Detection of offensive YouTube comments, a performance comparison of Deep Learning approaches

PRIYAM BANSAL

D17129566

A dissertation submitted in partial fulfillment of the requirements of
Technological University Dublin for the degree of
M.Sc. in Computer Science (Data Analytics)

2019

DECLARATION

I certify that this dissertation which I now submit for examination for the award of MSc. in Computing (Data Analytics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technological University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: **_____PRIYAM_BANSAL_______**

Date: **September 1, 2019**

1 ABSTRACT

Social media data is open, free and available in massive quantities. However, there is a significant limitation in making sense of this data because of its high volume, variety, uncertain veracity, velocity, value and variability. This work provides a comprehensive framework of text processing and analysis performed on YouTube comments having offensive and non-offensive contents.

YouTube is a platform where every age group of people logs in and finds the type of content that most appeals to them. Apart from this, a massive increase in the use of offensive language has been apparent. As there are massive volume of new comments, each comment cannot be removed manually or it will be bad for business for youtubers if they make their comment section unavailable as they will not be able to get any feedback of any kind.

For most, the best approach is to use sentiment analysis where a machine-learning model learns the problem and does the work for humans by detecting the emotion in content. As the problem discussed is of offensive YouTube comments many things are kept in mind. Sentiment analysis is a challenging task as each platform has different writers and different content, meaning there is no one-size-fits-all solution. The vast majority of comments on YouTube do not contain offensive content. This results in imbalanced data which must be taken into consideration when designing a model to detect this content.

This research employs state-of-the-art deep learning techniques from natural language processing to detect offensive content in YouTube comments. The precision and recall metric is used for this research where more focus is given to recall as it tells how many of the correct output was exactly found. All the performance metrics are taken into consideration to understand results for both offensive and non-offensive class equally. While most existing research only reports on overall model accuracy, this experiment pays close attention to the model's ability to detect offensive content specifically (recall), as this is a very important metric for offensive content detection models.

The research evaluates sampling techniques which can be used to compensate for imbalanced data, as well as a variety of deep learning approaches which can be used to train an effective model. It finds (with statistical significance) that a Hybrid approach of CNN-LSTM model with Synthetic Minority Oversampling is the most effective approach when compared with SVM baseline and CNN model with the recall of 33%. LSTM and LSTM-CNN models are also shown to be effective in this domain.

Keywords: *Sentiment analysis, offensive, SVM*, CNN*, LSTM*, sampling techniques, KFold cross-validation, Recall*

****SVM*** *– Support Vector Machine*

****CNN****- Convolutional Neural Network*

****LSTM****- Long Short Term Memory*

2 ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor **Jack O' Neill**, for his encouragement, constructive suggestions, recommendations and tremendous support throughout the dissertation process. I would have been lost without his guidance and pieces of advice.

I would like to thank **Hao Chen** (DIT research Staff) for providing me with data for this research from **Maral Dabvar**.

Finally, I would like to acknowledge the love and support that my friends and family showered during my studies. Big thanks to my friends **Arshdeep Kaur, Yogesh Huddar and Tushar Joshi** and my family members helping me out for this research in every way possible. This would not have been possible without them.

TABLE OF CONTENT

TABLE OF FIGURES

TABLE OF EQUATIONS

TABLE OF TABLES

CHAPTER 1 – INTRODUCTION

1.1. Background

YouTube is one of the largest and most popular online video sharing platforms. Recently, it has faced criticism for the proliferation of abusive content on its platform[1] When someone joins YouTube, he/she joins a community of people from all over the world. Everybody has their opinion and no one can stop them from expressing their emotions or thoughts. YouTube gives us the facility to disable the comments but doing this will lead to bad business and doing this prevents feedback on whether people liked the video or not.

Comments, whether positive or negative, can lead to improvements in a channel as well as inspiring new video ideas. Removing comments manually is also not a great option because one-by-one reading and removing have to be done. Analysis of these sentiments becomes extremely useful in helping people organize and choose channels to watch and extracting meaningful information from opinions is a critical portion of analysis. Computers and algorithms are wonderful for addressing large-scale problems.

Offensive content is not a problem for the uploaders but a very big problem for the people who join YouTube for entertainment. All ages of people access YouTube from children to older people and there is a whole lot of content to watch so this is creating issues as people are getting affected by this much hate speech. YouTube has made some improvements in their policies; recently closing the private message feature. Similarly, in February 2019, channels with kids have to disable the comments. These are some good steps but still lots of improvements have to be made. Offensive material does not just include abusive but also sarcastic and racist comments which are equally hurtful as abusive words.

Artificial Intelligence techniques are a highly active area of research. Text analytics, natural language processing (NLP), network behavior monitoring and machine learning techniques are used to develop and train algorithms to support pre-publication moderation. Deep learning, a branch of machine learning based on neural networks, is now delivering higher accuracy rates across many complex datasets. And for this mentioned problem Artificial Intelligence seems to be a good approach to detect these comments by learning on its own (Anagnostou, Mollas, & Tsoumakas, 2018).

[1] See, for example https://www.bbc.com/news/technology-47408969

1.2. Research Project/problem

Offensive comments detection requires Natural Language Processing. This project asks which end-to-end Machine Learning approach is most effective in detecting offensive comments on YouTube.

To do this analysis the first problem that is addressed is that of unbalanced datasets which can give biased results while predicting the output. To overcome this, different techniques of sampling the training data are used, such as over-sampling, under-sampling, SMOTE (Synthetic Minority Over-sampling Technique) and stratified K-Fold sampling. Different models will be trained to detect the offensive and non-offensive words to tell us whether the comment is offensive or not. Different sampling techniques can make a huge difference in the results and in training the model as it is necessary for both the classes to be trained correctly by a model.

As textual data is involved, the sentences must first be processed in order to make them understandable for models to learn. This will be done using different Natural Language Processing approaches.

Models will be trained with machine learning and deep learning algorithms such as Support Vector Machine (which will be considered as a baseline for this research), Convolutional Neural Network, Recurrent Neural Network Long Short-Term Memory and a hybrid form of Convolutional Neural Network/Recurrent Neural Network, and vice versa.

Neural Networks have hidden layers which explain how the model will work and the size and number of these layers can be adjusted in order to fit the data best. This experiment will analyse negative sentiments, emotions and thoughts of different people. It will show which algorithms and sampling techniques lead to better learning of models for this dataset. Analysing different models and different sampling techniques will help us in answering the research question.

1.3. Research Objectives

The objective of this research is to compare Support Vector Machines and Deep Neural Networks (which are Convolutional Neural Network, Long short-term memory, mixed Convolutional Neural Network-Long short-term memory and Long short-term memory-Convolutional Neural Network) on YouTube comments to find out which method will give better performance metrics using different sampling techniques. The study investigates the performance impact of oversampling the minority class randomly, creating synthetic instances of the minority class using the SMOTE technique, under-sampling the majority class randomly and using Stratified K-Fold sampling to balance the unbalanced data. Their performance on different Artificial Intelligence algorithms mentioned above is compared finding which model works best with which sampling technique for the YouTube Comments dataset after using Natural Language Processing for comments processing.

1.4. Research Methodologies

Review of the literature is conducted to examine issues, processing, features, methods and various algorithms used in sentiment analysis for detecting hate or offensive speeches. Secondary data is used for this research of offensive YouTube comments which is labeled manually by researchers (Dadvar, Trieschnigg, & Jong, 2014) which will be classified using some Artificial Intelligence algorithms. The research is an Empirical study where knowledge will be gained by performing experiments and direct observations. The reasoning approach for this research is deductive as a theory is made that will be accepted or rejected by performing some steps.

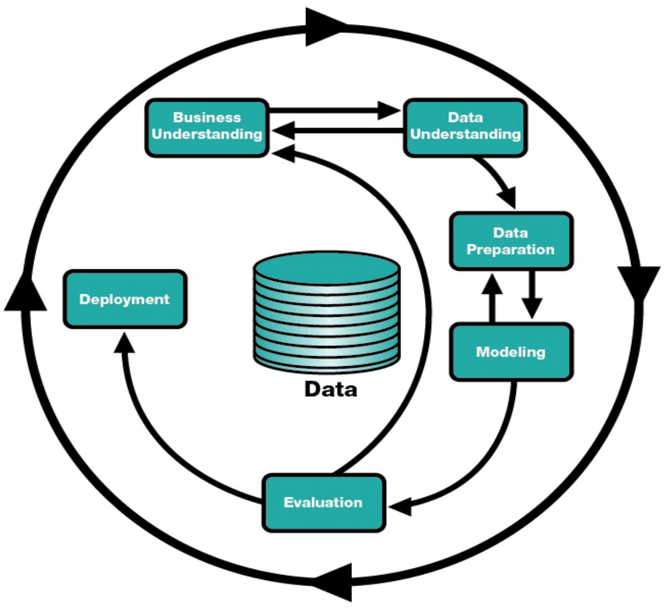


Figure 1.1: Cross-industry process for data mining (CRISP-DM)[2]

[2] https://docplayer.net/6187990-Business-intelligence-tutorial-for-rapid-miner-advanced-decision-tree-and-crisp-dm-model-with-an-example-of-market-segmentation.html

The research methodology broadly follows the well-known Cross-Industry Process for Data Mining (CRISP-DM) as illustrated in Figure 1.1: Cross-industry process for data mining (CRISP-DM) below which is a leading data mining process model. The analysis framework includes business understanding (see section 3 below), data understanding (see Section3.2 below), data preparation (see section 3.3 below), modelling (see section 3.7 below) and evaluation (see section 3.8 below).

1.5. Scope and Limitations

Sentiment analysis is a huge area where different types of data are used and analyzed for sentiment, YouTube itself is a very big area to cover and new datasets are developed for processing and classification. For this research, the dataset is ***YouTube_Dadvar_2014*** *by* (Dadvar, Trieschnigg, & Jong, 2014). A number of Artificial Intelligence algorithms are used on this data *i.e.* Natural Language Processing (NLP) for processing the text, Support Vector Machine (machine learning algorithm), Convolutional neural network, Recurrent Neural Network and their mixed forms (deep learning algorithms) for classification and performance comparison between them. The major scope of the research is to tackle the class imbalance problem using various sampling methods such as Random Under-sampling, Random Oversampling, and SMOTE (Synthetic Minority Over-sampling Technique) and stratified K-fold cross-validation algorithm and to build an effective model that predicts the offensive comments from YouTube.

A major limitation is processing the data as on social media mainly texting language is used which is not considered as dictionary words so a huge amount of pre-processing is required for this. In this research, efforts were made to process it but, with access to more computing power, more can be done which may lead to making a better embedding or vector. Another limitation is the size of the dataset as it is comparatively small for use on deep learning models but still, algorithms are trained well and adding more similar data can increase the accuracy of the results. A limitation that is recognized for this thesis is lack of access to high-powered computing resources, which could enable further training of models, and allow full evaluation of all the models under examination.

1.6. Document Outline

The outline of this Thesis report is given below.

Chapter 2- Literature Review discusses the literature related to sentiment analysis for different text datasets. It considers the class imbalance problem, sampling methods, processing techniques, feature extraction and predictive modelling algorithms used and worked for various researches. It includes the state of art for this research, related work and gaps in the research for those papers.

Chapter 3- Design and Methodology discuss the design of the research carried out. Each phase of CRISP-DM methodology followed is discussed in detail here. The process of obtaining the data, cleaning the data, transforming the data, data splitting, understanding the imbalanced nature of data, training the Support vector machine and Deep neural networks used for this research, evaluation of the models and performance measures used are discussed in this chapter.

Chapter 4- Result, Evaluation and Discussion present the results of the models used with different sampling techniques as the first section followed by their implementation and evaluation and finally the discussion which explains the results and the implementation and the final result for explaining the best-performed model is stated.

Chapter 5- Conclusion summarises the research carried out for this thesis. It discusses the contribution of the research towards the research question and briefly explained how the results are achieved. The chapter concludes with discussing areas of future research that can be carried out after this.

CHAPTER 2 – LITERATURE REVIEW AND RELATED WORK

2.1. Background

Text classification is a very large area and a lot of work has been done on that and is still continuing. In processing text, it has been seen that there is no proper set of rules or procedures and the result can totally differ based on the data or the type of data. This section includes the basic background of the approach selected, and related work carried out in the area of Sentiment Analysis

2.1.1. Sentiment analysis

Sentiment Analysis is a wide area of analysis where a lot of work is going on. It is the process of categorizing or predicting any opinion, text or any writing into positive, negative or neutral sentiments (referred to as Polarity) but there are other types of sentiment analysis also which give more precise results in the polarity level as very positive, positive, neutral, negative, and very negative (Tripathy, Agrawal, & Rath, 2016). Based on the application or the data there can be sentiments ranging from very negative to very positive. There can be different types of areas and applications for this which can also have different types of formats to be presented in as articles or government policies in document form, comments or reviews in a sentence to feedbacks in word or entity (Rojas-Barahona, 2016). (Zhang, Yuan, Wang, & Zhang, 2017)It can be on the basis of anger, fear, joy, sadness, sarcasm, *etc.* where the positivity of emotion is referred as polarity for that analysis. It has been practiced in many datasets such as Twitter, Wikipedia (Wulczyn, Thain, & Dixon, 2017) government policies, movie reviews, customer feedback, Multi-Perspective Question Answering (MPQA) corpus of news documents (Wiebe et al, 2005) The MPQA Opinion Corpus contains news articles from a wide variety of news sources manually annotated for opinions and other private states, web customer review data (Hu and Liu, 2004), Amazon review data (Blitzer et al., 2007) and many more (Uryupina, Plank, Severyn, Rotondi, & Moschitti, 2014). Researchers have developed and modified their work on this from past experience of past work that is going on using different datasets or different tools or different algorithms.

2.1.2. Evaluation metrics

There are some metrics for evaluating machine learning algorithms such as accuracy, loss function, confusion matrix, Area Under ROC (Receiver Operating Characteristic) Curve (AUC-ROC), F1-score, Mean Absolute Error (MAE) and Mean Square Error (MSE). Models give different results for different metrics, so while accuracy may be good for a certain algorithm another metric, say F1-score, can be very low. Usually, accuracy is defined as the number of correctly predicted data points as a proportion of all predictions but other metrics should be considered to have a true judgment of model.

F1 score is a harmonic mean between precision and recalls where Precision measures the accuracy of the model when predicting positive labels (see Equation 1). Recall measures the proportion of true positive labels which are successfully predicted as such (see Equation 2: Recall). These metrics are for binary classifiers. AUC-ROC is also most widely used for evaluation. It shows how much the model is capable of distinguishing between classes; if near to 1 it demonstrates a good measure of separability while a poor model has value near to 0. The ROC curve is plotted with TPR (True Positive Rate) against the FPR (False Positive Rate).

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \qquad - Equation\ (1)$$

Equation 1: Precision

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \qquad - Equation\ (2)$$

Equation 2: Recall

2.2. State-of -Art

This section includes the State-of-art of this research where general documentation of techniques used is stated. Also, the level of development reached and understood for working on this research.

2.2.1. Data Balancing

There are two main problems imposed by data with unequal classes: Machine problems where models are much likely to learn the majority class than the minority, and intrinsic problems where false values will be higher than true values (Kumar & Sheshadr, 2012). There are many solutions used for this, some of which are discussed below (Singh & Purohit, 2015):

Random Over–Sampling

This approach increases the number of minority class instances randomly in the training set using sampling with replacement and matches it with the majority class. An advantage of this approach is there is no data loss as all the observations from minority and majority class are kept but a big disadvantage of this is that it is prone to over-fitting as the minority class consists of many duplicates.

Random Under-Sampling

This approach decreases the number of majority class instances in the training set and matches the count of minority class using random sampling. As it is removing the observations from a dataset this can throw away meaningful information but can improve the model's overall performance.

Synthetic Minority Oversampling Technique

SMOTE (Synthetic Minority Oversampling Technique) considers the K nearest neighbors of the minority instances. This is a statistical technique for increasing the number of cases in training data in a balanced way. It constructs feature space vectors between these K neighbors, generating new examples that combine features of the target case and features of its neighbors. This increases the features available for each class and makes samples better and more general

Stratified K-Fold

This is used to ensure that training and validation datasets both contain the same percentage of classes. Stratification is the process of rearranging the data to ensure each fold is a good representative of the whole. For example in a binary classification problem where each class comprises unbalanced data, it is best to arrange the data such that in every fold, each class comprises the same ratio of positive and negative instances. Stratified K-Fold cross-validation ensures this.

2.2.2. Baseline model

Baseline models are existing state-of-the-art models against whose metrics or performance will be compared with other methods considering it as the base and to make continuation work effectively. (Wang & Manning, 2012). Baseline models give us a target to compare our proposed models against, to make sure that they improve on existing work. There have been previous papers where a baseline model related to the topic discussed in this paper (Lai, Xu, Liu, & Zhao, 2015) or have not considered at all (Wang, Jiang, & Luo, 2016).

SVM: Naive Bayes (NB) and Support Vector Machine (SVM) are often used as a baseline for text classification it has been seen (Wang & Manning, 2012) that support vector machines work well for text as compared to other machine learning model. It is an algorithm that makes a decision boundary between different groups separating them. The simplest way to separate two groups is with a straight line (1-D), a flat plane (2-D) or an N-dimensional hyperplane (N-D). Hyperplane(s) maximizes the margin between the two classes. One of the reasonable choices for best hyperplane is the one representing the largest separation, or margin, between the two classes.

However non-linear planes have also proved effective results. SVM decides where to make the boundary and can get the best line between the vector groups. Below is the representation of how SVM works in Figure 2.1 (Pang, Lee, & Vaithyanathan, 2002). Linear or non-linear is handled by a kernel function to map the group into different spaces. SVMs for non-linear classification use what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

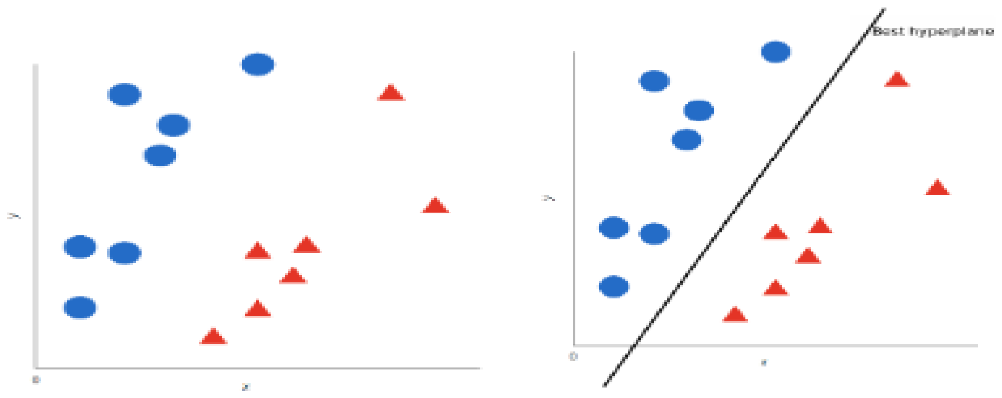


Figure 2.1: SVM architecture dividing two classes by linear hyperplane[3]

[3] https://www.datacamp.com/community/tutorials/support-vector-machines-r

2.2.3. Natural Language Processing

NLP is used in text analysis and speech analysis. It is being used in day-to-day life, Gmail has spam detection techniques, and Siri (Apple's virtual speaking assistant) has speech recognition techniques. Research in this area is ongoing since 1940. (Gaydhan, Doma, Kendre, & Bhagwat, 2018) Some of the basic steps for text processing in NLP are tokenization, Lemmatization, Stemming, stop-words, Regular expressions, bag-of-words, Term Frequency – Inverse Document Frequency (TF-IDF).

The purpose of carrying out NLP is paramount to decrease the number of attributes used during the process (Maas, et al., 2011). These steps are described below:

Tokenization first it is explained as partitioning or splitting the data into tokens. Tokens can be of any form; words, sentences or a defined sequence.

Lemmatization and stemming are used to reduce inflectional forms or removing the prefixes and suffixes i.e. formatting the grammatical categories of a word for example playing, played, plays will be formatted to play.

Stop words are the words that are used very frequently in a sentence but do not have some meaning in the sentence. A machine doesn't understand sentence meaning it matches the patterns for text analysis so the stop words like 'a', 'an', 'the', 'some' and many more can be considered from the sentences as words and removed or recognized. (Schmidt & Wiegand, 2017) It is not always considered as necessary because in some cases it can be useful as some words if removed decrease the sentiment polarity strength so to avoid decreasing the essential data custom stop words can be used or no need to remove these words at all.

Regular Expressions define search patterns for characters, words, numbers, punctuation, whitespace, or any pattern of letters. This refines the sentence or the document which cannot be detected by lemmatization or stop words. Regular expressions help us make custom formatting for the words.

Bag-of-words: After refining and cleaning and pre-processing the data features are extracted as machine learning cannot work on the text directly so it is converted into vectors. This is a popular approach for classification due to its efficiency and simplicity. It transforms the sentence into a collection of words which are used to classify the sentence. The Bag of Words approach has one downside in that it considers every word separately and no pattern is considered. To get over this there is another similar approach called n-gram which groups words of a sentence on the basis of a sequence of words as bigram is a sequence of two words; trigram is sequence of three words and so on this also faces same problem (Alberto, Lochter, & Almeida, 2015).

TF-IDF: This comes into the picture where there is a problem with scoring word frequency. TF-IDF is Term Frequency-Inverse document frequency evaluates the importance of a word in sentence or document. This will finally tell how relevant the term is in the document is. The TF-IDF is expressed as a proportion of the number of words in the documents in corpus which tells that some words appear more frequently in a document in general and it increases proportionally to the number of time a word appears in a document.

Word Embedding: Nowadays there is a huge use of word embedding with deep neural networks for NLP tasks. (Ghosal, Bhatnagar, Akhtar, Ekbal, & Bhattacharyya, 2017) This is a distributed representation of words. Word embeddings can capture meaningful syntactic and semantic regularities; contextual information, Word2Vec; or correlation between words,

Glove: This represents words that occur with similar context (Pennington et al., 2014). Each word represented by a vector has mostly tens or hundreds of dimensions. Simple approaches to using pre-trained word embeddings for comment representation include averaging (Djuric, et al., 2015) and concatenating (Rojas-Barahona, 2016) the word vectors of all words in the comment. GloVe, FastText and Word2Vec are some of the pre-trained neural network models. There are many more but these two are the most used models for English. A pre-trained model is a set of word embeddings that has been created somewhere else and is being used for the embedding matching matrix. These are huge datasets with billions of different words and are a vast corpus of language that captures statistically robust word meanings. They are also available in different languages other than English. Using pre-trained models saves the time spent in processing another large datasets for words.

2.2.4. Neural Network

The most basic form of the definition of Neural Network can be ‘a computer system modeled on the human brain and nervous system.’ It can adapt inputs that frequently change so the network doesn’t have to redesign output and generates best possible results for that (Siersdorfer, Chelaru, Nejdl, & Pedro, 2010). Because of this feature it is widely used for pattern recognition. As the human brain consists of neurons there are nodes that receive inputs which are part of layers. Based on the input they provide some output which can again be taken as input for another node or say as in human body neuron. (Chen, Mckeever, &

Delany, Abusive Text Detection Using Neural Networks, 2017) Deep learning and NN are closely connected as when there are more to the layers of the network it is termed as deep learning. (Chen, Mckeever, & Delany, 2018)A deep learning system is self-teaching, learning as it goes by filtering information through multiple hidden layers, in a similar way to humans.

Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a deep learning algorithm. (Georgakopoulos, Tasoulis, Vrahatis, & Plagianakos, 2018). CNN works well with spatial data (Georgakopoulos, Tasoulis, Vrahatis, & Plagianakos, 2018). These were previously used only for the letter as seen in a research paper (Niu & Suen, 2012) which uses an MNIST data where classification of recognizing handwriting is done or small simple image classification but now it is being used for self-driving cars, video recognition complex image processing.

The image is effectively checked for patterns in some specific section by using filters which are stacks of weights. Pooling is done on that another layer where input image is partitioned for output value. Convolution + relu layer followed by pooling layer make the whole model whose final output is classified. (Waseem, Davidson, Warmsley, & Weber, 2017)The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.

This has worked well on text datasets also (Yin, Kann, Yu, & Schütze, 2017). CNN gave some promising results for NLP. Figure 2.2 shows the basic architecture of the CNN model when text classification is done. By varying kernel sizes and concatenating outputs it allows detecting patterns of multiple sizes.

Based on computation time CNN seems to be much faster than RNN. Convolutions are a central part of computer graphics and implemented on a hardware level on GPUs. Applications like text classification or sentiment analysis don’t actually need to use the information stored in the sequential nature of the data. Predicting whether the sentiment was good or bad a CNN model may be sufficient and even better in terms of computation.

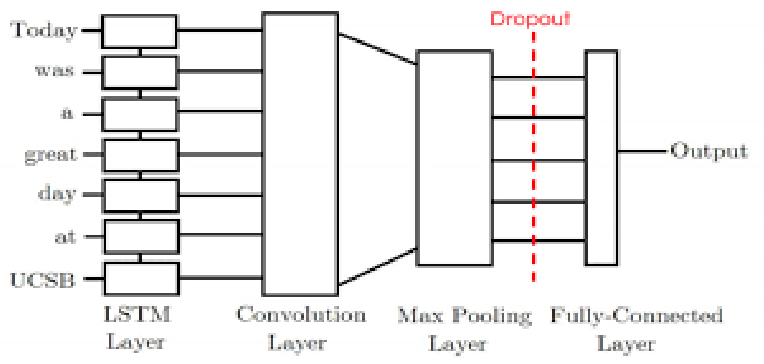


Figure 2.2: Basic Convolutional Neural Network architecture for Text classification[4]

Recurrent Neural Network

A Recurrent Neural Network (RNN) is a type of deep learning algorithm where data moves only in forward direction as in the form of feedback for another layer. (Lai, Xu, Liu, & Zhao, 2015) The predictions are done by memorising the inputs. These are most widely used for speech recognition, translation etc. It holds the data about previous layer in sequence and calculates a predicted word vector at a particular timestamp. Here the problem with RNN arrives as the weights are same for each timestamp so LSTM (Long Short Term Memory) and GRU (Gated Recurrent Unit) came into the picture.

Remembering information for long periods of time is practically LSTMs default behavior, not something struggling to learn as basic RNN behavior (Pitsilis, Ramampiaro, & Langseth, 2018). The LSTM does have the ability to remove or add information to the layer carefully regulated by structures called gates. The LSTM network is trained using backpropagation (BP) over time and can effectively address this problem.

GRU is like a long short-term memory (LSTM) with ‘forget gate’ but has fewer parameters than LSTM, as it lacks an output gate. LSTM can easily perform unbounded counting that is the reason it is strictly stronger than GRU for some cases (Pitsilis, Ramampiaro, & Langseth, 2018)

[4] http://konukoii.com/blog/2018/02/19/twitter-sentiment-analysis-using-combined-lstm-cnn-models/

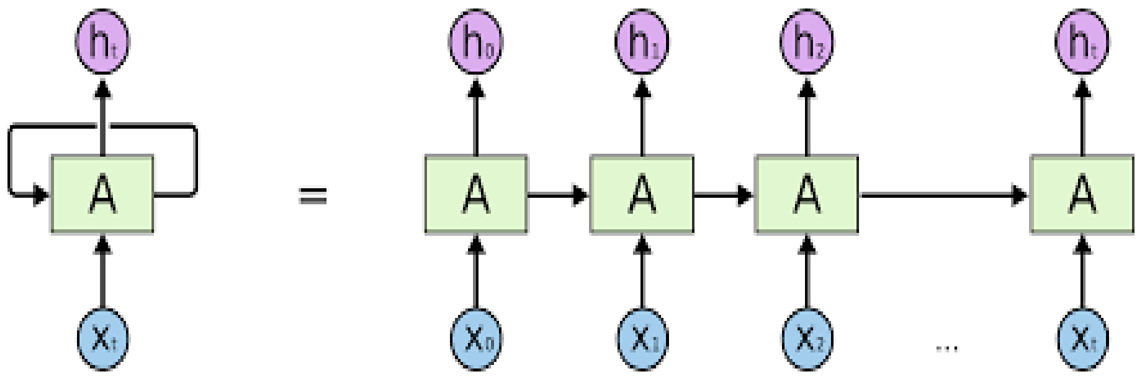


Figure 2.3: Basic Recurrent Neural Network architecture for Text classification [5]

CNN-LSTM

Figure 2.4, below, shows the mixed CNN & RNN model in (Zhang, Yuan, Wang, & Zhang, 2017) with text as input and a fully connected layer giving the output of class, which consists of two main components: Convolutional neural network (CNN) and long short term memory network (LSTM).

This model is used when their input has spatial structure (Yin, Kann, Yu, & Schütze, 2017) as words in sentence, images, paragraph or documents with temporal structure in input i.e. words in text, order of images in a video or require the generation of output with temporal structure such as words in a textual description.

"[CNN LSTMs are] a class of models that is both spatially and temporally deep, and has the flexibility to be applied to a variety of vision tasks involving sequential inputs and outputs" (Donahue, et al., 2016)

LSTM-CNN is similar kind of hybrid model of deep learning where CNN will be added after the LSTM layer which helps the model to backpropagate and CNN will look at it through different patterns.

[5] https://medium.com/jatana/report-on-text-classification-using-cnn-rnn-han-f0e887214d5f

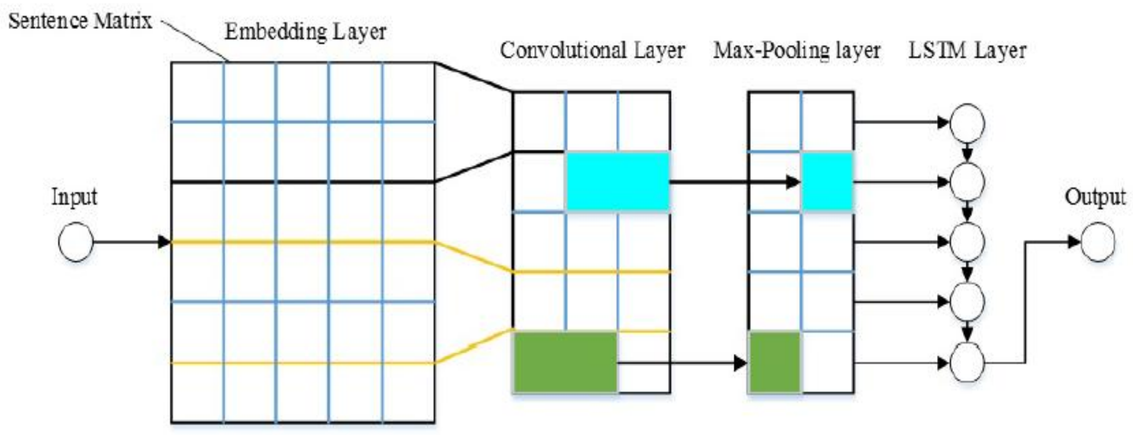


Figure 2.4: CNN-LSTM architecture explained through diagram (Zhang, Yuan, Wang, & Zhang, 2017)

2.3. Related work

YouTube is a very wide area which has videos of all the topics. It's like Google with videos. There is much work going on and done before on videos or comments or video title classification. Comments areas can be viewed by anybody and on lots of topics, people show all types of emotions (Waseem, Davidson, Warmsley, & Weber, 2017). As text data is involved first words are separated by text analysis and then by processing it well before applying deep learning models. In other papers, researchers have considered overall social media or other media sites to conduct experiments (Shylaja & Chavan, 2015). Researchers have used SVM (Niu & Suen, 2012), N-Gram (Uryupina, Plank, Severyn, Rotondi, & Moschitti, 2014), CNN (Georgakopoulos, Tasoulis, Vrahatis, & Plagianakos, 2018) and many more text analysis, Natural language processing (Gibert, Perez, García-Pablos, & Cuadros, 2018), machine learning and deep learning approaches to find the best fit for the problem and to what extent the accuracy can be obtained (Djuric, et al., 2015), (Uryupina, Plank, Severyn, Rotondi, & Moschitti, 2014), (Waseem, Davidson, Warmsley, & Weber, 2017). The paper (Anagnostou, Mollas, & Tsoumakas, 2018) has classified the comments into different categories, comments rating on different areas and according to those areas classification is done. Precision-Recall curve is formed on the basis of categories which used SVM, Naïve-Byes methods. Similarly SVM methods are applied and tested on many datasets such as Twitter (Gaydhan, Doma, Kendre, & Bhagwat, 2018), (Tungthamthiti, Shirai, & Mohd, 2017) IMDB (Tripathy, Agrawal, & Rath, 2016), (Verma & Yenter, 2017), and many text comments mixed forming a single huge dataset in which, MySpace, Twitter, YouTube, Facebook all the data are considered and one and on that model are tested (Chen, Mckeever, & Delany, 2018)

Different datasets should have different data processing approaches which are specified in many papers as researchers for this applied all the types of processing (Alberto, Lochter, & Almeida, 2015) and others didn't have any pre-processing because this doesn't always work well it can also change the data meaning to a large extent (Uryupina, Plank, Severyn, Rotondi, & Moschitti, 2014). Different papers have different approaches for terming the offensiveness and the abusiveness of the comments. The various perspective of classifying different types of abuses are classified as racist, sarcastic, abusive or (Stefan & Josh, 2010), (Derczynski, Ritter, Clark, & Bontcheva, 2013)

Machine learning models have shown high accuracy in detecting sarcastic comments on Twitter (Sood, ChurchillElizabeth, & Antin, 2012). Later, how to label a dataset and train a model was researched with some results and some backgrounds. (Prasar, Sanjana, Bhat, & Harish, 2017) CNN and RNN is applied on the dataset of IMDB movies in a very good way which will help in further implementation on comments (Verma & Yenter, 2017), giving the highest accuracy of 89.5% when changing different hyper parameters for the model

The IMDB movie dataset is used by many different papers using different pre-processing and different models applied on (Verma & Yenter, 2017) similar to which are used with different other datasets to experiment in different papers (Tripathy, Agrawal, & Rath, 2016). Similarly, In Twitter, the context is contained entirely in the tweets, and the conversations people had on YouTube, the video is an additional layer of context, and the comments are secondary to the platform experience.

Processing using N-gram, TF-IDF or using word embedding with machine learning models and deep learning gives 91-95% F1-score on different class labels. Detection twitter data for non-static glove plus word2vec CNN while being tested on various datasets (Zhang & Wallace, 2015).

A paper by (Chen, Mckeever, & Delany, 2017) has used the YouTube dataset which is an unlabelled dataset and used different approaches for labeling like crowdsourcing or manually labeling it. This paper used YouTube data with facebook witter and MySpace also (Davidson, Warmsley, Macy, & Macy, 2017). They used different methods for classification as SVM, CNN, RNN, Naive Bayes, logistic Regression, Decision Tree, Random Forest, etc on different processing and pre-processings.

Natural language processing tools were used and compared to find out which is the best approach for this paper (Derczynski, Ritter, Clark, & Bontcheva, 2013). Other Researchers also used many different NLP techniques for better analysis. Methods are applied to different types of text data also. NLP and Neural Network approaches have been split into character level classification and word-level classification for many papers.

The paper (Li, Liu, Zhang, Liu, & Liu, 2019) proposes a sentiment-feature-enhanced deep neural network (SDNN) to address the problem by integrating sentiment linguistic knowledge into a deep neural network via a sentiment attention mechanism. It uses movie Reviews and Stanford Sentiment Treebank dataset and applies all different deep learning approaches to test the accuracy for the data which shows that CTrees-LSTM gives the maximum accuracy for movie reviews. The text is processed by removing the stopwords and embedding the words with pre-trained GloVe. It also uses word2vec for twitter sentiment analysis choosing F1 score as the metric.

Pre-processing by (Chatzakou, et al., 2017) was seen using various approaches as a stopword, lemmatization, removing numbers and punctuations were done in this paper as on different dataset to explain how the text is processed before applying models.

The dataset used for this paper (Dadvar, Trieschnigg, & Jong, 2014) has worked with SVM finding the AUC of 0.57 as results for classification of different classes getting on the basis of other factors involved which are Number of Comments, Number of Subscribers, Membership Duration, Number of Uploads, Profanity in UserID, Age. With Multi-Criteria Evaluation System, machine learning approaches and a hybrid form of those models

(Deng & Yu, 2014) shows how the deep learning and their hybrid forms can make a complex model having all the qualities of the simple model combined as one.

Offensive/hate speech detection on social is a big problem of unbalanced data. One of the many approaches which can help solve this problem is resampling the training data and then training the model (Chawla, Japkowicz, & Kotcz, 2004), one of the most used and powerful resampling methods is SMOTE (Kegelmeyer, Hall, Bowyer, & Chawla, 2002)

2.4. Gaps in Research

Until now the research related to this says that there are many ways to detect the offensive language in comments or anywhere i.e. text mining, SVM, N-Gram, CNN and other many more Learning methods (Sood, ChurchillElizabeth, & Antin, 2012) (Chen, Mckeever, & Delany, 2018) but none have basically used both RNN and CNN to detect these on YouTube comment especially. Furthermore, there has been research on a wide scale across all social media platforms (Twitter, Facebook, news blogs, movie reviews, thumbs up thumbs down posts, etc) (Antonias 2015)(Anna & Mickel 2017).

The Dataset created and used as training data is not modelled properly so that there can be better efficiency or use different approaches as used multiple data sources to test the model but different platforms have different types of data and should use processing by considering all as one will have inferior results.

Apart from this NLPs Framework CNN is applied (Piyoras, Kiyoki & Masnizah 2017), (Anukars, Sanjana & Harish 2017) but then also some future work can be seen as the data availability is on a very large platform and can have various features to be classified. RNN and CNN can be used to detect this but haven't implemented clearly yet used for other datasets where the text is a bit simple as compared to that used in this paper.

To improve more efficiency, no measures are yet applied but can say that they have said in their future work. (Gaydhani, Doma, Kendre & Bhagwat, 2018), (Thomas 2002), (Xingyou, Weijie & Zhiyong 2016) Offensive means everything abusive or mean saying but no sarcastic (bad words but in subtle language) is also included in this area.

Papers have included the spams filter (Alberto, Lochter, & Almeida, 2015)or (Zhang, Yuan, Wang, & Zhang, 2017) have different emotions compared with different machine learning models. Machine learning is used in lots of the papers after processing the data but on that trying a deep learning model is a good option which is not seen.

The effectiveness of the methods used to do this is very low as can’t claim the results with a higher percentage. (Anagnostou, Mollas, & Tsoumakas, 2018), (Elvis 2018) or it is very high with least cleaning the data which can be said as not a very good approach.

CHAPTER 3 – DESIGN AND METHODOLOGY

3.1. Business Understanding

YouTube is a platform where people put up their thoughts on the most informal manner. This does not contain full language word. A new form of language called texting or messaging is mostly used by people which can be a challenging task. To solve the problem of offensive and non-offensive different approach is to be considered than normal text classification.

Mislabelling a non-offensive comment as offensive isn't a big deal. But mislabeling an offensive comment as not-offensive is because data is usually unbalanced for hate speeches and to understand it both classes are to be evaluated. To do that precision, recall and F1-score are evaluated for both the class (positive and negative). As recall is percentage of examples the classifier predicted for a given class of the total number of examples it should have predicted for that class. It increases when offensive commented are predicted more at the cost of low precision as it is the total number of correctly classified classes but due to imbalanced data, this situation is right for letting the model predict better.

3.2. Data Understanding

A deep understanding of the data is necessary for doing some research on it. Detailed description analysis of the YouTube comments dataset is done in this section.

3.2.1. Dataset Description

The dataset is of YouTube Comments (Dadvar, Trieschnigg, & Jong, 2014)[6] which was manually labeled by the researcher mentioned and hasn't been used in conjunction with Deep Neural Network models. It has *UserIndex, Comments, Number of Comments, Number of Subscribers, Membership Duration, Number of Uploads, Profanity in UserID, Age, Class* columns where Comments is the Text to be train data and Class is the Label specifying '0' as Negative and '1' as positive for offensive with '3464' rows. The *Comments* datatype is String and the *Class* datatype is an integer. Other columns are used by the mentioned researchers paper (Dadvar, Trieschnigg, & Jong, 2014) but for this research only Predictor variable as Comments and Target variable as Class are used to classify the models. The Dataset was checked for missing and duplicate value as

[6] Personally asked for this research

3.2.2. Target Variable Investigation

Class is considered as Target variable with 0 and 1 values as a binary class with 0 non-offensive comments and 1 as offensive comments. This dataset contains more non-offensive comments compared to offensive i.e. 3046 non-offensive as opposed to 418 offensive which shows that the dataset is highly unbalanced and has been represented by bar graph in Figure 3.1. This can give biased results in future so oversampling, under-sampling or stratified K-fold approach for sampling are used makes the data unbiased. In this paper, Oversampling is done using two approaches *RandomOverSampling and SMOTE Imblearn* functions.

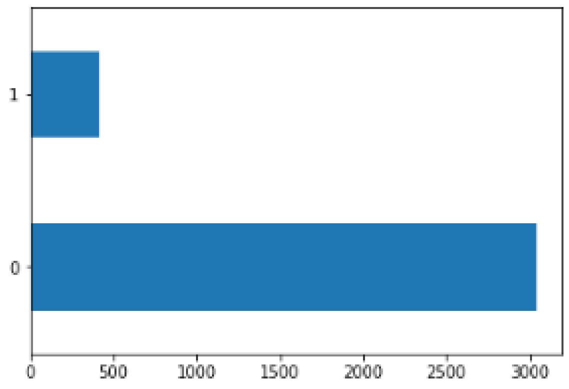


Figure 3.1: Data Visualisation of class attributes showing the variable count by a bar graph

3.3. Data Preparation (Pre-Processing)

The data is processed where the comments attributes are pre-processed using different natural Language Processing approaches, first stopwords, punctuations and numbers are removed to bring the data into a form that makes it predictable and analyzable for further steps. All the words are changed to lowercase to make consistency in the data. Then the Top 5 most common words are removed by their count to reduce noise. Similarly, the top 10 rarest words are removed from the document. Some duplicate letters are removed as “lolololol” gets cut short similarly both “boyyyyyy” and “boyyyy” are changed to “boyyy” which keeps the emotion of saying the word and reduces the noise also. “@patrick” changed to “@username” to reduce noise and consider all usernames as a pattern followed by lemmatization and stemming to bring the word to remove inflections and map a word to its root form then removing the columns that are not required for the analysis.

The target data which is Class is *Label Encoded* which refers to converting the labels into the numeric form so to convert it into the machine-readable form.

3.4. Data Splitting

After the preparation of the data, the data is split using 2 techniques first training and test data with 80% and 20% split respectively called the Hold-Out and the other is K-Fold where K value is 4 which means the data is divided into 4 samples and each sample is tested with other samples as training data and these steps are repeated 4 times. Hold-out is tested for all the models but K-Fold is tested for SVM, CNN and CNN-LSTM and interesting results are to be seen for different data splitting which is explained in the evaluation section for each experiment. The only hold-Out approach was tested on other models used for this research which are LSTM and LSTM-CNN due to lack of computational capacity and time to run.

3.5. Data Balancing

This is considered a major problem for the data used as the target variable is heavily imbalanced and there are much fewer positive class instances. To solve this problem multiple techniques are compared to see which one performs well for models and to increase the recall value for the positive class specifically. Techniques used for balancing the training data to have better learning for the model are *RandomOverSampling, SMOTE, RandomUnderSampling and Stratified K-Fold*. Oversampling and undersampling can lead to increase or decrease in the performance. All four techniques are applied for the models used in this research and tested. These data balancing techniques help the model to learn correctly as data of each class is passed with equal frequency which leads to no bias in learning the data. This is further tested on the validation data to check if the model has learned correctly or not or to what extent.

3.6. Feature Extraction

Two types of features are extracted for this research. TF-IDF (N-gram) for SVM models and Word embedding for Neural Networks. For NN Feature extraction is done after tokenizing and padding the sentences to a maximum length of sentence in a document to maintain consistency and don’t have to worry about tensor dimension for models. This process is for Deep learning models where embedding matrix is formed as explained below.

3.6.1. N-Gram and TF-IDF

N-gram is an approach for feature extraction which is performed by combining N-words into a single set and using each such set as a feature in the dataset. For example, a bi-gram approach looks at each pair of words, a tri-gram approach looks at each triplet, etc.

Bi-gram approach makes sets using the first word and the consecutive word as one set and then the second word and it's consecutive and so on, this gives the leverage for the machine to process a word twice (in its context) and understand it better. It is then passed through TF-IDF, vectorizing the words and considering them with their frequencies of appearance in a document. This is used for text classification by SVM model.

3.6.2. Word Embedding

Word embedding is a better and booming approach for feature extraction and works best for Deep learning models, so GloVe, a pre-trained Stanford text dataset is used to make the embedding matrix which is passed to the Keras models of deep neural network. This matches most of the words from the dataset. For missing words, random embedding with the same mean and standard deviation as the GloVe embedding is used to include each word in matrix and none of it is changes to 0 as in not matched element.

3.7. Modeling

After the data understanding, pre-processing, balancing, splitting and their feature extracted models are fitted on the training data. A baseline model is created to be compared with other models and other experiments tested in this part considering that model as the state-of-art for this research.

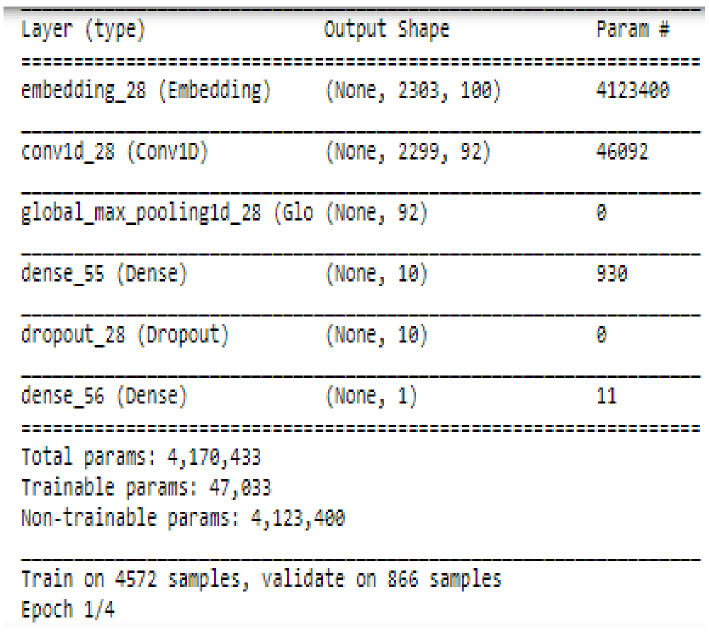
3.7.1. Experiment 1(SVM)

This is considered as the baseline model for experiments. The training data is passed through the SVM model to fit the model on the training data with the TF-IDF (bi-gram) vector and predicting the result on the test data. Trying and testing different parameters for it to achieve a better accuracy score giving the classification report and confusion matrix explaining its results in more details.

3.7.2. Experiment 2 (CNN)

This model is a sequential model as layers are added one by one to it and the model performs by the sequence of those layers. The embedding layer uses the embedding matrix formed from pre-trained Glove data in it with the training input data and this layer is considered as the input layer. Next layer is the Convolutional 1 dimension layer with 92 filters where the kernel size is 5 and activation function *'tanh'* after that global max pooling layer is added to it which is ordinary *max pooling* layer with pool size equals to the size of the input. Then a fully connected *dense layer* is added where it will have 10 neurons with *kernel_regularizer* which is used to decay the weight, this allows to add penalty for weight size to *loss function* which helps to reduces the overfitting then dropout layer is added to model with a value of '0.5' this is a technique where randomly selected neurons are ignored during training. This helps the model if it is starting to overfit.

Then finally a sigmoid fully connected layer is added to it which gives the binary output. These layer and hyperparameters are manually tuned depending on achieving better results for the model training and predicting. Figure 3.2 below shows a summary of the model which is then compiled with optimizer *adam*, *binary crossentropy* loss and metrics having accuracy and mean a predicted class for true test and predicted test.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_28 (Embedding)     (None, 2303, 100)         4123400
_________________________________________________________________
conv1d_28 (Conv1D)           (None, 2299, 92)          46092
_________________________________________________________________
global_max_pooling1d_28 (Glo (None, 92)                0
_________________________________________________________________
dense_55 (Dense)             (None, 10)                930
_________________________________________________________________
dropout_28 (Dropout)         (None, 10)                0
_________________________________________________________________
dense_56 (Dense)             (None, 1)                 11
=================================================================
Total params: 4,170,433
Trainable params: 47,033
Non-trainable params: 4,123,400
_________________________________________________________________
Train on 4572 samples, validate on 866 samples
Epoch 1/4
```

Figure 3.2: Model summary of Convolutional Neural Network for this research

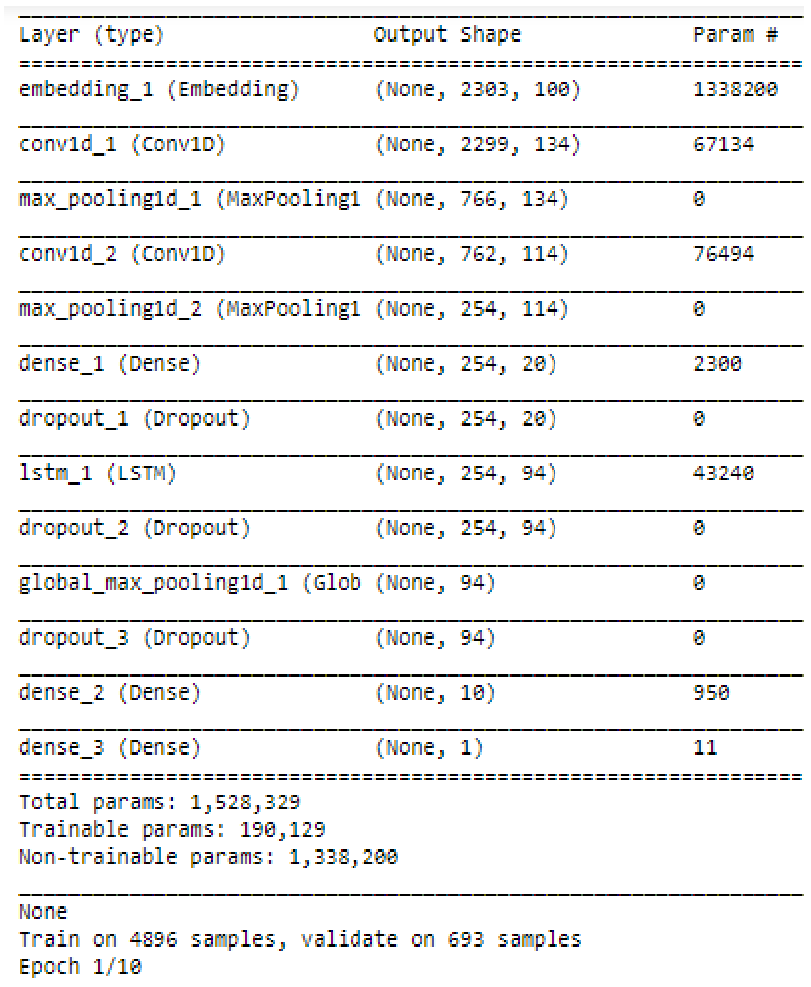
To understand the right working of this model classification report and confusion matrix is being viewed showing results for both the classes how well the model is learning and predicting for classes and learning graphs explaining if there any noticeable overfitting or underfitting.

3.7.3. Experiment 3 (CNN-LSTM)

CNN-LSTM a mixed form of deep learning algorithms is the third experiment for this research. It is a model in which after the Convolutional layer is added, the next layer will be of long short term memory. For this research, manual tuning is done for the model to achieve the best results and fit the training data correctly removing any noise in the model.

It is a sequential model for which the input layer is an embedding layer through which tokenized training data is passed. Adding to it is the Conv1D layer which has 134 filters with kernel size of 5 and the activation function *tanh*. Next is the *max-pooling* layer with a pool size of 3. These 2 layers are added again with the same parameters to have 2 layers for CNN and to complicate the model. After this a dense layer (fully connected) with 20 neurons whose output are passed through LSTM layers. Before adding an LSTM layer dropout and a *regularizer* are added to prevent overfitting.

For this dataset 1 layer of LSTM is added with the parameters 94 output and a *tanh* activation function. This output is passed through *global max pooling* layer but both the layers are added with dropout parameter which is then passed to a fully connected 10 neurons dense layer with *tanh* activation function. This is then passed through 1 *dense* layer with sigmoid function which is used to give a binary output. These parameters were tuned, layers removed and added till a good fitting model was achieved. Figure 3.3 below shows how these layers were added what shape and values were passed through each layer as a model summary.



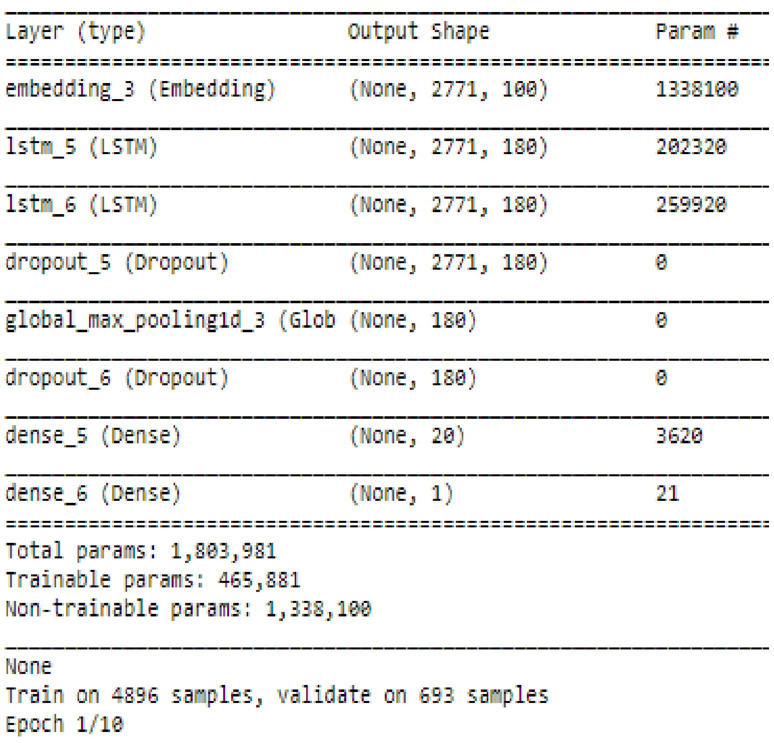
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 2303, 100)         1338200
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 2299, 134)         67134
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 766, 134)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 762, 114)          76494
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 254, 114)          0
_________________________________________________________________
dense_1 (Dense)              (None, 254, 20)           2300
_________________________________________________________________
dropout_1 (Dropout)          (None, 254, 20)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 254, 94)           43240
_________________________________________________________________
dropout_2 (Dropout)          (None, 254, 94)           0
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 94)                0
_________________________________________________________________
dropout_3 (Dropout)          (None, 94)                0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                950
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 11
=================================================================
Total params: 1,528,329
Trainable params: 190,129
Non-trainable params: 1,338,200
_________________________________________________________________
None
Train on 4896 samples, validate on 693 samples
Epoch 1/10
```

Figure 3.3: Model summary of Convolutional Neural Network - Long Short Term Memory for this research

3.7.4. Experiment 4 (LSTM)

LSTM model is a part of RNN and for this research, a complex form of LSTM architecture is required which found out when tuning was being done for the best fit. Here, the embedding layer is the input for a sequential model with 100 dimensions of *embedding* in an embedding matrix. Two LSTM layers are added consecutively having 180 outputs followed by a dropout layer to prevent from overfitting. Next, a *global max-pooling* layer is added which max pools all of the input from the previous layer after which again a dropout is added as per the model required for this problem. These outputs are fully connected by 20 neurons in a dense layer with the *tanh* activation function this is finally connected giving 1 *dense* output stating the class with *sigmoid* activation function.

In between the core layer dropout is added to prevent the model overfitting. Similarly, *kernel_regularizer* is added to decay the weights. The model architecture is made using the reference of CNN model developed earlier to which layers were added and removed. Below is the model summary of LSTM in Figure 3.4



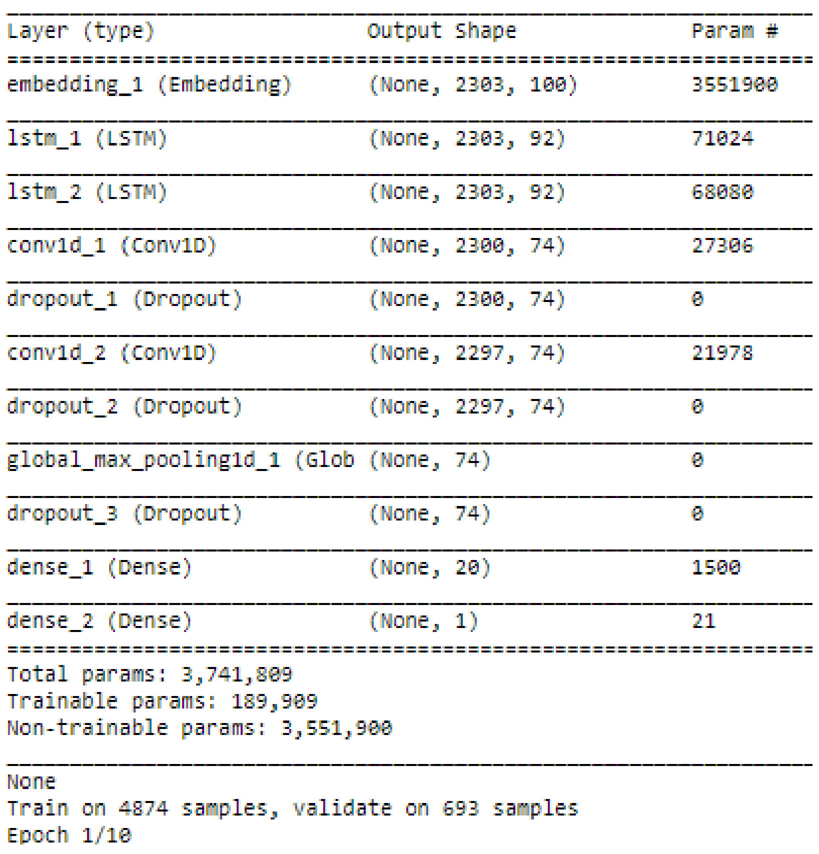
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_3 (Embedding)      (None, 2771, 100)         1338100   
_________________________________________________________________
lstm_5 (LSTM)                (None, 2771, 180)         202320    
_________________________________________________________________
lstm_6 (LSTM)                (None, 2771, 180)         259920    
_________________________________________________________________
dropout_5 (Dropout)          (None, 2771, 180)         0         
_________________________________________________________________
global_max_pooling1d_3 (Glob (None, 180)               0         
_________________________________________________________________
dropout_6 (Dropout)          (None, 180)               0         
_________________________________________________________________
dense_5 (Dense)              (None, 20)                3620      
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 21        
=================================================================
Total params: 1,803,981
Trainable params: 465,881
Non-trainable params: 1,338,100
_________________________________________________________________
None
Train on 4896 samples, validate on 693 samples
Epoch 1/10
```

Figure 3.4: Model summary of Long Short Term Memory for this research

3.7.5. Experiment 5 (LSTM-CNN)

LSTM-CNN is a hybrid approach of deep learning in which CNN layers are added to the LSTM layers which are just the opposite but very similar to CNN-LSTM. For this research it was found out that although previous experiments report that complex LSTM models work better, a simple *conv1d* layer is better for this model. So keeping that in mind LSTM-CNN was developed which had *embedding* layer as the input layer having training array as the input which is compared to the embedding matrix made. Along with which 2 LSTM layers are added with *tanh* activation function having 102 and 92 neurons respectively as output which will be connected from the embedding layer. Next layer added is a Convolutional layer where 74 filters are used with the kernel size of 4 after which to remove overfitting a *dropout* layer is added. This output is passed through *global max-pooling* layer and then another dropout is added. 20 neurons dense layer takes this as input with the *regularizer* to check weights which will finally give an input of 1 neuron having sigmoid activation function.

This model is compiled with *adam* optimizer and *binary crossentropy* loss with accuracy metric. The model is fit on the training data obtained from different sampling techniques with the *batch size* of 110 and 4 *epochs*. Model summary for this model can be seen in Figure 3.5.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 2303, 100)         3551900
_________________________________________________________________
lstm_1 (LSTM)                (None, 2303, 92)          71024
_________________________________________________________________
lstm_2 (LSTM)                (None, 2303, 92)          68080
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 2300, 74)          27306
_________________________________________________________________
dropout_1 (Dropout)          (None, 2300, 74)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 2297, 74)          21978
_________________________________________________________________
dropout_2 (Dropout)          (None, 2297, 74)          0
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 74)                0
_________________________________________________________________
dropout_3 (Dropout)          (None, 74)                0
_________________________________________________________________
dense_1 (Dense)              (None, 20)                1500
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 21
=================================================================
Total params: 3,741,809
Trainable params: 189,909
Non-trainable params: 3,551,900

_________________________________________________________________
None
Train on 4874 samples, validate on 693 samples
Epoch 1/10
```

Figure 3.5: Model summary of Long Short Term Memory - Convolutional Neural Network for this research

3.8. Performance Evaluation

Performance evaluation can be done by considering different measures but totally depending on one factor is not the right way for understanding how better the model works. Other factors should be considered equally and then it should be judged which model is working better. Let’s say if a model is having 95% accuracy but the all the classes are not showing the balanced prediction for precision, recall and F1-score. A model could get 95% accuracy by always predicting the same label on an imbalanced dataset, so we want to make sure that the model can recognize both positive and negative instances. This tells that for this research **Recall** is considered as the main performance evaluation metric because recall refers to the percentage of total relevant results correctly classified by the algorithms. Accuracy is defined as the ratio of correctly predicted examples by the total predicted class. Falsely predicted does not make a big difference for this and where each class result is not necessary, it gives results for the overall model. For this case predicting if the comment is offensive it is better to have high recall as the model developed do not want to lose the classes and predict wrong. It comes with the cost of decrease in the precision which is acceptable here.

This is true for algorithms only if both the classes are performing well for other metrics also which are precision and F1-score. To understand the result more and make sure models are not giving biased results a confusion matrix is evaluated along with the classification report which will tell the precision, recall, F1-score and other factors for both the target values and each of them is giving results correctly or not.

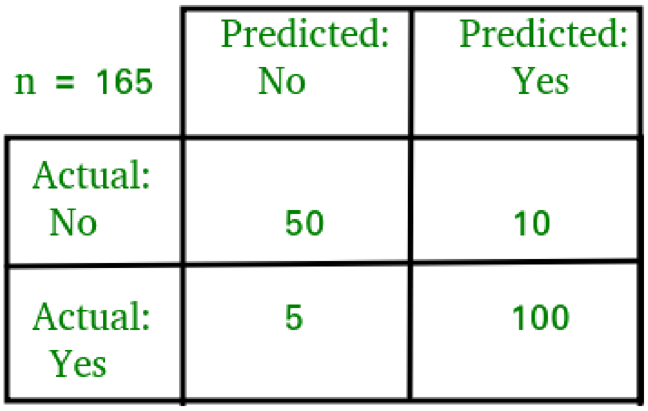
To statistically prove that the performance evaluation done is correct or not a T-Test is performed which compares the models on the data statistically (Kyun, 2015). A T-test is a type of statistical test that is used to compare the means of two groups. It is divided into two types independent T-Test and paired T-Test i.e. when groups under comparison are independent and when groups under comparison are dependent on each other. There are two statistical inferences parametric and non-parametric, in which one defines the probability distribution of probability variables and makes inferences about the parameters of the distribution and vice versa respectively. For this research independent T-test on nonparametric inferences is tested. There are many ways of doing the T-test depending on the parameters or the approach but here Friedman Chi-Square test is done as it is used when more than 2 methods are compared. This is the nonparametric version of the repeated measures ANOVA or repeated measures analysis of variance test.

Generally, each test calculates a statistic that is interpreted with some statistics background and a deeper knowledge of the statistical test. Tests also return a p-value that is used to interpret the result of test. The p-value is the probability of observing the two or more than two data samples given the base assumption (null hypothesis) that the two samples were drawn from a population with the same distribution. It is compared to the alpha value of 0.05, if less, the null hypothesis is rejected. If not, it fails to reject the null hypothesis. 4 samples are created for each model tested which are SVM, CNN and CNN-LSTM.

CHAPTER 4 – RESULTS, EVALUATION AND DISCUSSION

4.1. Results

This part will cover the final results achieved for different experiments and a description of the performance metrics shown by the classification report which gives the report on a per-class basis. This gives a deeper intuition of the classifier behaviour over accuracy which can mask functional weaknesses in one class of a multiclass problem. The metrics are defined on the basis of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). True Positive is when the actual class is positive and the model predicts positive and false positive is when the actual class is negative but the estimated class is positive. In this research 0 is non-offensive and 1 is offensive and TN represents offensive comments which have less data as compared to non-offensive. Using these terminologies Precision, Recall, Support and F1-score results are explained. The report also includes macro average (averaging the un-weighted mean per label) and weighted average (averaging the support-weighted mean per label). A confusion matrix is a matrix which shows the performance of a classification model on test data for true values as shown below in Figure 4.1.



n = 165	Predicted: No	Predicted: Yes
Actual: No	50	10
Actual: Yes	5	100

Figure 4.1: Confusion Matrix[7]

4.1.1. Experiment 1 Results (SVM)

Table 4.1 and Table 4.2 shows baseline (Machine learning) results for Support Vector Machine based on different balancing techniques for data. Here there are 2rowsshowingthe

[7] https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

offensive class results and non-offensive class results followed with different sampling techniques whose performance metrics are stated as Precision, Recall and F1-score values. As Recall is considered as evaluation score for this research it is seen that for Hold-out random oversampling the model is having 100% for non-offensive but 0% for non-offensive and for precision and F1- score the result is 0% for offensive class. Similar results are seen when stratified cross-validation is used. For SMOTE and random oversampling with 4 folds, the model is poor learning with 54, 04 and 07% precision-recall and F1-score for offensive class, unlike non-offensive class. But for Random under-sampling with 4 fold SVM model is giving some values for both the classes as the scores are 92, 73 and 82 for non-offensive and 23, 53 and 32 for offensive. As it can be seen from the accuracy column that undersampling is having less accuracy of 67.58% then also both the classes are learning well. So with Recall of 53% under-sampling approach is giving better results for SVM model.

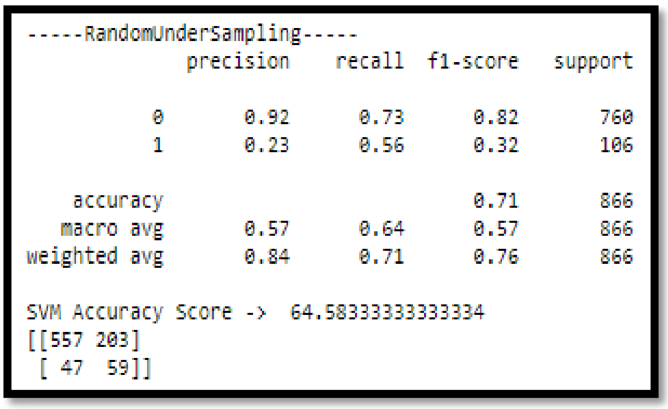
From another table Table 4.2 which is stating the TP, FP, TN and FN as in the confusion matrix values to make the results more clear. It is seen that FP are less but in the same way TN are much less for holdout, oversampling, smote and stratify which are not considered correct result for a good prediction model. However, under-sampling is having values 557, 203, 47 and 59 respectively where the error values are less than the true values which confirm the result stated more rigid. Figure 4.2 below in shows the classification report and confusion matrix for under-sampling from which results in the table are recorded.

	SVM		Precision	**Recall**	F1-score
0/non-offensive	Hold-out Random oversampling		86	**100**	93
	Random oversampling		88	**100**	94
	SMOTE		88	**99**	93
	Random undersampling		92	**73**	82
	Stratified K-Fold		88	**100**	94
1/offensive	Hold-out Random over sampling		00	**00**	00
	Random oversampling		54	**04**	07
	SMOTE		40	**04**	07
	Random undersampling		23	**53**	32
	Stratified K-Fold		00	**00**	00

Table 4.1: Results for SVM model for different sampling and classes

	TP	**FP**	**FN**	**TN**	**Accuracy**
Hold-out Random over sampling	598	0	35	0	86.29
Random over sampling	757	3	102	4	99.38
SMOTE	754	6	102	4	95.88
Random under sampling	557	203	47	59	67.58
Stratified K-Fold	761	0	104	0	87.91

Table 4.2: SVM confusion matrix score and accuracy for different sampling techniques



```
-----RandomUnderSampling-----
              precision    recall  f1-score   support

           0       0.92      0.73      0.82       760
           1       0.23      0.56      0.32       106

    accuracy                           0.71       866
   macro avg       0.57      0.64      0.57       866
weighted avg       0.84      0.71      0.76       866

SVM Accuracy Score ->  64.58333333333334
[[557 203]
 [ 47  59]]
```

Figure 4.2: Classification Report and Confusion matrix obtained from implementation for SVM using under-sampling technique

4.1.2. Experiment 2 Results (CNN)

Table 4.3 and Table 4.4 shows the results for the CNN model. Similar to Figure 4.3 the classification report and confusion matrix is developed while the implementation for different sampling techniques which is then converted into a table given below. As discussed earlier both the classes should be predicting well to have a good model it can be seen that stratified cross-validation has the highest accuracy of 87.97 but the model is biased towards non-offensive class and prediction 0% for offensive. For hold-out random oversampling 26, 23 and 24 % precision, recall and F1-score can be seen more than SVM for offensive class. With K-fold for oversampling and undersampling, recall is 91 and 92 for non-offensive and 23 and 29 respectively for offensive class which can be explained more using the confusion matrix that these approaches for SVM are letting model learn better or not. FP (Type I error are less) but FN (Type II error) are more as compared to TN for oversampling and undersampling.

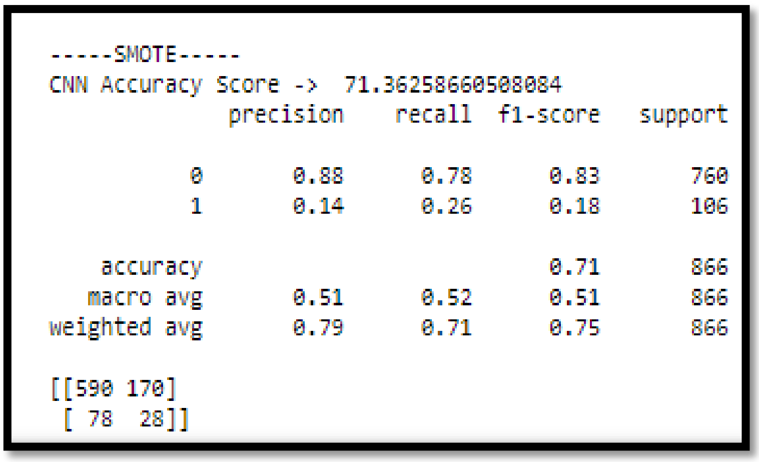
SMOTE, a type of oversampling which uses synthetic minority classes to balance the training data has 88, 78, 83 performance metrics for non-offensive and 14, 26, 18 for offensive class giving good metric and stating that SMOTE learns and predicts better for CNN model. This can be seen from the confusion matrix also as TP is 590 with type I error of 170 and TN is 28 greater than other approaches with lesser FN.

CNN		Precision	**Recall**	F1-score
0/non-offensive	Hold-out Random oversampling	87	**92**	89
	Random oversampling	89	**91**	90
	SMOTE	88	**78**	83
	Random undersampling	89	**92**	90
	Stratified K-Fold	88	**100**	94
1/offensive	Hold-out Random over sampling	18	**12**	14
	Random oversampling	26	**23**	24
	SMOTE	14	**26**	18
	Random undersampling	24	**29**	21
	Stratified K-Fold	00	**00**	00

Table 4.3: Results for CNN model for different sampling and classes

	TP	**FP**	**TN**	**FN**	**Accuracy**
Hold-out Random over sampling	549	49	11	84	80.80
Random over sampling	690	70	24	82	82.44
SMOTE	590	170	28	78	71.36
Random under sampling	697	63	20	86	82.79
Stratified K-Fold	761	0	0	104	87.97

Table 4.4: CNN confusion matrix score and accuracy for different sampling techniques



```
-----SMOTE-----
CNN Accuracy Score ->  71.36258660508084
              precision    recall  f1-score   support

           0       0.88      0.78      0.83       760
           1       0.14      0.26      0.18       106

    accuracy                           0.71       866
   macro avg       0.51      0.52      0.51       866
weighted avg       0.79      0.71      0.75       866

[[590 170]
 [ 78  28]]
```

Figure 4.3: Classification Report and Confusion matrix obtained from implementation for CNN using SMOTE sampling technique

4.1.3. Experiment 3 Results (CNN-LSTM)

CNN-LSTM model is a hybrid of CNN and LSTM. For this model it is observed from Table 4.5 that Hold-out with random oversampling is giving a good accuracy of 33% for offensive and for non-offensive class it is giving 76% recall which is far better than k-fold random oversampling and under-sampling as precision, recall and f1-score for them for non-offensive class is 89, 97 and 93 and 93, 26 and 40 but for offensive class it is 33, 10 and 16 with 14, 87 and 24. For this model stratified cross-validation is giving results with 24% recall for offensive class and 94 % for the non-offensive class which shows that the model is comparatively good than previous two as SVM and CNN were giving all the values as FN.

From Table 4.6, as for the accuracy score which will not matter other predicting values don't give right values but as can see that random oversampling and stratified cv is giving better results than other i.e. around 86% but they cannot be considered correct due to other scores it is giving.

From the TP, FP, FN and TN scores it has been seen that TP are more as it should be than FP and TN is less but predicting well for the class except for under-sampling which can be due to less amount of data. So from all the above observations smote is giving good recall for this model whose results were observed while implementation from classification report and confusion matrix which can be seen in Figure 4.4

CNN-LSTM		Precision	**Recall**	F1-score
0/non-offensive	Hold-out Random oversampling	88	**76**	81
	Random oversampling	89	**97**	93
	SMOTE	90	**76**	82
	Random undersampling	93	**26**	40
	Stratified K-Fold	90	**94**	92
1/offensive	Hold-out Random over sampling	18	**33**	23
	Random oversampling	33	**10**	16
	SMOTE	18	**37**	24
	Random undersampling	14	**87**	24
	Stratified K-Fold	37	**24**	29

Table 4.5: Results for CNN-LSTM model for different sampling and classes

	TP	FP	TN	FN	Accuracy
Hold-out Random over sampling	455	143	31	64	70.12
Random over sampling	738	22	11	95	86.48
SMOTE	577	183	39	67	71.37
Random under sampling	195	565	92	14	33.14
Stratified K-Fold	719	42	25	79	86.01

Table 4.6: CNN-LSTM confusion matrix score and accuracy for different sampling techniques

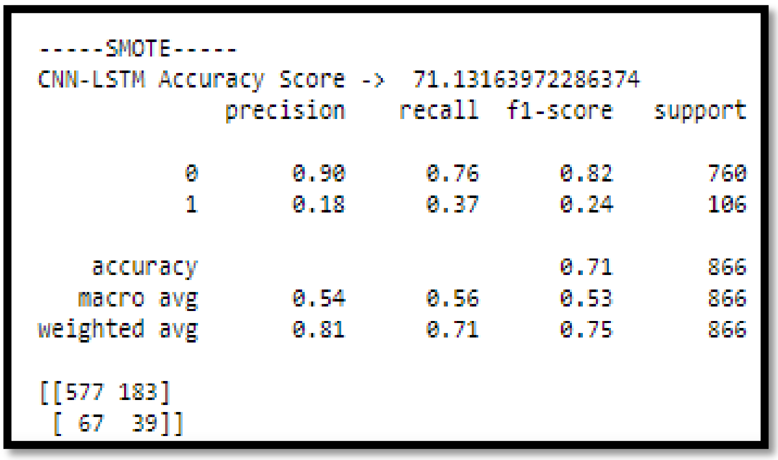
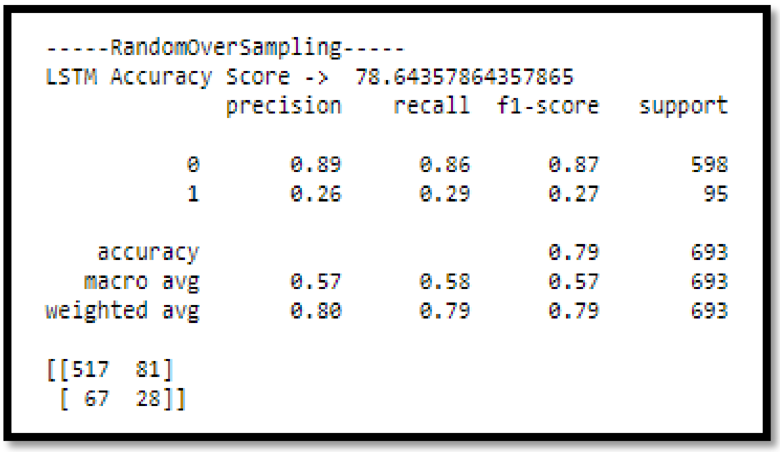

```
-----SMOTE-----
CNN-LSTM Accuracy Score ->  71.13163972286374
              precision    recall  f1-score   support

           0       0.90      0.76      0.82       760
           1       0.18      0.37      0.24       106

    accuracy                           0.71       866
   macro avg       0.54      0.56      0.53       866
weighted avg       0.81      0.71      0.75       866

[[577 183]
 [ 67  39]]
```

Figure 4.4: Classification Report and Confusion matrix obtained from implementation for CNN-LSTM using SMOTE sampling technique

4.1.4. Experiment 4 Results (LSTM)

Figure 4.5, below shows the classification report for the LSTM model where the accuracy is 78.64% and both the classes are also learning better than the previous experiments results for holdout as the precision, recall and f1-score for both the classes have 89 and 26%, 86 and 29% and 87 and 27% prediction for non-offensive and offensive class respectively. From the report, it can be seen that 598 and 95 are the support (instances) for the model. And confusion matrix shows the True (correct) classes are also predicted well. For this model, K-fold is not tested as it requires more computational time than other models which is considered for future scope.

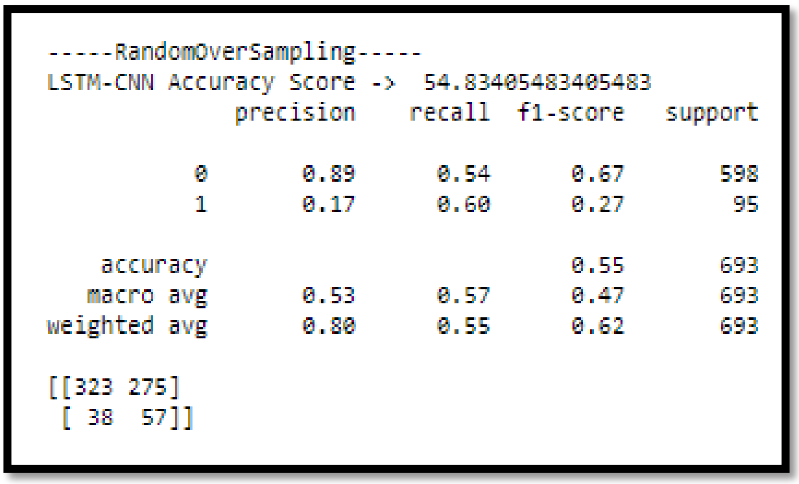


```
-----RandomOverSampling-----
LSTM Accuracy Score ->  78.64357864357865
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       598
           1       0.26      0.29      0.27        95

    accuracy                           0.79       693
   macro avg       0.57      0.58      0.57       693
weighted avg       0.80      0.79      0.79       693

[[517  81]
 [ 67  28]]
```

Figure 4.5: Classification Report and Confusion matrix obtained from implementation for LSTM using Random Oversampling technique

4.1.5. Experiment 5 Results (LSTM-CNN)

This experiment shows the results obtained for the LSTM and CNN model, a mixed form of LSTM and CNN as the name suggests. Here accuracy is 54.83% with 598 and 95 support as from the Figure 4.6 below it is observed that non-offensive classes learning well with 54% recall and offensive class are also learning well with 60% recall. As discussed earlier accuracy only does not contribute in determining model goodness but other factors are also included so all the performance metrics are taken into consideration and clearly it can be seen that both classes are learning equally and very well while results are also considered and based on confusion matrix and seen that TP and TN have better prediction with 323 and 57 than FP and FN with 275 and 38. As LSTM model has longer computational time, for this research LSTM-CNN was tested only for Hold-out approach. And cross-validation can be considered as future scope.



```
-----RandomOverSampling-----
LSTM-CNN Accuracy Score ->  54.83405483405483
              precision    recall  f1-score   support

           0       0.89      0.54      0.67       598
           1       0.17      0.60      0.27        95

    accuracy                           0.55       693
   macro avg       0.53      0.57      0.47       693
weighted avg       0.80      0.55      0.62       693

[[323 275]
 [ 38  57]]
```

Figure 4.6: Classification Report and Confusion matrix obtained from implementation for LSTM-CNN using Random Over Sampling technique

4.2. Evaluation

The dataset is cleaned and processed by using Natural Language Processing. The sentences (comments) are processed and prepared for classification by converting them into lowercase, removing the stopwords, punctuations, numbers, top-most common and the rarest by count as well as duplicated as explained in the Design and methodology. After processing two types of features were extracted. TF-IDF with Bi-gram for SVM model and Word Embedding using pre-trained Glove for Neural Network. The further explanation includes how the models were implemented. The models are tested with different sampling techniques for balancing the data on 4 fold cross-validation and before that it is also tested on Hold Out technique too with random oversampling the training data.

4.2.1. Experiment 1 (SVM)

After converting the words in the document into a vector with TF-IDF, *Hold-Out* was used to split the data into training and testing sets where the training data was oversampled by random oversampling function and passed through the SVM linear model. The results were predicted with test data when there were good results for accuracy but the precision, recall, and f1- score was very less for the positive class (offensive which is considered as True Negative). Seeing these results model passed through the *K-fold* with number of split as 4, this let us train the 4 models using 4 different samples which eventually make the model learn more on same data.

This was applied for four different sampling techniques Imblearn packages' Random Over Sampling, SMOTE (Synthetic Minority Over-sampling Technique), Random Under Sampling and Stratified K-Fold and can see from Table 4.1 great results of 99.3% was achieved for oversampling but for the true negative (offensive class) the model is not learning very well and the predicted class cannot match the true class correctly. There is more False Negative than True Negatives. Precision is 100% but recalls only 4% also f1-score 9% which means the model is biased. Similarly for SMOTE and Stratified k-fold using cross-validation which is giving the accuracy of 96% and 87% but the model can't learn the class correctly like oversampling and having much lower performance metrics for the positive class. Unlike these techniques, under-sampling is performing not that great with 67% if only accuracy is considered but the recall is greater for positive class i.e. offensive class is having good results while predicting true class.TP and TN also explain how under-sampling is working well for this model. Every approach for this model had changes in the model parameters as *class_weights* were added to make it balanced and *C (penalty)* parameter values were increased and decreased to improve the working of model. So for SVM under-sampling is showing the best result for YouTube comments dataset as under-sampling reduces/removes the instances for the majority class randomly to make it equal to the minority class which is letting the model to get trained better than others for this experiment.

4.2.2. Experiment 2 (CNN)

The second experiment includes the first implemented neural network model used for dataset i.e. CNN. This experiment has layers in the model architecture with embedding as the first input followed by 1 dimension Convolutional layer with max-pooling layer connected with two consecutive dense layers the last giving the binary class output.

The model is tested and trained firstly by Hold-out split of 80% - 20% data for training and testing respectively but as baseline model after oversampling it was seen that the CNN is rigged for this test as it was giving unstable results as for the negative class as the precision, recall and f1-score value was very low. It can be seen from the classification report to overcome this problem 4 fold cross-validation was applied and its results were checked using four different sampling techniques. Random oversampling as discussed in the results section, Table 4.3 and Table 4.4 states that in spite of having good accuracy than other techniques it cannot be considered as better than others because it is having much less precision, recall and f1-score for offensive classes with fewer instances. Here can be seen that the results are better than the machine learning model for all the techniques, this is due to the fact that Deep learning models have better learning approach and are complex models which means it learns the data in difficult way. For SMOTE and under-sampling it has been seen that results are good for both the classes but as the baseline model, CNN is also not learning when stratified 4 fold cross-validation is tested as it is giving the performance metrics for offensive class 0% correct classes at all.

From observing all the techniques for the CNN it can be seen that SMOTE and undersampling is giving better results for data where SMOTE is having better accuracy of 71.36% than under-sampling so oversampling the deep neural network (CNN) and Recall also is 32% greater than all the other techniques which is correctly predicting for both the classes with synthetic minority classes is learning better for offensive and non-offensive class.

From these results it can also be concluded that the CNN model is showing better performance than the machine learning model which is considered as the state-of-art for this research as instead of SVM having higher accuracy the performance for classes it is not that good but Convolutional Neural Network can show better results for the other performance metrics and recall especially for both the classes which are considered as very important to show the results and not only and discretely based of accuracy measure.

4.2.3. Experiment 3 (CNN-LSTM)

This is a mixed Deep Neural Network. In this model, CNN is combined with an LSTM to make a new and better neural network with the GloVe-embedded text layer as its input. This is followed by two 1-dimensional Convolutional layers and added with one LSTM layers having dropout layers in between. A Max pooling layer is further added with the fully connected layers giving binary output. These layer parameters are inspired by the previous models made and changes made for them in accordance to receive better results.

As previous experiments and as shown in the results section this model in Table 4.5 and Table 4.6 was tried with two approaches Hold-out and k-fold for which the results achieved are interesting as it is discussed that SVM worked better for under-sampling and for CNN smote technique was better considering Recall as performance evaluation for this research and it can be seen that for CNN-LSTM also SMOTE technique is best performing when k-fold is used with the recall of 37% which is greater than 10, 33 and 24 for positive class compared to other techniques. As for under-sampling recall is 87% but also recall for negative class it much less with 27%. This can be explained as under-sampling removes the majority classes to make both the classes' equal leads to data amount decrease which is not considered very helpful for deep learning model to learn as they require more amount of data to learn and predict better. There should be a balance between the learning of both of the classes as only then can it be considered as a better performing approach or technique for the model used. Confusion matrices also play a very important role for the model in explaining how the model is predicting and for this model it can be seen that while using SMOTE techniques which has better recall results have better correctly predicted positive and negative class. So it can be concluded the Smote technique works best for the model with the accuracy of 37%.

4.2.4. Experiment 3 (LSTM)

For this dataset, LSTM model requires a complex architecture which has the embedding layer as the input with two LSTM layers with dropouts after each layer on it. A max-pooling layer is added with another dropout and then two consecutive fully connected layers are added to have a binary output at the end. This model is trained and tested on the training and testing data to get some results for evaluation. Hyper parameters are manually tuned finding out the best result that can be achieved from them.

This experiment is carried out for Hold-out approach only as it requires more computational capacity to run the K-fold approach but for hold-out also it can be seen that results achieved are 78% accuracy with 86% recall for non-offensive and 29 % for offensive class which tells that the model is running well when oversampling techniques is used. LSTM model uses the back-propagation to learn better and match patterns. This is a good approach to test the model working as when the model is made it will go each and every step back to test for errors and get to the root of it to correct it which will help the model to learn better. So for this experiment, it can be said that the on Hold-out it is performing well and performance metrics can be improved when tested with folds.

4.2.5. Experiment 5 (LSTM-CNN)

This is a mixed Deep Neural Network. In this model LSTM is combined with CNN to examine another similar approach as the previous neural network with embedding layer as the input but followed by two LSTM layers and added with one 1D Convolutional layer having dropout layers in between to reduce the overfitting to the minimum further adding max pooling layer with the two fully connected layers giving binary output in the last fully connected layer. Adding layers is inspired by the previous models and changes made for them are in accordance to receive better results.

LSTM-CNN approach can be seen with great results as the model is learning very well for hold-out oversampling approach as compared to other experiments. This model has less accuracy of 54.83 % but the recall for both the classes is predicted very well with the accuracy of 54 and 60% for non-offensive and offensive class. This proves that as the model is becoming more and more difficult to run that are giving better results for research dataset. It really matters which approach is used to sample the training data but as for this model oversampling is only tested giving better results it can be considered a very good future approach for this model to test other techniques and then compare the results.

4.3. Discussion

From the results and evaluation discussed above performance evaluation are compressed in a table and shown below in Table 4.7 which give the result that hybrid model are working better for this research topic followed by rejecting the null hypothesis i.e. mixed deep learning model do not perform statistically significantly better than other simple deep neural network and machine learning model on imbalance YouTube comments dataset for predicting offensive class.

There are many reasons for saying this which will be discussed further in this section:

Machine learning and deep learning are very different from each other. Deep Learning models have hidden layers which help the model to work better for some problem because they are complicated models and they can be changed on the basis of problem needs and fitting requirements for the data. How the model is fitting with training data, is there any over-fitting and under-fitting are observed by the learning curves which tells how stable the model is for the data. But in the case of machine learning, it does not provide many options making the model as simple as possible for the data to fit. This can be considered a good approach or bad approach depending on the problem one is considering. For this research it has been observed that complex deep learning models are performing better than machine learning as CNN, CNN-LSTM and LSTM-CNN are learning and predicting well for both the classes than Support Vector Machine.

Sampling approaches are used for this research as the dataset is imbalanced dataset and this played a very important role in defining and categorizing the final model performance. As seen earlier all the four sampling techniques used for this thesis are getting different results for different models.

The main observation that is seen is SVM performing well with random under-sampling and deep neural networks are performing well for SMOTE. SMOTE is an approach where the minority classes are replaced by synthetic samples which do not randomly duplicate the instances but creates new synthetic ones. This approach can lead to both increasing or decreasing the performance as for SVM it can be seen that random under-sampling which is reducing the majority classes is letting the model learn better and give better predictions. Unlike SVM, deep neural networks are working better with SMOTE techniques. Deep neural networks are complex models which require complex and more data to learn well. SMOTE makes the data more complex by creating synthetic instances allowing the model to learn well as can be seen from Figure 4.7 where orange is the minority class and blue is the majority class and the green are the samples which are added to the data for the minority class which blends in with the training data.

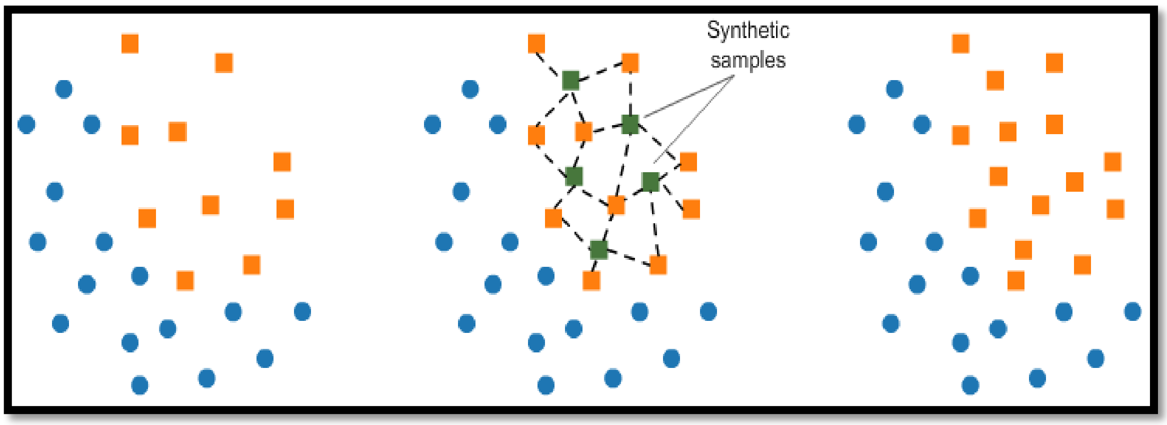
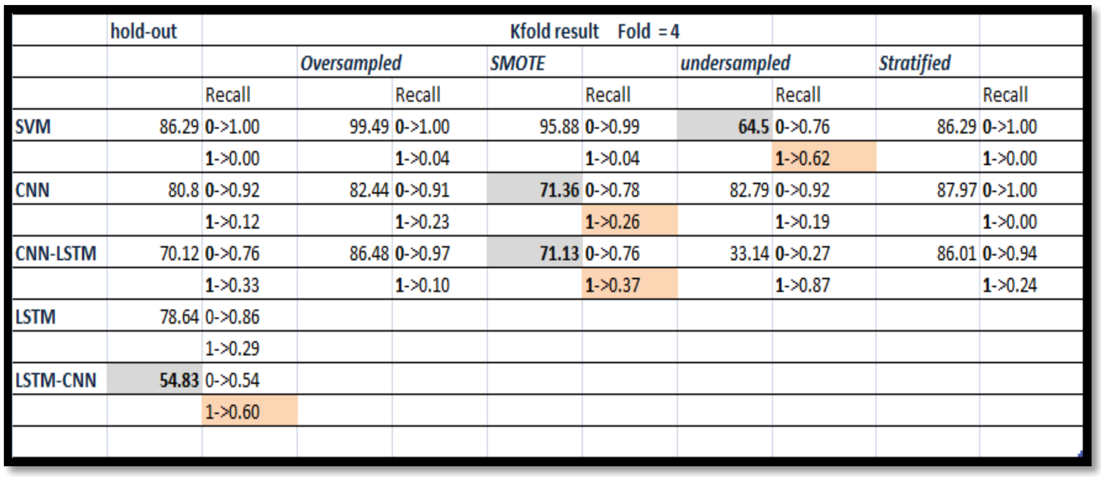


Figure 4.7: SMOTE strategy for imbalanced data[8]

[8] https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets



	hold-out		Kfold result Fold =4							
			Oversampled		*SMOTE*		*undersampled*		*Stratified*	
		Recall		Recall		Recall		Recall		Recall
SVM	86.29	0->1.00	99.49	0->1.00	95.88	0->0.99	**64.5**	0->0.76	86.29	0->1.00
		1->0.00		1->0.04		1->0.04		1->0.62		1->0.00
CNN	80.8	0->0.92	82.44	0->0.91	**71.36**	0->0.78	82.79	0->0.92	87.97	0->1.00
		1->0.12		1->0.23		1->0.26		1->0.19		1->0.00
CNN-LSTM	70.12	0->0.76	86.48	0->0.97	**71.13**	0->0.76	33.14	0->0.27	86.01	0->0.94
		1->0.33		1->0.10		1->0.37		1->0.87		1->0.24
LSTM	78.64	0->0.86								
		1->0.29								
LSTM-CNN	**54.83**	0->0.54								
		1->0.60								

Table 4.7: Performance comparison of the model tested with different sampling techniques and different data splitting approach

Hybrid models for deep learning are better learning than simple approaches as can be seen from the results given in the table above. If considering hold-out approach has been seen that the hybrid model i.e. CNN-LSTM and LSTM-CNN are performing well with 33 and 60% recall for offensive class which tells when the models are getting complicated it is providing better results. CNN-LSTM is the best approach for this research. LSTM-CNN can also be considered as good model as it is performing very well on hold-out so if tried with K-fold can give better results also which was not considered for this research due to lack of computational resources.

These results are explained with proof by doing the statistical T-Tests on them. For this research as more than 2 methods are used Friedman Chi-Square test is done which give results when more than 2 methods or data are involved. Each model is tested for samples of recall score 10 times giving the array of ten values. From the results, it can be seen that the value of the statistic is 13.282 and p-value is 0.001 which is less than 0.05 rejecting the null hypothesis as stated earlier. For these three models are compared i.e. SVM, CNN and CNN-LSTM whose result was obtained by k-fold. This can be added with more methods when and if tried with K-fold cross-validation to obtain samples (measurements in an array) to be compared with other models.

CHAPTER 5 – CONCLUSION

5.1. Research Overview

The research carried out is to predict offensive comments from YouTube dataset. Nowadays the problem of offensive language is increasing like anything over social media. YouTube is a platform where people have the freedom to talk about whatever they want to say. This is a social platform where human being of every age joins and find content relevant for them. Comments cannot be stopped but can be detected and then it’s the users choose what they want to do with it like blocking or updating their own content from it. Sentiment analysis is a booming approach and a very wide area. Many approaches are considered to do this analysis. Here Deep learning, Machine Learning and Natural Language Processing are used to do offensive sentiment analysis for YouTube comments.

5.2. Problem Definition

The research problem was defined by the question ‘which end to end Machine Learning approach is most effective in detecting offensive comments on YouTube’.

The primary purpose of the model was to establish the validity of:

Null Hypothesis: Simple deep learning models (CNN) approaches give higher statistically significant performance metrics than Hybrid deep learning (CNN-LSTM) approach while detecting offensive comments from YouTube.

Alternate Hypothesis: Hybrid deep learning (CNN-LSTM) approach give higher statistically significant performance metrics than simple deep learning models (CNN) approaches while detecting offensive comments from YouTube.

The main problem that was observed was the imbalanced class problem as offensive class instances were comparatively less than non-offensive class. To overcome these different sampling approaches were used to try how they perform to train a model.

Also, models were tested for two techniques of splitting the data i.e. hold-out and K-Fold which also showed tremendous variation in results. Other deep learning models were also trained which were LSTM and LSTM-CNN.

5.3. Design/Experimentation, Evaluation & Results

CRISP-DM is the approach for the design of the research. YouTube comments dataset is understood, prepared, modeling is done and then these models are evaluated. Different experiments are carried out when then data is pre-processed and features are extracted from it. Support Vector Machine is implemented as the baseline model. The main experiments carried out are the deep learning models which are Convolutional Neural Network, Convolutional Neural Network- Long-Short Term Memory, Long-Short Term Memory, and Long-Short Term Memory-Convolutional Neural Network. CNN-LSTM and LSTM-CNN are the hybrid models of simple CNN and LSTM. These models are tested using different sampling approaches and different splitting techniques because of the big imbalanced problem faced. I.e. oversampling, under sampling, SMOTE and Stratified k-fold cross-validation. For hold-out and 4 fold cross-validation.

Interesting results were observed as it was seen that the baseline model (SVM) works best for under-sampling approach as it is considered as a simple model. But for the deep neural network models, it was seen that the results for SMOTE sampling approach are better as deep learning models are complicated and SMOTE techniques are also complex so they learn more for this approach with K-fold. When making the model more complex as in using the hybrid approach of the models it was seen that they performed better than simple deep learning approaches.

The model was compared on the basis of the recall performance metric considering other metrics too and if both the classes are learning well for the model. It was observed that when accuracy is high F1-score and recall was less for the offensive class but when the accuracy starts to decrease F1- score and recall starts to increase giving a better learning model. This proves that accuracy cannot be always considered as the performance metric for the model but other metrics equally contribute to increasing or decreasing the performance of the models.

It was observed that hybrid approaches are a better model for this problem as CNN- LSTM worked very well with the highest recall in K-fold. LSTM-CNN model was also working well for the hold-out approach itself. These results were verified by doing some of the statistical tests. For this problem, Friedman Chi-square statistical test was applied giving the p-value of 0.001 and rejecting the null hypothesis.

5.4. Contributions and impact

In this work, a thorough analysis and processing of the data from YouTube having offensive comments were performed. As the dataset used is new and manually labeled. SVM is tested for this approach before considering AUC as the performance metrics but deep learning models are not tested for this dataset which can explain totally different and new research to be carried out for YouTube comments.

This will impact sentiment analysis and how its approach can be improved or different ways can be used to increase the effectiveness. Apart from applying the methods processing data techniques result in different results.

5.5. Future Work & Recommendations

LSTM models require more computational capacity and resources to run. With access to better resources, LSTM and LSTM-CNN can be used with K-folds stating their results and proving that those can be considered as good approach to solve this problem with more accurate and better results.

The amount of data available is a big factor while considering problems with deep learning approaches. More and more data can be added to this and models tested to prove the results and performing statistical tests for those to give a much better understanding of the problem and answer to them. T-tests are usually conducted when approx 30 datasets are used to compare the results (Demsar, 2006).

Other sampling approaches can be used to test the models such as *Under-sampling: Tomek links*, *Under-sampling: Cluster Centroid*, Over-sampling followed by under-sampling, *Recommended reading,* etc to look for different, interesting and various results.

As for this research, manual tuning is carried out for the models but another better approach of choosing the hyper parameters is by using grid search. This lets the model tune different values for different parameters giving the best fit values for the model. This can be done at the cost of high computational capacity.

3 BIBLIOGRAPHY

Alberto, T. C., Lochter, J. V., & Almeida, T. A. (2015). TubeSpam: Comment Spam Filtering on YouTube. *IEEE 14th International Conference on Machine Learning and Applications*, pp. 82-104. doi:10.1109/ICMLA.2015.37

Anagnostou, A., Mollas, I., & Tsoumakas, G. (2018). Hatebusters: A Web Application for Actively Reporting YouTube Hate Speech. *In IJCAI*, pp. 5796-5798.

Chatzakou, D., Kourtellis, N., Blackburn, J., Cristofaro, E. D., Stringhini, G., & Vakali, A. (2017). Detecting Aggressors and Bullies on Twitter. *Proceedings of the 26th International Conference on World Wide Web Companion*, pp. 190-201. doi:doi:10.1145/3041021.3054211

Chawla, N. V., Japkowicz, N., & Kotcz, A. (2004). Special issue on learning from imbalanced dataset. *ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets 6(1)*, pp.1-6. doi:10.1145/1007730.1007733

Chen, H., Mckeever, S., & Delany, S. J. (2017). Abusive Text Detection Using Neural Networks. *International Conference on Web Intelligence*, pp. 884-890. Retrieved from https://doi:10.1145/3106426.3106456.

Chen, H., Mckeever, S., & Delany, S. J. (2018). A Comparison of Classical Versus Deep Learning Techniques for Abusive Content Detection on Social Media Sites. *S. Staab et al. (Eds.): SocInfo 2018, LNCS 11185*, pp. 117–133. doi:10.1007/978-3-030-01129-1_8

Dadvar, M., Trieschnigg, D., & Jong, F. (2014). Experts and Machines against Bullies:A Hybrid Approach to Detect Cyberbullies. *Springer International Publishing Switzerland 2014*, pp. 275–281. doi:10.1007/978-3-319-06483-3_25

Davidson, T., Warmsley, D., Macy, M., & Macy, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. *In Eleventh international aaai conference on web and social media*, pp.91-120. Retrieved from http://arXiv:1703.04009v1

Demsar, J. (2006). Statistical Comparisons of Classifiers over multiple data sets. *Journal of Machine Learning Research 7*, pp.1–30. Retrieved from http://citeseerx.ist.psu.edu

Deng, L., & Yu, D. (2014). Deep Learning: Methods and Applications. *Foundations and Trends® in Signal Processing: Vol. 7: No. 3–4*, pp 197-387. Retrieved from http://dx.doi.org/10.1561/2000000039

Derczynski, L., Ritter, A., Clark, S., & Bontcheva, K. (2013). Twitter Part-of-Speech Tagging for All: Overcoming Sparse and Noisy Data. *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pp.198-206. Retrieved from https://www.aclweb.org/anthology/R13-1026

Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V., & Bhamidipati, N. (2015). Hate Speech Detection with Comment Embeddings. *WWW 2015 Companion*, pp.18-22. Retrieved from http://dx.doi.org/10.1145/2740908.2742760.

Donahue, J., Hendricks, L. A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., & Darrell, T. (2016). Long-term Recurrent Convolutional Networks for Visual Recognition and Description. *TPAMI journal article*, pp.42-92. Retrieved from https://arXiv:1411.4389

Gaydhan, A., Doma, V., Kendre, S., & Bhagwat, L. (2018). Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TFIDF based Approach. *presented at IEEE International Advance Computing Conference 2018*, pp. 20-40. Retrieved from http://arXiv:1809.08651v1

Georgakopoulos, S. V., Tasoulis, S. K., Vrahatis, A. G., & Plagianakos, V. P. (2018). Convolutional Neural Networks for Toxic Comment Classification. *SETN '18 Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, pp. 4503-6433. doi:10.1145/3200947.3208069

Ghosal, D., Bhatnagar, S., Akhtar, M. S., Ekbal, A., & Bhattacharyya, P. (2017). IITP at SemEval-2017 Task 5: An Ensemble of Deep Learning and Feature Based Models for Financial Sentiment Analysis. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pp. 899–903. doi:10.18653/v1/S17-2154

Gibert, O. d., Perez, N., García-Pablos, A., & Cuadros, M. (2018). Hate Speech Dataset from a White Supremacy Forum. *Proceedings of the Second Workshop on Abusive Language Online (ALW2)*, pp. 11–20. doi:10.18653/v1/W18-5102

Kegelmeyer, P., Hall, L. O., Bowyer, K. W., & Chawla, N. V. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research 16*, pp.321-357. Retrieved from https://doi.org/10.1613/jair.953

Kumar, A. M., & Sheshadr, H. (2012). On the Classification of Imbalanced Datasets. *International Journal of Computer Applications*, pp.0975 – 8887.

Kyun, K. T. (2015). T-test as a parametric statistic. *Korean J Anesthesiol.*, pp. 540–546. doi:10.4097/kjae.2015.68.6.540

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent Convolutional Neural Networks for Text Classification. *AAAI Publications, Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 419-500. Retrieved from www.aaai.org

Li, W., Liu, P., Zhang, Q., Liu, W., & Liu. (2019). An Improved Approach for Text Sentiment Classification Based on a Deep Neural Network via a Sentiment Attention Mechanism. *Future Internet 2019*, pp. 11-19. Retrieved from https://doi.org/10.3390/fi11040096

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Andrew, Y. N., & Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pp. 142–150. Retrieved from https://dl.acm.org

Mara, l. D., Dolf, T., & Franciska, d. J. (2014). Experts and Machines against Bullies:A Hybrid Approach to Detect Cyberbullies. *Springer International Publishing Switzerland 2014*, pp. 275–281. doi:10.1007/978-3-319-06483-3_25

Niu, X.-X., & Suen, C. Y. (2012). A novel hybrid CNN–SVM classifier for recognizing handwritten digits. *Pattern RecognitionVolume 45, Issue 4*, pp. 1318-1325. Retrieved from https://doi.org/10.1016/j.patcog.2011.09.021

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment Classi¯cation using Machine Learning Techniques. *Proceedings of the Conference on Empirical Methods in NaturalLanguage Processing (EMNLP)*, pp. 79-86.

Pitsilis, G. K., Ramampiaro, H., & Langseth, H. (2018). Detecting Offensive Language in Tweets Using Deep Learning. *Applied Intelligence, 48(12)*, pp. 4730-4742. doi:10.1007/s10489-018-1242-y

Prasar, A. G., Sanjana, S., Bhat, S. M., & Harish, B. S. (2017). Sentiment Analysis for Sarcasm Detection on Streaming Short Text Data. *2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA)*, pp. 1-5. doi:10.1109/ICKEA.2017.8169892

Rojas-Barahona, L. M. (2016). Deep learning for sentiment analysis. *Lang Linguist Compass*, pp701–719.

Schmidt, A., & Wiegand, M. (2017). A Survey on Hate Speech Detection using Natural Language Processing. *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pp.1-10. doi:10.18653/v1/W17-1101

Shylaja, S., & Chavan, V. S. (2015). Machine Learning Approach for Detection of Cyber-Aggressive Comments by Peers on Social Media Network. *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp.7-13. doi:10.1109/ICACCI.2015.7275970

Siersdorfer, S., Chelaru, S., Nejdl, W., & Pedro, J. S. (2010). How Useful are Your Comments?Analyzing and Predicting YouTube Comments and Comment Ratings. *International World Wide Web Conference Committee(IW3CC2)*, pp.26-30. doi:10.1145/1772690.1772781

Singh, A., & Purohit, A. (2015). A Survey on Methods for Solving Data Imbalance problem for classification. *International Journal of Computer Applications*, pp.204 – 477.

Sood, S. O., ChurchillElizabeth, F., & Antin, J. (2012). Automatic Identification of Personal Insults on social news sites. *Journal of the american society for information Sciencee and Technology*, pp. 270-285. doi:10.1002/asi.21690

Tripathy, A., Agrawal, A., & Rath, S. K. (2016). Classification of sentiment reviews using n-gram machine learning approach. *ElsevierLtd.*, pp 0957-4174. Retrieved from http://dx.doi.org/10.1016/j.eswa.2016.03.028

Tungthamthiti, P., Shirai, K., & Mohd, M. (2017). Recognition of Sarcasm in Microblogging Based on Sentiment Analysis and Coherence Identification. *Information and meadia technologies12*, pp. 80-102. Retrieved from https://doi.org/10.11185/imt.12.80

Uryupina, O., Plank, B., Severyn, A., Rotondi, A., & Moschitti, A. (2014). SenTube: A Corpus for Sentiment Analysis on YouTube Social Media. *in LREC*, pp 59-90.

Verma, A., & Yenter, A. (2017). Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis. *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pp.540-546. doi:10.1109/UEMCON.2017.8249013

Wang, S., & Manning, C. D. (2012). Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. *In Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*, (pp. 90-94).

Wang, X., Jiang, W., & Luo, Z. (2016). Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis for Short Text. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp.2428-2428.

Waseem, Z., Davidson, T., Warmsley, D., & Weber, I. (2017). Understanding Abuse: A Typology of Abusive Language Detection Subtasks. *the proceedings of the 1st Workshop on Abusive Language Online.*, pp. 066-149.

Wulczyn, E., Thain, N., & Dixon, L. (2017). Ex Machina: Personal Attacks Seen at Scale. *International World Wide Web Conference Committee (IW3C2)*, pp.13-91. Retrieved from http://dx.doi.org/10.1145/3038912.3052591

Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative Study of CNN and RNN for Natural Language Processing. *arXiv preprint arXiv:1702.01923.*, pp. 89-215.

You, Z., Hang, Y., Jin, W., & Xuejie, Z. (2017, September). YNU-HPCC at EmoInt-2017: using a CNN-LSTM model for sentiment intensity prediction. . *In Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pp. 200-204.

Zhang, L., Wang, S., & Liu, B. (2018). Deep Learning for Sentiment Analysis: A Survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4)*, pp.64-92. Retrieved from https://doi.org/10.1002/widm.1253

Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *arXiv:1510.03820 [cs.CL]*, 882-990. Retrieved from https://arxiv.org/abs/1510.03820

Zhang, Y., Yuan, H., Wang, J., & Zhang, X. (2017). YNU-HPCC at EmoInt-2017: using a CNN-LSTM model for sentiment intensity prediction. *In Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pp. 200-204.

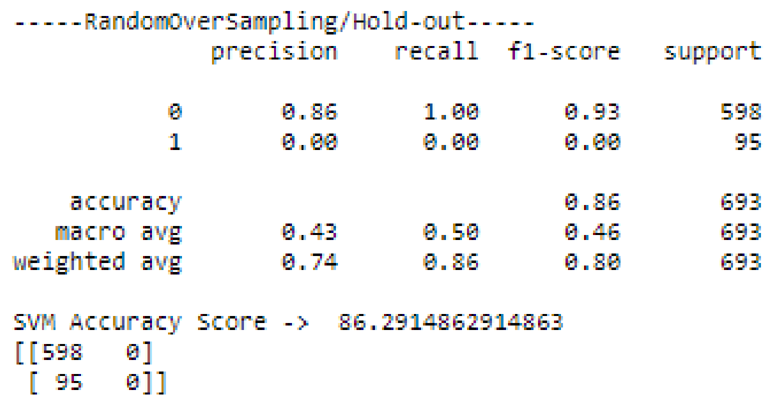
Jeffrey P., Richard S., and ChristopherD. M. (2014). Glove: Global vectors forword representation. *In Empirical Methods in NaturalLanguage Processing (EMNLP)*. pp. 1532-1543. http://www.aclweb.org/anthology/D14-1162.

Godin F., Baptist V., Wesley De N. and Rik V. de W. (2015). Multimedia shared task: named entity recognition for twitter microposts using distributed word representations. *ACL-IJCNLP 2015*, pp.146–153.

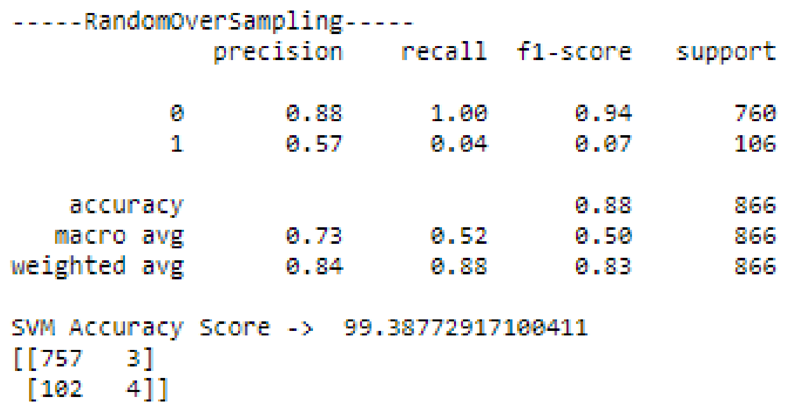
Theresa W., Janyce W. and Paul H. (2005). Recognizing contextual polarity in phrase level sentiment analysis. *In Proceedings of the conferenceon human language technology and empirical methods in natural language processing.* pp. 347–354.

4 APPENDIX

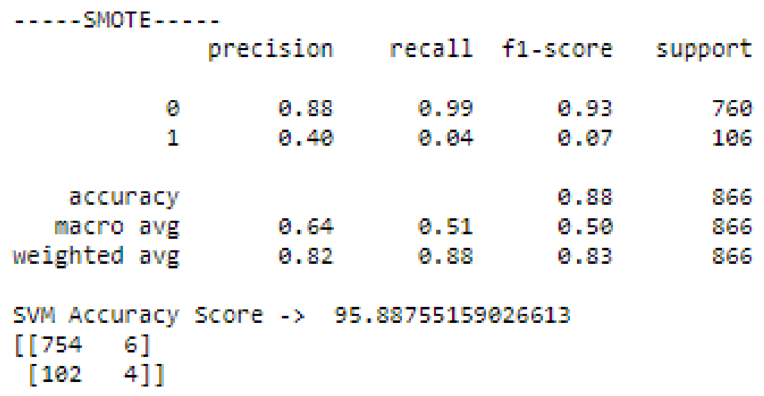
SVM Results



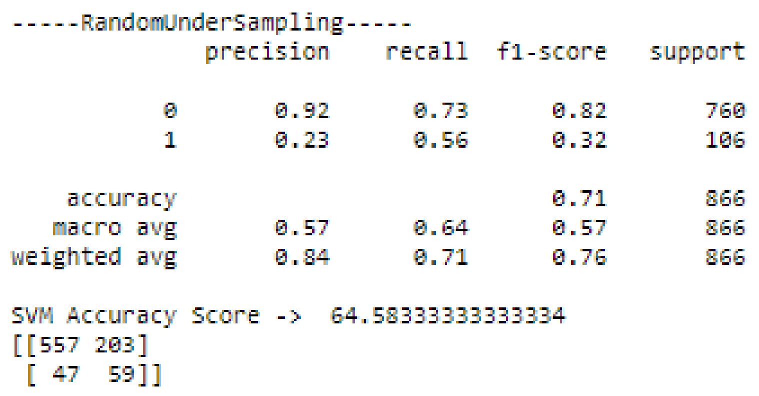
```
-----RandomOverSampling/Hold-out-----
              precision    recall  f1-score   support

           0       0.86      1.00      0.93       598
           1       0.00      0.00      0.00        95

    accuracy                           0.86       693
   macro avg       0.43      0.50      0.46       693
weighted avg       0.74      0.86      0.80       693

SVM Accuracy Score ->  86.2914862914863
[[598   0]
 [ 95   0]]
```



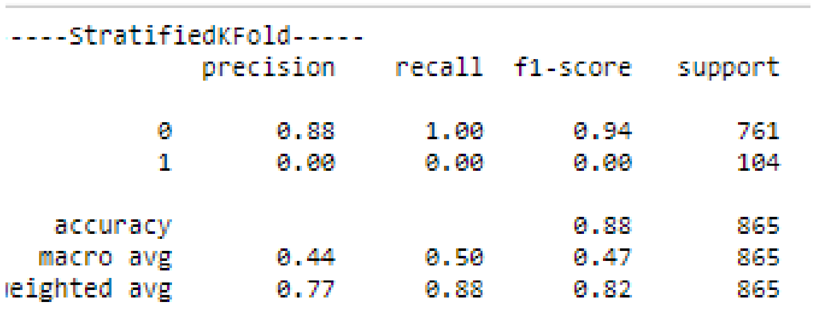
```
-----RandomOverSampling-----
              precision    recall  f1-score   support

           0       0.88      1.00      0.94       760
           1       0.57      0.04      0.07       106

    accuracy                           0.88       866
   macro avg       0.73      0.52      0.50       866
weighted avg       0.84      0.88      0.83       866

SVM Accuracy Score ->  99.38772917100411
[[757   3]
 [102   4]]
```



```
-----SMOTE-----
              precision    recall  f1-score   support

           0       0.88      0.99      0.93       760
           1       0.40      0.04      0.07       106

    accuracy                           0.88       866
   macro avg       0.64      0.51      0.50       866
weighted avg       0.82      0.88      0.83       866

SVM Accuracy Score ->  95.88755159026613
[[754   6]
 [102   4]]
```

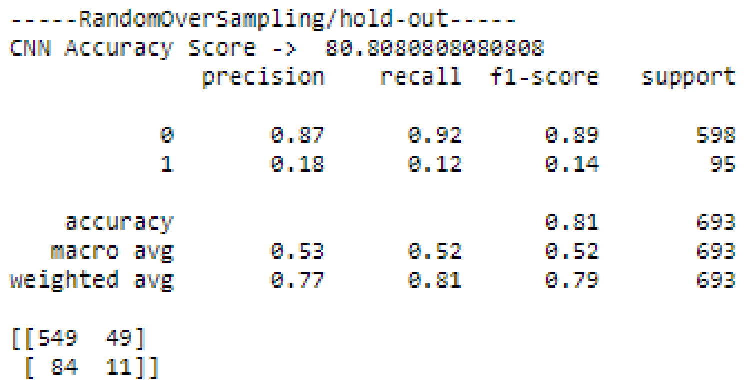


```
-----RandomUnderSampling-----
              precision    recall  f1-score   support

           0       0.92      0.73      0.82       760
           1       0.23      0.56      0.32       106

    accuracy                           0.71       866
   macro avg       0.57      0.64      0.57       866
weighted avg       0.84      0.71      0.76       866

SVM Accuracy Score ->  64.58333333333334
[[557 203]
 [ 47  59]]
```

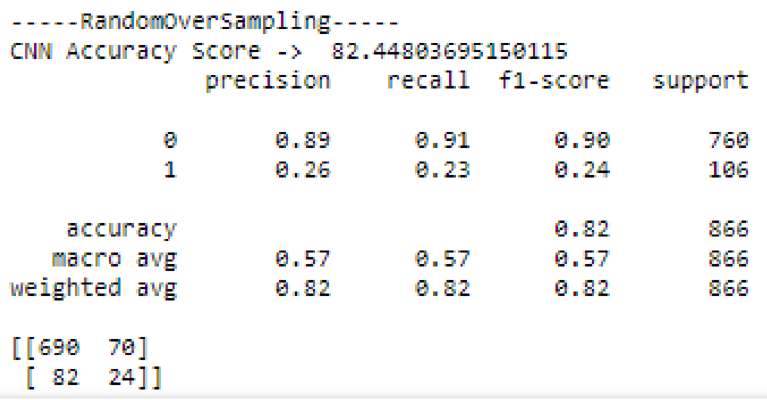


```
----StratifiedKFold-----
              precision    recall  f1-score   support

           0       0.88      1.00      0.94       761
           1       0.00      0.00      0.00       104

    accuracy                           0.88       865
   macro avg       0.44      0.50      0.47       865
ieighted avg       0.77      0.88      0.82       865
```

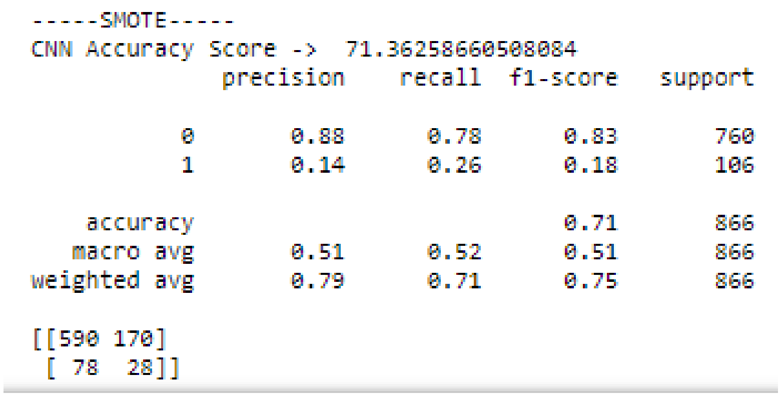
CNN Results



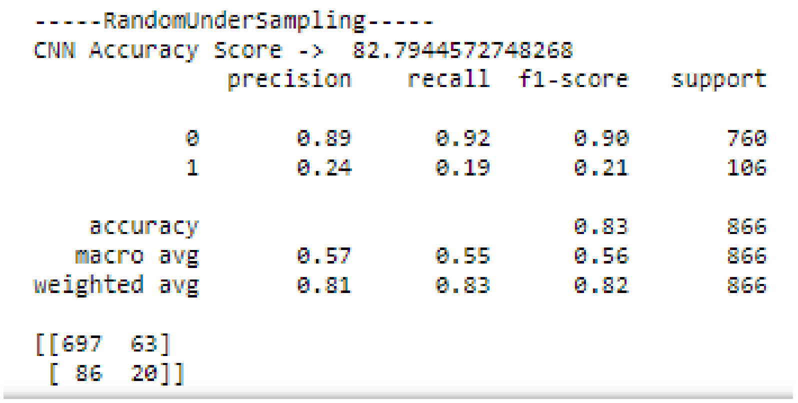
```
-----RandomOverSampling/hold-out-----
CNN Accuracy Score ->  80.8080808080808
              precision    recall  f1-score   support

           0       0.87      0.92      0.89       598
           1       0.18      0.12      0.14        95

    accuracy                           0.81       693
   macro avg       0.53      0.52      0.52       693
weighted avg       0.77      0.81      0.79       693

[[549  49]
 [ 84  11]]
```



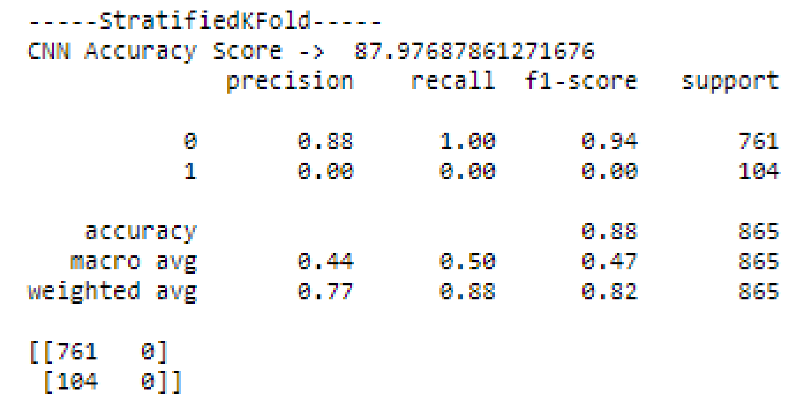
```
-----RandomOverSampling-----
CNN Accuracy Score ->  82.44803695150115
              precision    recall  f1-score   support

           0       0.89      0.91      0.90       760
           1       0.26      0.23      0.24       106

    accuracy                           0.82       866
   macro avg       0.57      0.57      0.57       866
weighted avg       0.82      0.82      0.82       866

[[690  70]
 [ 82  24]]
```



```
-----SMOTE-----
CNN Accuracy Score ->  71.36258660508084
              precision    recall  f1-score   support

           0       0.88      0.78      0.83       760
           1       0.14      0.26      0.18       106

    accuracy                           0.71       866
   macro avg       0.51      0.52      0.51       866
weighted avg       0.79      0.71      0.75       866

[[590 170]
 [ 78  28]]
```

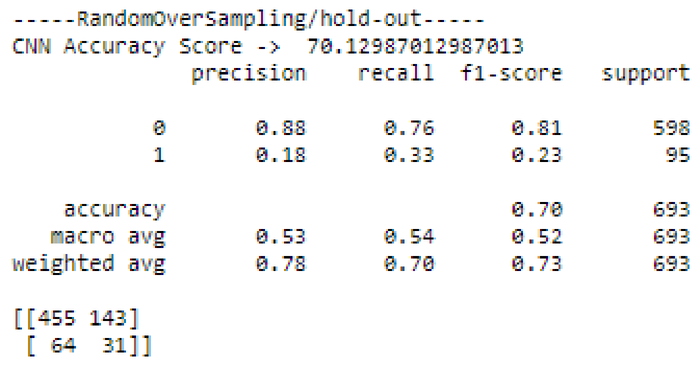


```
-----RandomUnderSampling-----
CNN Accuracy Score ->  82.7944572748268
              precision    recall  f1-score   support

           0       0.89      0.92      0.90       760
           1       0.24      0.19      0.21       106

    accuracy                           0.83       866
   macro avg       0.57      0.55      0.56       866
weighted avg       0.81      0.83      0.82       866

[[697  63]
 [ 86  20]]
```

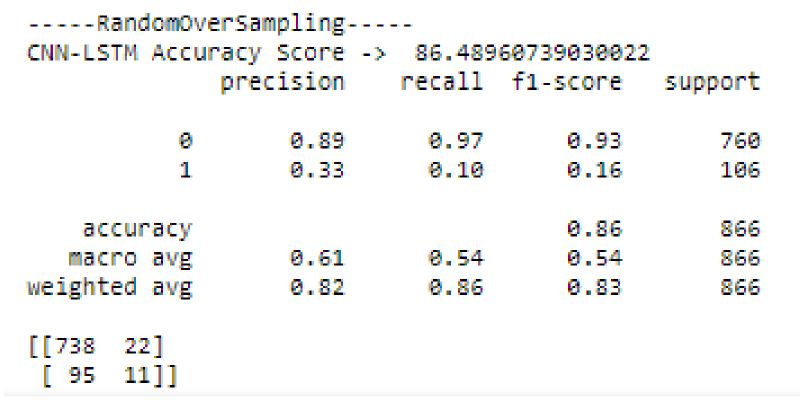


```
-----StratifiedKFold-----
CNN Accuracy Score ->  87.97687861271676
              precision    recall  f1-score   support

           0       0.88      1.00      0.94       761
           1       0.00      0.00      0.00       104

    accuracy                           0.88       865
   macro avg       0.44      0.50      0.47       865
weighted avg       0.77      0.88      0.82       865

[[761   0]
 [104   0]]
```

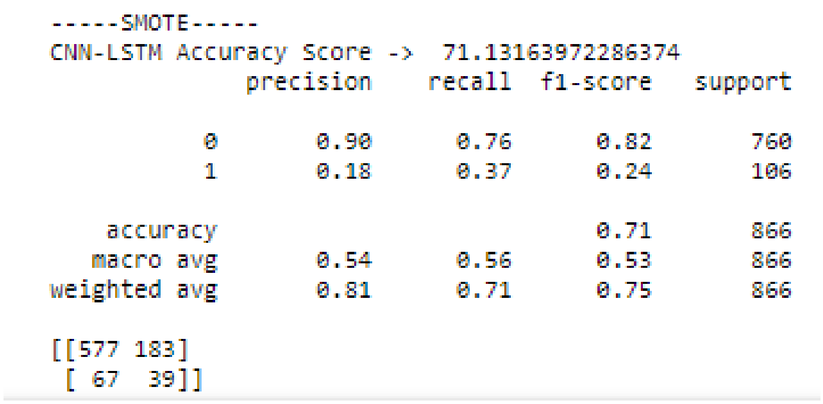
CNN-LSTM Results



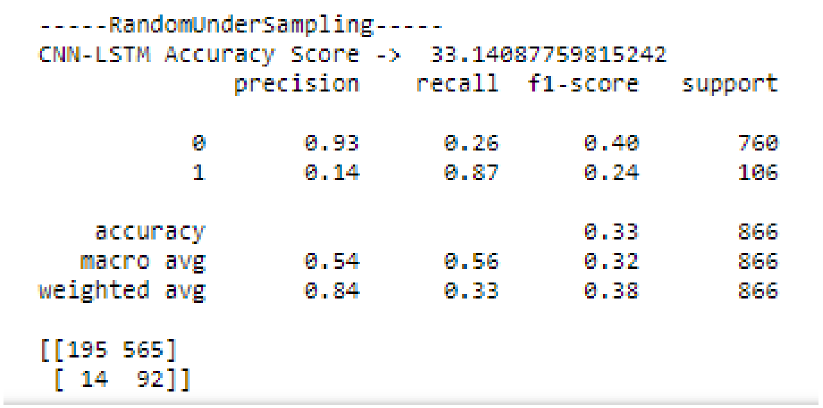
```
-----RandomOverSampling/hold-out-----
CNN Accuracy Score ->  70.12987012987013
              precision    recall  f1-score   support

           0       0.88      0.76      0.81       598
           1       0.18      0.33      0.23        95

    accuracy                           0.70       693
   macro avg       0.53      0.54      0.52       693
weighted avg       0.78      0.70      0.73       693

[[455 143]
 [ 64  31]]
```



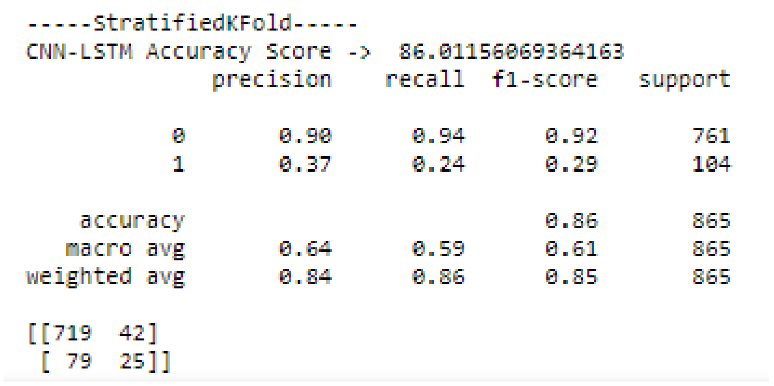
```
-----RandomOverSampling-----
CNN-LSTM Accuracy Score ->  86.48960739030022
              precision    recall  f1-score   support

           0       0.89      0.97      0.93       760
           1       0.33      0.10      0.16       106

    accuracy                           0.86       866
   macro avg       0.61      0.54      0.54       866
weighted avg       0.82      0.86      0.83       866

[[738  22]
 [ 95  11]]
```



```
-----SMOTE-----
CNN-LSTM Accuracy Score ->  71.13163972286374
              precision    recall  f1-score   support

           0       0.90      0.76      0.82       760
           1       0.18      0.37      0.24       106

    accuracy                           0.71       866
   macro avg       0.54      0.56      0.53       866
weighted avg       0.81      0.71      0.75       866

[[577 183]
 [ 67  39]]
```

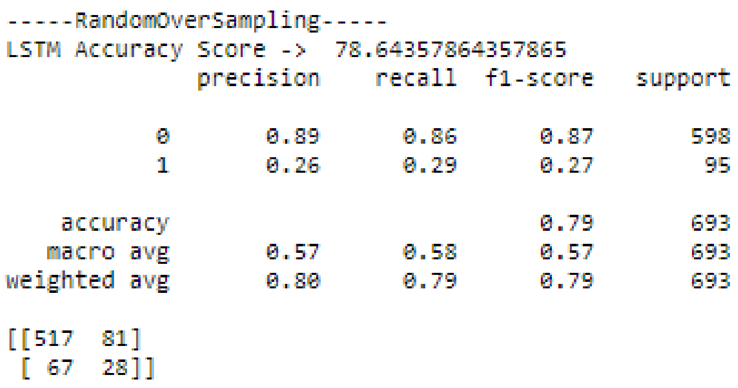


```
-----RandomUnderSampling-----
CNN-LSTM Accuracy Score ->  33.14087759815242
              precision    recall  f1-score   support

           0       0.93      0.26      0.40       760
           1       0.14      0.87      0.24       106

    accuracy                           0.33       866
   macro avg       0.54      0.56      0.32       866
weighted avg       0.84      0.33      0.38       866

[[195 565]
 [ 14  92]]
```



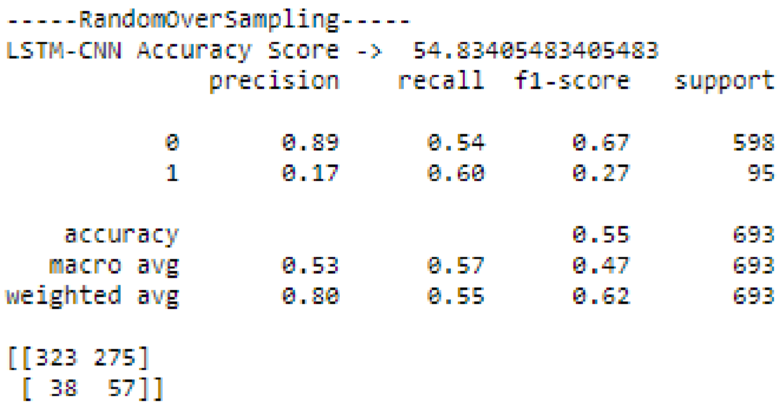
```
-----StratifiedKFold-----
CNN-LSTM Accuracy Score ->  86.01156069364163
              precision    recall  f1-score   support

           0       0.90      0.94      0.92       761
           1       0.37      0.24      0.29       104

    accuracy                           0.86       865
   macro avg       0.64      0.59      0.61       865
weighted avg       0.84      0.86      0.85       865

[[719  42]
 [ 79  25]]
```

LSTM Results



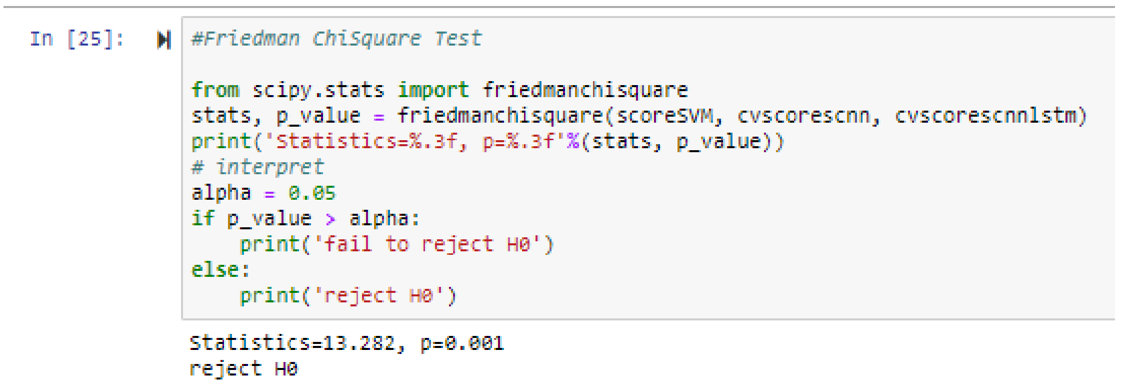
```
-----RandomOverSampling-----
LSTM Accuracy Score ->  78.64357864357865
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       598
           1       0.26      0.29      0.27        95

    accuracy                           0.79       693
   macro avg       0.57      0.58      0.57       693
weighted avg       0.80      0.79      0.79       693

[[517  81]
 [ 67  28]]
```

LSTM-CNN



```
-----RandomOverSampling-----
LSTM-CNN Accuracy Score ->  54.83405483405483
              precision    recall  f1-score   support

           0       0.89      0.54      0.67       598
           1       0.17      0.60      0.27        95

    accuracy                           0.55       693
   macro avg       0.53      0.57      0.47       693
weighted avg       0.80      0.55      0.62       693

[[323 275]
 [ 38  57]]
```

Friedman Chi Square T-Test Result



```
In [25]:  #Friedman ChiSquare Test

          from scipy.stats import friedmanchisquare
          stats, p_value = friedmanchisquare(scoreSVM, cvscorescnn, cvscorescnnlstm)
          print('Statistics=%.3f, p=%.3f'%(stats, p_value))
          # interpret
          alpha = 0.05
          if p_value > alpha:
              print('fail to reject H0')
          else:
              print('reject H0')

          Statistics=13.282, p=0.001
          reject H0
```