable params: 0
_________________________________________________________________
```

**D. DistilBERT with text features**

The input is passed to bert_encode() function. It adds special tokens like [CLS] and [SEP] at the beginning and end of each sentence respectively. We then add paddings to the sentences that are shorter than the maximum length to make up the length. These word embeddings are

then concatenated with the textual features extracted from the feature extraction module. The output is passed as an input to the Neural Network which contains 2 fully connected layers with 512 and 256 neurons respectively. We have added a dropout function that prevents the model from overfitting. The sigmoid activation function is applied here, as we are dealing with a binary classification problem. The output of this will always be between 0 and 1. Regularization techniques are also applied to further prevent any possibilities of overfitting. The model is compiled with Adam Optimizer and binary cross-entropy loss.

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_word_ids (InputLayer)     [(None, 160)]        0
__________________________________________________________________________________________________
tf_distil_bert_model (TFDistilB TFBaseModelOutput(la 66362880    input_word_ids[0][0]
__________________________________________________________________________________________________
tf.__operators__.getitem (Slici (None, 768)          0           tf_distil_bert_model[0][0]
__________________________________________________________________________________________________
input_1 (InputLayer)            [(None, 5)]          0
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 773)          0           tf.__operators__.getitem[0][0]
                                                                 input_1[0][0]
__________________________________________________________________________________________________
dropout1 (Dropout)              (None, 773)          0           concatenate[0][0]
__________________________________________________________________________________________________
fullyConnected1 (Dense)         (None, 512)          396288      dropout1[0][0]
__________________________________________________________________________________________________
dropout2 (Dropout)              (None, 512)          0           fullyConnected1[0][0]
__________________________________________________________________________________________________
fullyConnected2 (Dense)         (None, 256)          131328      dropout2[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 1)            257         fullyConnected2[0][0]
==================================================================================================
Total params: 66,890,753
Trainable params: 66,890,753
Non-trainable params: 0
__________________________________________________________________________________________________
```

- ***Model Evaluation and Metrics***

The models after training were evaluated using the test dataset which had 2140 samples. Accuracy was chosen as the evaluation metric to compare the performance of the different models.

Accuracy is the ratio of the total number of correct predictions and the total number of predictions.

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + False\ Positive + True\ Negative + False\ Negative}$$

Precision is the ratio between the True Positives and all the Positives.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

The recall is the measure of our model correctly identifying True Positives.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

## EXPERIMENTS AND RESULTS

For training the models, 6420 training samples and 2140 test samples were used. 25% of the training samples were used for validation.

During training, for models which do not incorporate the extracted features only the text is fed as input. For other models, both the text and extracted features are fed as input. The BERT based model obtained an accuracy of 84.11% and the DistilBert model obtained an accuracy of 92.75%.

On concatenating the extracted textual features, the BERT based model achieved an accuracy of 86.54% and the DistillBert model achieved accuracy of 94.03%.

The following table shows the comparison of the accuracies among different models that have been implemented,

| S.No | MODEL | ACCURACY |
|---|---|---|
| 1 | BERT | 84.11 |
| 2 | BERT with text features | 86.54 |
| 3 | DistilBERT | 92.75 |
| 4 | DistilBERT with text features | 94.03 |

## CONCLUSION

A deep learning model capable of classifying Covid-19 misinformation was developed. Feature extraction techniques were applied to generate profanity scores and sentinel features. Different types of architectures were tested and the proposed DistilBert with text features achieved the highest accuracy of 92%. Moreover, our proposed approach of combining textual features along with word embedding proved to enhance the classification accuracy on both the Bert Based and the DistilBert based model.