ANALYSIS OF DISASTER TWEETS' AUTHENTICITY

PROJECT MIDTERM REPORT - GROUP 39

Srilekha Gudipati, Sasank Marabattula, Sukruthi Modem, Srihitha Reddy Kaalam
Department of Computer Science, North Carolina State University, Raleigh, NC 27695
sngudipa, smaraba, smodem, skaalam

ABSTRACT

The influence of social media on users has made it uncertain to identify if the content provided is authentic or not. A major challenge in fake news detection or authenticity verification is the unavailability or the shortage of labelled data for training the detection models. Our aim is to solve a specialized subset of this problem i.e, to classify whether a given tweet is about a real disaster or not. Committees of interest can use this method to identify and help disaster-stricken communities as soon as possible. In this paper, we discuss the Data exploration techniques used and the experiment setup. Various embeddings and classification models are explored to design a proper model. Evaluation metrics like accuracy, F1score, recall and cross validation methods are used to identify the best model for this problem statement.

1 Introduction

The impact of social media on society has been growing. Over the last two decades, the amount of social media applications and its users have been exponentially growing and so is the impact of social media influence over its users. Fake news creation has become a norm across several platforms. Given the influence of social media, it is uncertain to identify if the content provided is authentic. As quoted in an article by The Harvard Gazzette, "A new study argues that social networks have unwittingly become complicit in amplifying fake news, and that a multidisciplinary effort is needed to better understand how the Internet spreads content and how it is consumed." One of the best examples is the 2021 Capitol Hill Riots, where a lot of fake news has been spread that caused a ripple effect, leading to violent incidents across the United States.

Twitter has been subject to spreading a lot of false news, which could be misleading across several domains. It's crucial to help the afflicted persons as soon as possible for organizations that provide rescue and aid after natural or man-made disasters. The fact that the number of environmental disasters will increase in future and the need for machine learning algorithms to contain and combat the spread of misrepresentative information has become imperative. There are multiple ways in which a tweet can imply false information. For example, the tweet "Today's basketball match was on fire!!!", fire doesn't actually imply that there is an actual disaster, it is just internet jargon used to describe the event. Comparing the above tweet to this tweet "Fire just broke out in the basketball court! Hope everyone is OK", here the tweet actually implies an actual disaster. So, the involvement of an intelligent system that can actually detect if a social media post implying disaster comes into picture.

This classification of tweets requires the data to be clean. For this, Natural Language Processing can be utilised to clean the tweet data and obtain the cleaned tweets. The machine learning algorithms require the input data of text to be in word embeddings, which can be represented in vector format and are helpful in providing the semantic representation of words in low-dimensional space. We explore TFIDF[5] and word2vec models and choose the better fit. Once this is done, the data is sent into four machine learning models (Logistic Regression, XGBoost[2], SVM [4], Neural Network[1]) for training. The statistics - Accuracy, k-fold CV Accuracy, Precision, Recall and F1 score are computed for every model. At the end, the best model is determined using the above computed scores.

2 Method

The project is aiming to build a classifier model to predict the authenticity of a disaster tweet. The dataset is created by the company figure-eight and originally shared on their 'Data For Everyone' website (https://appen.com/pre-labeled-datasets/). The dataset consists of 10,900 hand classified tweets with 4 main columns which are location, keyword, the tweet itself and the target. For the project, a set of basic methods are followed to first understand the data and then extract useful information to build the final working model.

2.1 Exploratory Analysis

The data consists of 4 attributes - keyword, location, tweet and target. Each attribute is to be studied and understood. The importance of each attribute and its information value for the result is explored while data analysis.The data contains many missing values across the keyword and location attributes. The statistic visualizations show that almost 30% of the rows have missing location values. Since keyword is already present in the tweet, it can be ignored. For data exploration, data visualization techniques are used to display various graphs to help visualize the various aspects of data. The corpus corresponding to the 'Relevant' class(target value is 1) which means an actual disaster and 'Irrelevant' class(target value is 0) which mean fake tweets are analyzed separately. The most common words in the total corpus is visualized. It is evident from this bar graph(Figure 1) and wordcloud(Figure 4) that the data needs cleaning. Word Embeddings are visualized as Scatter plots(Figure 5) using latent sentiment analysis - a transformer that performs linear dimensionality reduction by means of truncated singular value decomposition that aids us to choose an embedding model.

2.2 Data Preprocessing

The handling of missing data is important during the preprocessing of the dataset as many machine learning algorithms do not support missing values. We explore different ways of addressing this issue like dropping rows of missing values/ dropping columns of missing values/ prediction of missing values. The columns - keyword and location are dropped as these are basically extracted from the text itself and has no other significance.

From Exploratory Analysis, it is also evident that the most common words are 'https', '#', 'the' and other such URL parts, hashtags and stopwords. For proper extraction of the words from these tweets and to be able to properly vectorized by word2vec embedding model, each tweet is cleaned with the following steps:

- The entire tweet is converted into lower case.
- All mentions of other accounts and hashtags are removed.
- All websites and URLs are deleted.
- All common abbreviations are replaced with their full forms.
- The tweet is stripped down to only have alphanumeric characters to make the dataset free from emoticons, symbols and punctuation or words in other transcripts.
- Any stopwords present are eliminated from the tweet.

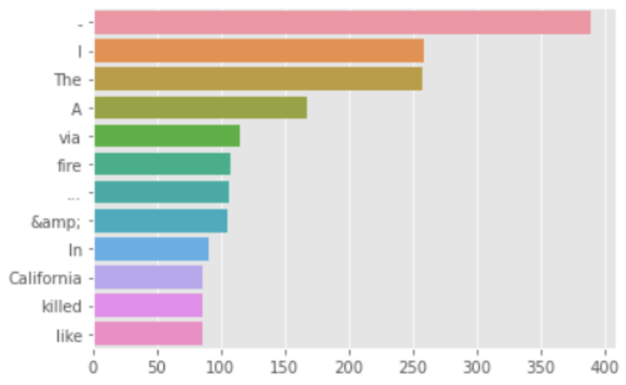


Figure 1: Most common words

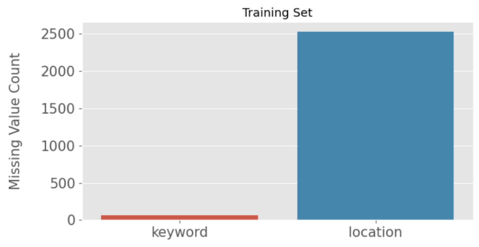


Figure 2: Missing value counts in the columns 'Keyword' and 'Location'

The elimination of all words less than 2 characters is also considered. This particular step is analysed for it's contribution to the model's accuracy. Upon Analysis, it is found that it does not contribute much, hence this step is skipped in the finalized model to make the model cost-effective.

The most critical step in pre-processing text data is to vectorize them. There are multiple ways to vectorize data and all of them fall under two broad categories. TF-IDF technique from dataset-based-corpus methods and word2vec technique from pretrained-global-corpus are selected. From the Scatter plots of these two embeddings on our dataset, it is evident that word2vec can aid the model better since it can be clustered more easily than the plot of TF-IDF. The tweets are now vectorized using word2vec embedding and are ready to be fed into various classification models.

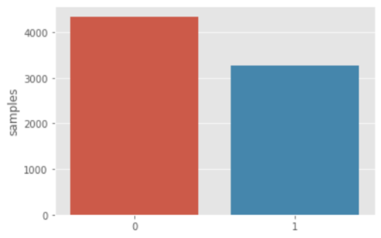


Figure 3: Number of samples in class 0 vs class 1

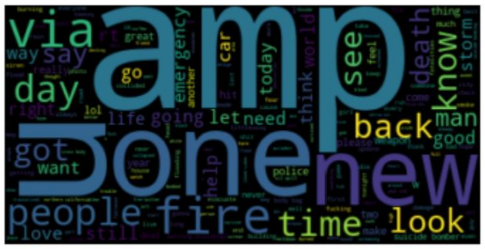


Figure 4: Wordcloud

2.3 Model Building

The project aims at using thoroughly pre-processed data to first segregate into training, testing datasets and later create supervised learning models like Logistic Regression, Support Vector Machine, XGBoost and LSTM neural network.

Logistic regression is a classification technique borrowed by machine learning from the field of statistics. It uses a logistic function to model the dependent variable. The dependent variable is dichotomous in nature, i.e. there could only be two possible classes (ex: either the tweet is authentic or not). In machine learning, support-vector machines are models with associated learning algorithms that analyze data for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM[4] training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

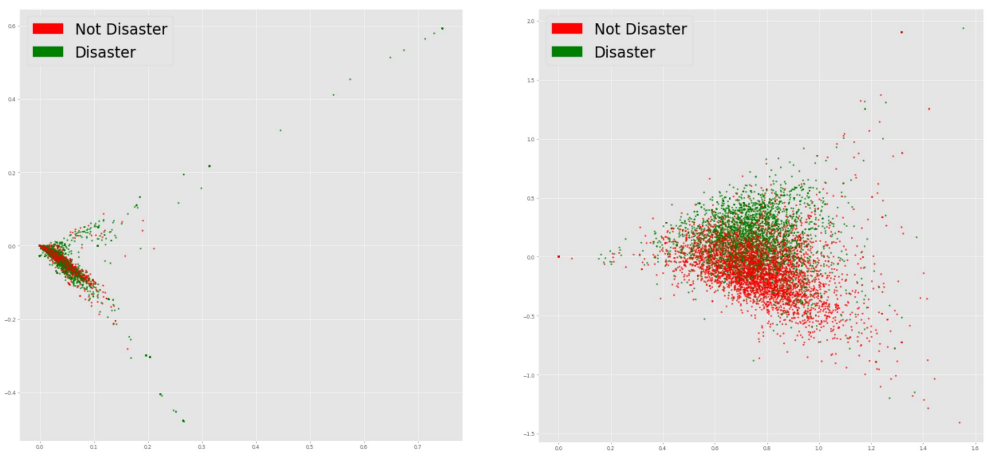


Figure 5: Scatter Plot - Visualisation of TF-IDF and word2vec respectively

XGBoost is a popular and efficient open-source implementation of the gradient boosted trees algorithm. Gradient boosting is a supervised learning algorithm, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models. Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies.

As per K-fold validation metrics(Figure 8), all the three non neural classifiers work better than the Neural network itself. Comparing the accuracies and F1 scores for the non neural models as per Figure 7, XGBoost Classifier gives marginally better results than everything else. To converge on the number of epochs to train XGBoost, we have plotted the training and testing scores for various number of epochs as per Figure 6. It can be inferred from the diagram that there is no improvement in the training accuracy after 50 epochs, hence the final model is run through the same number of epochs.

```
train scores [0.8177339901477833, 0.8520525451559934, 0.9064039408866995, 0.9599343185550082, 0.9824302134646963]
test scores [0.7754432042022325, 0.7951411687458962, 0.8122127380170716, 0.814182534471438, 0.8154957321076822]
```

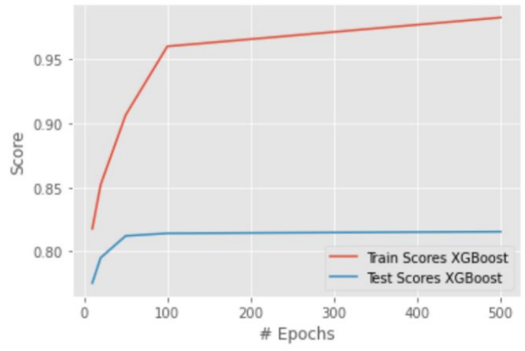


Figure 6: Accuracy of XGBoost for various epochs

2.4 Selecting suitable model

To compare the performances of SVM, LogisticRegression and XGBoost classifiers, suitable metrics and validation techniques like K-Fold Cross validation are explored.

• As the class is imbalanced(Refer to Figure 3), F1 score evaluation metric is used. F1 score is considered as it combines the precision and recall of a classifier into a single metric by taking their harmonic mean. It is primarily used to compare the performance of two classifiers. Recall because the occurrence of false negatives is unaccepted/intolerable, which means its okay to get some extra false positives(false alarms of disasters) over saving some True negatives, i.e, not recognizing actual disasters. As these scores don't differ by much ans show marginal variation, we have also considered K-fold validation accuracy to choose the best classifier.

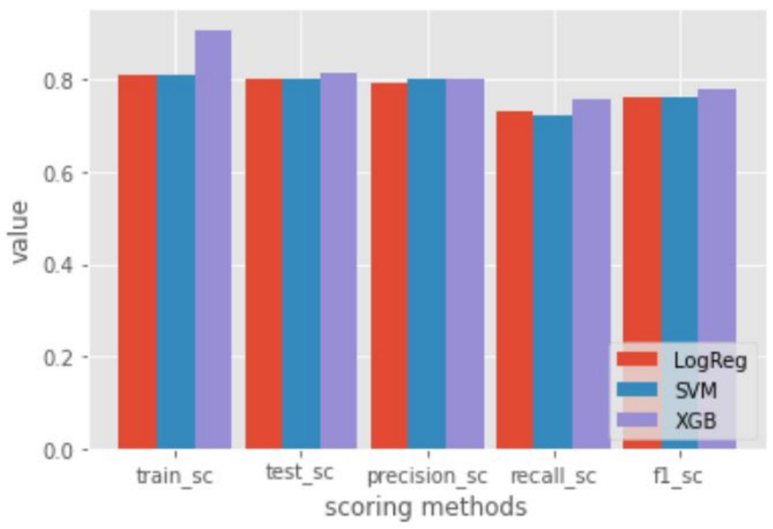


Figure 7: Comparison of Metrics

• K-Fold Validation: K-fold validation is a technique in which the dataset is separated into k partitions where in k iterations, each partition acts as a testing dataset while the rest of the data is used to train the model. This provides a more real-world approximation of measures of performance for a model. The standard choice of number of folds is 5 or 10. For the purpose of this particular problem, number of folds is chosen as 6 because the input is approximately 11k rows and 6 would give around 2k rows for calculating the testing scores which results in a rough 80-20 split of data. 6-fold can give a generalized metric to compare in the given scenario. As per the results in Figure 8, XGBoost performs best even giving around 0.81 accuracy score.

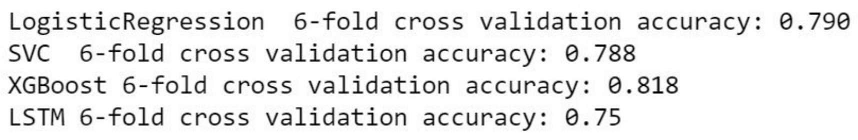

```
LogisticRegression  6-fold cross validation accuracy: 0.790
SVC  6-fold cross validation accuracy: 0.788
XGBoost 6-fold cross validation accuracy: 0.818
LSTM 6-fold cross validation accuracy: 0.75
```

Figure 8: 6-fold Cross Validation Output of all the models

3 Experiment Setup

3.1 Exploratory Analysis and Data Pre-processing

Given the nature of dataset, our assumption was that an ensemble model would outperform a neural network model due to insufficient data and also based on the observation that the average word count for disaster tweets in our data set is in the range of 7-7.5, and for non-disaster tweets is in the range of 4.5-5. We tested this hypothesis on our dataset after cleaning the data and pre-processing it. Some questions we wanted to answer were : What is the structure of data?, what features are important for training the models?, How to compare the models we have chosen? We figured the answers to these questions by performing exploratory data analysis on the data by using plots, data visualization, graphs e.t.c. Initially we decided on feature extraction technique n-gram TF-IDF as it is commonly used for structured data generation but tried additional techniques like word2vec as it promised to give better results as per the embedding visualisations.

The data set has 10,900 rows and 4 columns(ID, Keyword, Location, Text). The team did exploratory analysis of the data. After noticing that the column 'location' has a significantly large number of missing values(Refer to Figure 2) we dropped it. Additionally, the column 'keyword' is also dropped because this is part of the text and is not of much significance on its own. Data visualisation prior to processing as well as for comparing models and results was done by the team:

- Count Plot (Figure 1).

- Bar Graphs (Figure 2,3,7).
- Word Cloud (Figure 4).
- Scatter Plots (Figure 5).
- Line Graph (Figure 6)

The plots helped the team to understand how the tweets were distributed. After interpretation, a few final pre-processing techniques were used to make the data more understandable, easier for the models to interpret and fit. We used the data generated by N-gram word2vec with additional features like word count, URLs in tweets [4], replacing all abbreviations. We concluded that we don't need URLs in our data after examining the results.

- All the mentions, websites, hashtags and URLs are removed.
- Transforming words to lowercase.
- Removing stop words.
- All common abbreviations are replaced with their full forms.
- Vectorizing the words using word2vec.

For text features, we utilized different types of features such as Bag of Words, TF-IDF [5], word2vec features. After analyzing the accuracy results and based on the representation shown in figure [5], we proceeded with word2vec for embeddings.

Since there are only two target classes, we chose binary classifiers like Logistic Regression, Support vector Machine and also explored a bit more complex models like LSTM Neural Network which is known to perform well in the context based semantic space and an ensemble method on decision trees called XGBoost. A comparative analysis of the performance of these models was done.

3.2 Models

• For development of models, the team split the pre-processed data into training and testing in 80-20 ratio. The training set would be used to train supervised learning models like Logistic Regression, Support Vector Machine, XGBoost, LSTM Neural Network that were specified and would be then evaluated by their prediction of authenticity of the tweet. 6-Fold Cross Validation is performed on all the models.

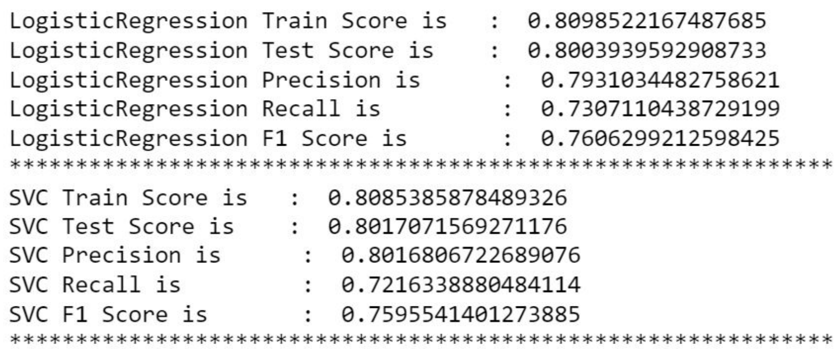
• We set the maximum depth of the tree to 4 for the XGBoost Classifier and used grid search in order to choose better parameters to train XGBoost. It is a boosting algorithm which limits the depth of the trees and also limits the amount of data for training each tree. Thereby avoiding overfitting for each of the boosted trees. Since it is an Ensemble model, it performs better than other simple models like Linear Regression or Support Vector Machine.

• The Neural network - LSTM is chosen as is it known to work well with semantics, various parameters are fine-tuned. When the model was trained with word2vec embeddings, the testing accuracy was very low, approx 60%. When the embeddings were removed, the testing accuracy turned out to be around 72% though training accuracy shoots up to 93% i.e, inferring overfitting. Overfitting could be caused because the dataset is too small (11k sentences) for a neural network that has over 100k trainable parameters. The low accuracy with embeddings can also be attributed to overfitting because the word2vec embeddings is in a dimensional space of 300 vs the dimensional space of bag of words which is 20 for our dataset. The huge dimensional space and low data resulted in a sparse representation, which is difficult to fit even for neural networks.

4 Result

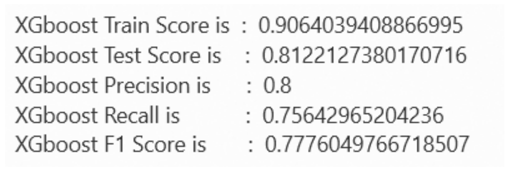
- Data across columns has been visualised using bar graphs.
- The columns Location and Keyword are dropped based on the visualisation(Refer to Figure 1).
- Data pre-processing has been done.
- After a comparative analysis of accuracy, word2vec is finalised for performing vectorization.
- LSTM's testing accuracy with word2vec 60% and with bag of words is 73%. It is inferred that it performs better with bag of words over word2vec embeddings because of the small training dataset which has 8k rows (after splitting 80-20%). Word2vec has to be fitted by the model in a dimensional space of 300 vs the dimensional space of 20 for bag of words. The 8k rows of data is too low for the LSTM to learn in the 300 dimensional space. Therefore, it does better with bag of words.

- As per K-fold validation metrics, all the three non neural classifiers work better than the neural network as the later succumbs to overfitting. The Training accuracy increases to 93% but the testing accuracy falls down by 10% in just 10 epochs which can be clearly attributed to the low amounts of data we have in our dataset as well as the fact that the number of trainable parameters for the neural network are very high, approx 100k. The huge difference in sizes of trainable parameters and training data set can easily lead the model to overfit.
- The combined metrics of (6-fold cross validation accuracy, Precision, Recall, F1 Score - from Figure 9,10) proved that XGBoost can fit the data better than Logistic Regression, Support Vector Machine and LSTM neural network.
- The comparative analysis of the models based on the evaluation metrics shows that XGBoost provides better results with accuracy around 82%. XGBoost performs better because of two main reasons:
  - It provides accurate results by sequentially building shallow decision trees and it avoids overfitting by regularization through both LASSO (L1) and Ridge (L2) and as it is an ensemble method aggregating many decision trees by gradient boosting.
  - It fits the training data much better than Logistic regression/SVM, because while encountering missing values on nodes, it learns which path to take for the missing values in future, but the other two do not perform well with the missing data.
  - It also uses a model complexity penalty term that protects loss of accuracy.



```
LogisticRegression Train Score is   :  0.8098522167487685
LogisticRegression Test Score is    :  0.8003939592908733
LogisticRegression Precision is      :  0.7931034482758621
LogisticRegression Recall is         :  0.7307110438729199
LogisticRegression F1 Score is       :  0.7606299212598425
**************************************************************
SVC Train Score is   :  0.8085385878489326
SVC Test Score is    :  0.8017071569271176
SVC Precision is      :  0.8016806722689076
SVC Recall is         :  0.7216338880484114
SVC F1 Score is       :  0.7595541401273885
**************************************************************
```

Figure 9: Logistic Regression and SVC Output



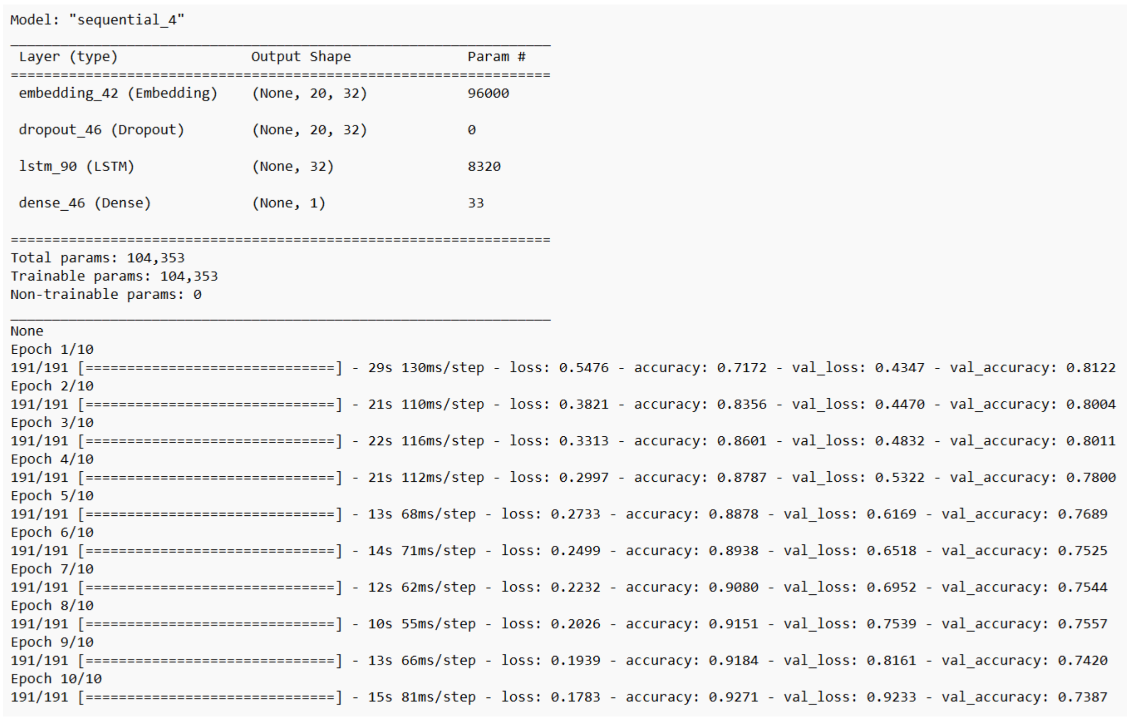
```
XGboost Train Score is  :  0.9064039408866995
XGboost Test Score is   :  0.8122127380170716
XGboost Precision is    :  0.8
XGboost Recall is       :  0.75642965204236
XGboost F1 Score is     :  0.7776049766718507
```

Figure 10: XGBoost Output

5 Conclusion

- Exploratory Data Analysis is performed on the data and nonessential columns have been dropped and Data pre-processing has been completed. On comparison between TF-IDF and word2vec embeddings for this dataset, word2vec gives better results.
- Initially, Logistic Regression and Support Vector Machine are considered for a comparative analysis of accuracy. We found that the performance of Logistic Regression model classification is better than the SVM model classification.
- We considered XGBoost classifier and LSTM Neural Network for further depth of the project. As per stats, XGBoost, Logistic Regression, SVM perform better over Neural Network(LSTM). This could be explained by the fact that LSTM only associates weights with the features, rather than finding new ones.
- Exploring various other datasets and acquiring a larger dataset can definitely result in an improved accuracy for the neural network.

- XGBoost model's performance can be enhanced by further fine-tuning as it has a very comprehensive tunable parameters.
- Various embeddings like BERT, GLOVE can be tested to check if they contribute to improve the accuracy of the project.



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_42 (Embedding)    (None, 20, 32)            96000

 dropout_46 (Dropout)        (None, 20, 32)            0

 lstm_90 (LSTM)              (None, 32)                8320

 dense_46 (Dense)            (None, 1)                 33

=================================================================
Total params: 104,353
Trainable params: 104,353
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/10
191/191 [==============================] - 29s 130ms/step - loss: 0.5476 - accuracy: 0.7172 - val_loss: 0.4347 - val_accuracy: 0.8122
Epoch 2/10
191/191 [==============================] - 21s 110ms/step - loss: 0.3821 - accuracy: 0.8356 - val_loss: 0.4470 - val_accuracy: 0.8004
Epoch 3/10
191/191 [==============================] - 22s 116ms/step - loss: 0.3313 - accuracy: 0.8601 - val_loss: 0.4832 - val_accuracy: 0.8011
Epoch 4/10
191/191 [==============================] - 21s 112ms/step - loss: 0.2997 - accuracy: 0.8787 - val_loss: 0.5322 - val_accuracy: 0.7800
Epoch 5/10
191/191 [==============================] - 13s 68ms/step - loss: 0.2733 - accuracy: 0.8878 - val_loss: 0.6169 - val_accuracy: 0.7689
Epoch 6/10
191/191 [==============================] - 14s 71ms/step - loss: 0.2499 - accuracy: 0.8938 - val_loss: 0.6518 - val_accuracy: 0.7525
Epoch 7/10
191/191 [==============================] - 12s 62ms/step - loss: 0.2232 - accuracy: 0.9080 - val_loss: 0.6952 - val_accuracy: 0.7544
Epoch 8/10
191/191 [==============================] - 10s 55ms/step - loss: 0.2026 - accuracy: 0.9151 - val_loss: 0.7539 - val_accuracy: 0.7557
Epoch 9/10
191/191 [==============================] - 13s 66ms/step - loss: 0.1939 - accuracy: 0.9184 - val_loss: 0.8161 - val_accuracy: 0.7420
Epoch 10/10
191/191 [==============================] - 15s 81ms/step - loss: 0.1783 - accuracy: 0.9271 - val_loss: 0.9233 - val_accuracy: 0.7387
```

Figure 11: LSTM Output

6 Github Link

https://github.ncsu.edu/sngudipa/engr-ALDA-Fall2022-P39

References

[1] Oluwaseun Ajao, Deepayan Bhowmik, Shahrzad Zargari. IEEE Conference on Fake News Identification on Twitter with Hybrid CNN and RNN Models, 2018.

[2] Julio C. S. Reis, Andre Correia, Fabrıcio Murai, Adriano Veloso, and Fabrıcio Benevenuto. IEEE Publication on Supervised Learning for Fake News Detection, 2019

[3] Zaitul Iradah Mahid, Selvakumar Manickam and Shankar Karuppayah. IEEE Conference on Fake News on Social Media: Brief Review on Detection Techniques, 2018.

[4] Khairisyah Yuliani Firlia, Mohammad Reza Faisal, Dwi Kartini, Radityo Adi Nugroho and Friska Abadi Analysis of New Features on the Performance of the Support Vector Machine Algorithm in Classification of Natural Disaster Messages, 2021.

[5] Ganesh Nalluru, Rahul Pandey, Hemant Purohit IEEE International Conference on Smart Computing(SMARTCOMP): Relevancy Classification of Multimodal Social Media Streams for Emergency Services, 2019.