

CLICKBAIT DETECTION USING MACHINE LEARNING AND DEEP LEARNING ALGORITHMS

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
ARTIFICIAL INTELLIGENCE

Submitted by

KAPIL KUMAR YADAV
(2K21/AFI/26)

Under the supervision of

Mr. Nipun Bansal



**DEPARTMENT OF COMPUTER SCIENCE
ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY**
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

MAY, 2023

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, **Kapil Kumar Yadav**, Roll No – (2K21/AFI/26) student of M.Tech (**Department of Computer Science Engineering**), hereby declare that the project Dissertation titled “**CLICKBAIT DETECTION USING MACHINE LEARNING AND DEEP LEARNING ALGORITHMS**” which is submitted by me to the **Department of Computer Science Engineering**, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi

Kapil Kumar Yadav

Date: 31.05.2023

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled “**CLICKBAIT DETECTION USING MACHINE LEARNING AND DEEP LEARNING ALGORITHMS**” which is submitted by **Kapil Kumar Yadav**, Roll No – (2K21/AFI/26), **Department of Computer Science Engineering**, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Nipun Bansal

Date: 31.05.2023

SUPERVISOR

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I wish to express my sincerest gratitude to Mr. **Nipun Bansal** for his continuous guidance and mentorship that he provided me during the project. He showed me the path to achieve my targets by explaining all the tasks to be done and explained to me the importance of this project as well as its industrial relevance. He was always ready to help me and clear my doubts regarding any hurdles in this project. Without his constant support and motivation, this project would not have been successful.

Place: Delhi

Kapil Kumar Yadav

Date: 31.05.2023

(2K21/AFI/26)

Abstract

Clickbait is a type of providing false content, intentionally to gain a variety of users and get engagement and monetary benefits. It makes users curious to click the link and follow the content in various format like audio, video, text, images. As the online user base is getting bigger and bigger and more and more users are coming online, the unusual activities, scam and clickbait is becoming more common. These clickbait links will take users to some random websites which will have irrelevant information and completely exploits the user experience. The motive behind the clickbait links is to get more views to generate more ad revenue. Clickbait Detection is a crucial and difficult task to be done. Many researchers have proposed various techniques using deep learning and machine learning techniques like Logistic Regression, Linear Support Vector Machine, Adaboost, Random Forest, Multilayer Perceptron, Convolution Neural Networks(CNN). To give the clear overview about the efficient algorithms, we went through some existing studies over the period of 2016-2022 which proposed various clickbait detection methods.

Keywords: Clickbait, Deep Learning, Bi-LSTM, Machine Learning, Naive Baye's.

Contents

Candidate's Declaration	i
Certificate	ii
Acknowledgement	iii
Abstract	iv
Content	vi
List of Tables	vii
List of Figures	ix
List of Symbols, Abbreviations	x
1 INTRODUCTION	1
1.1 Machine Learning	1
1.2 Types of Machine Learning:	2
1.2.1 Supervised Learning:	2
1.2.2 Unsupervised Learning	2
1.2.3 Semi-Supervised Learning.....	3
1.2.4 Reinforcement Learning.....	3
1.3 Deep Learning	3
1.3.1 Long Short-Term Memory	5
1.3.2 Bi-Directional Long Short Term Memory.....	6
1.3.3 Gated Recurrent Unit.....	7
1.4 Introduction to Clickbait and fake news.....	8
2 LITERATURE REVIEW	11
3 METHODOLOGY	18
3.1 Dataset Creation	19
3.2 Data Preprocessing	20
3.3 Text Processing	21
3.4 Machine Learning Models	23
3.4.1 Naive Bayes	23
3.4.2 Random Forest	24
3.4.3 Support Vector Machine (SVM).....	25
3.4.4 Logistic Regression	26

3.4.5	XGBoost	27
3.5	Deep Learning Models	28
3.5.1	Dense Model	28
3.5.2	Long Short Term Memory.....	31
3.5.3	Gated Recurrent Unit.....	34
3.5.4	Bidirectional Long Short Term Memory Model:.....	37
4	RESULTS AND DISCUSSION	42
4.1	Experimental Setup	42
4.2	Evaluation Metrics	42
4.3	Training and Validation Performance Curves	43
4.4	Result Analysis	43
4.5	Handling Overfitting	45
5	CONCLUSION AND FUTURE SCOPE	46
	LIST OF PUBLICATIONS	47

List of Tables

2.1	Performance of SVM classifier.	12
2.2	Performance of blocking approaches.	12
2.3	Performance of Applied Algorithms.	14
2.4	Comparison of Bi-LSTM with Machine Learning Algorithms.	16
2.5	Summary of existing models	17
3.1	Implementation Details and Layerwise Parameters of Bi-LSTM Model	37
3.2	Parameters used to Train the model	41
3.3	Parameters for Data Preprocessing	41
4.1	Performance of Machine Learning Approaches.	44
4.2	Performance of Deep learning approaches.	45

List of Figures

1.1	Artificial Intelligence vs Machine Learning vs Deep Learning	4
1.2	Some News Headlines	8
1.3	Bi-Directional RNN architecture for clickbait detection	9
3.1	Proposed Model Flow	18
3.2	Histogram of Created Dataset: Clickbait and Non-Clickbait Headlines	19
3.3	Count of Number of Words in a Headline.....	20
3.4	Converting Text into Lowercase.....	20
3.5	Data Preprocessing.....	21
3.6	Top 20 Clickbait Headline Words	21
3.7	Top 20 Non-Clickbait Headline Words	22
3.8	Naive Bayes Classifier.....	23
3.9	Confusion Matrix of Naive Bayes Classifier	23
3.10	Random Forest Classifier.....	24
3.11	Confusion Matrix of Random Forest	24
3.12	Support Vector Machine	25
3.13	Confusion Matrix of Support Vector Machine	25
3.14	Logistic Regression	26
3.15	Confusion Matrix of Logistic Regression.....	26
3.16	XGBoost.....	27
3.17	Confusion Matrix of XGBoost	27
3.18	Dense Model	28
3.19	Summary of the Dense model	29
3.20	Training and Validation loss of Dense Model.....	30
3.21	Training and Validation accuracy of Dense Model.....	30
3.22	Long Short Term Memory Model Flow	31
3.23	Long Short Term Memory Model	31
3.24	Summary of the Long Short Term Memory Model	32
3.25	Training and Validation loss of LSTM.....	33
3.26	Training and Validation accuracy of LSTM.....	33
3.27	Gated Recurrent Unit Model Flow	34
3.28	Gated Recurrent Unit Model.....	34
3.29	Summary of the Gated Recurrent Unit Model	35
3.30	Training and Validation loss of GRU	36
3.31	Training and Validation accuracy of GRU	36
3.32	Bidirectional Long Short Term Memory Model Flow	38
3.33	Bidirectional Long Short Term Memory Model	38
3.34	Summary of the Bidirectional Long Short Term Memory Model	39
3.35	Training and Validation loss of Bi-LSTM.....	40
3.36	Training and Validation accuracy of Bi-LSTM.....	40

4.1	Confusion Matrix	42
4.2	Comparison of Machine Learning Models	44
4.3	Comparison of Deep Learning Models	45

List of Abbreviations

ML	-	Machine Learning Deep
DL	-	Learning
TF-IDF	-	Term Frequency-Inverse Document Frequency Convolution
CNN	-	Neural Network
NLP	-	Natural Language Processing
LSTM	-	Long-Short Term Memory
SVM	-	Support Vector Machine
ANN	-	Artificial Neural Network
AI	-	Artificial Intelligence
R-CNN	-	Region Based Convolution Network

CHAPTER 1

INTRODUCTION

1.1 Machine Learning

In the last few years, Internet has ruled the world, and the data flow through internet had been increased significantly. The data is available in large chunk, containing organized data as well as unorganized data. With the increase in data flow, maintaining the data as well as collecting relevant information from the data becomes a challenging task. Data allows several businesses to make decision using the data in such that can be conclude and represent by companies.

It's a very tedious task to find the useful information from the available data, it may be semi organized or unorganized. If we organize data manually, it will take years to organize the data, and that will be of no use, so we need machines to organize the data and they should be smart enough to find out the relevant information from the chunk of data. We need to train the machines, so that machines should keep learning from their past experiences. In the recent years, Artificial Intelligence domain has reach to the new heights, especially machine learning, deep learning and natural language processing, have gained a lot of popularity with the introduction of new fast and efficient algorithms with high computation power.

Machine learning deals with the computers and machines and gives them the ability to learn without being externally programmed to do so. ML algorithms use past data as an input data to find out new output data. The idea is to make the systems learn from the previous data, analyses the data, identify the patterns and take decisions without human involvement.

1.2 Types of Machine Learning:

Different types of Machine learning algorithms are: Supervised learning, Unsupervised Learning, Semi-Supervised Learning, Reinforcement Learning.

1.2.1 Supervised Learning:

It is a Supervised Learning Technique. Supervised Learning uses “labeled” dataset which we use to train our machines and our machine predicts the output based on the learning technique used. Labeled data here means that for input values in the dataset, we have corresponding outputs for the particular input.

Some of the real-world applications of Supervised Learning are Market prediction, Image Classification, spam Detection, Speech Recognition, Object-Recognition etc.

1.2.2 Unsupervised Learning

Unsupervised Learning needs no supervision. Unsupervised Learning uses “Unlabeled dataset”, which we use to train our machines and the machine do prediction without any supervision. Clustering is used in Unsupervised learning, which creates model based on certain common properties and group them together in a cluster. For example, whether a credit card should be given to the user or not is based on the properties of the users, which matches the similar behavior of users with same characteristics past.

1.2.3 Semi-Supervised Learning

Semi-Supervised Learning works with both Labeled dataset as well as unlabeled dataset. The unlabeled data set is in large amount than the labeled dataset. We use the supervised learning to train the model and feed the labels to supervised learning.

For Example – It can be used in Image dataset, where we have only some part of labeled dataset, e.g. Tiger, Cat, dog, etc have major part of unlabeled dataset.

1.2.4 Reinforcement Learning

Reinforcement Learning is a unexplored area of Machine Learning. It can be consider as close to human in terms of learning. At every stage, it try to maximize the reward, by interacting with the system and the environment. It tries to learn the behavior and pattern by regular feedback.

1.3 Deep Learning

Deep learning methods eliminate the need for explicit feature engineering by automatically discovering and extracting useful patterns and representations from unstructured data. Deep neural networks, which include many interconnected layers of artificial neurons or nodes, are used to achieve this. Each neuron generates an output by taking input data, applying weights and biases, and then putting it through an activation function. It is a powerful approach to solving complex problems by training deep neural networks.

Forward propagation and backpropagation are the two key processes in the training of a deep learning model. Forward propagation involves feeding input data into the network so it can generate predictions. Then, an error metric, such as mean squared error, is calculated by comparing these predictions to the actual target values. The network's weights and biases are then iteratively adjusted via backpropagation to reduce the error. Deep learning offers the advantage of automatically

learning hierarchical representations from data. As data flows through the layers of a deep neural network, each layer learns to represent increasingly difficult and abstract features.

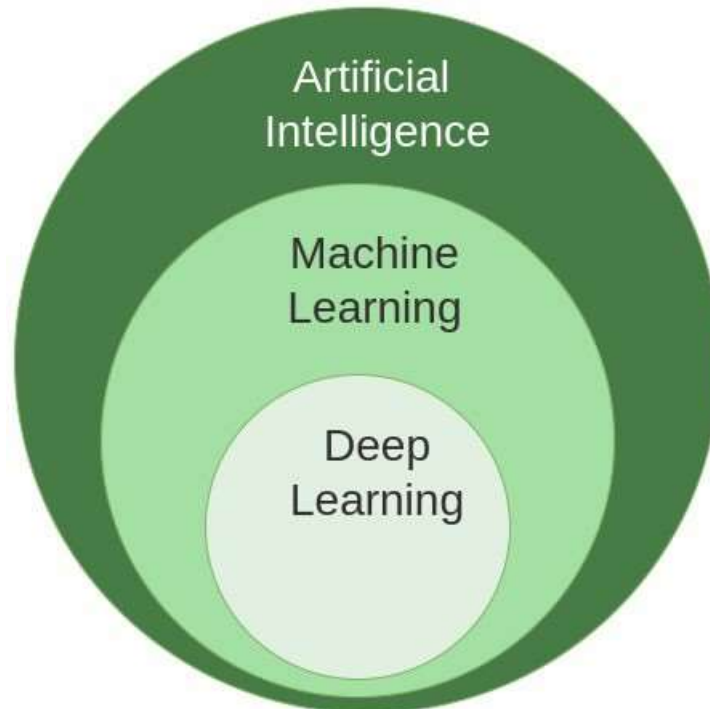


Figure 1.1: Artificial Intelligence vs Machine Learning vs Deep Learning

However, deep learning also presents challenges. Training deep neural networks requires substantial labeled training data, which can be time-consuming and expensive to obtain. Additionally, deep learning models can be computationally intensive, necessitating powerful hardware like graphics processing units (GPUs) for efficient training. Furthermore, issues such as overfitting and interpretability of deep learning models are still areas of ongoing research.

In conclusion, deep learning is an influential and rapidly advancing field of artificial intelligence that shows great potential in solving complex problems across various domains. By leveraging large datasets and deep neural networks, Deep learning models are capable of discovering patterns and representations from data on their own, which enables them to perform well in tasks that were previously difficult for traditional machine learning techniques.

1.3.1 Long Short-Term Memory

In many areas of artificial intelligence and machine learning, Long Short-Term Memory (LSTM) has attracted substantial attention and success. The vanishing gradient problem, which makes it difficult to detect long-term relationships in sequential data, was one of the problems with classic RNNs that LSTM was created to solve.

Memory cells and gating mechanisms are specially incorporated into LSTM networks to collect and model sequential patterns. The memory cells act as storage units, enabling the network to retain and utilize important information even when there are long time gaps between relevant events in the sequence. , LSTM has found extensive applications in various domains and has demonstrated remarkable performance in tasks involving sequential data. In natural language processing, LSTM has been successfully employed for tasks such as text generation, sentiment analysis and machine translation. In time series analysis, LSTM has shown promise in applications like weather forecasting, and energy load prediction. It has also been utilized in speech recognition, handwriting recognition, and other areas where sequential data analysis is crucial.

Capturing both short-term and long-term dependencies in data is one of the key benefits of LSTM. The LSTM is very useful for modelling complex sequential patterns because it has memory cells and gating features that allow it to selectively store and use relevant information. LSTM networks are versatile and appropriate for jobs involving inputs of varied durations because they can handle variable-length sequences.

However, We should keep in mind that LSTM models can be very expensive and can cost a huge, when working with large datasets. Additionally, determining the optimal architecture and hyperparameters for a specific task often requires careful experimentation and tuning. By using the memory cells and gating mechanisms, LSTM networks have demonstrated the ability to effectively model complex sequential patterns.

1.3.2 Bi-Directional Long Short Term Memory

The bidirectional processing used in Bi-LSTM allows it to capture both past and future context in sequential input. It is an expansion of the conventional LSTM architecture. Bi-LSTM networks have seen a lot of success and attention in a variety of sequential data applications, including time series analysis, speech recognition, and natural language processing.

Unlike traditional LSTMs that process sequences in a unidirectional manner, The input sequence is processed by the Bi-LSTM simultaneously in forward and reverse. This is achieved by using two sets of hidden states, with one set processing the sequence from the beginning to the end, and the other set processing the sequence in the reverse order. By considering information from both directions, Bi-LSTM can capture dependencies and patterns that may exist in either the past or the future context of each time step.

Bi-LSTM combines the forward and backward hidden states at each time step, giving the network access to data coming from both directions. This enables the model to make more informed predictions or representations by considering a broader context compared to traditional LSTMs. Bi-LSTM has demonstrated significant advantages in tasks where context from both past and future time steps is crucial. By considering the context both before and after a word, Bi-LSTM can better understand capture the nuanced meaning and relationships in natural language.

In time series analysis, Bi-LSTM has also shown promising results. By leveraging information from both earlier and future time points, Bi-LSTM can better capture temporal dependencies and make more accurate predictions. This is particularly useful in finding weather prediction, and anomaly detection.

One important consideration when using Bi-LSTM is the increased computational complexity compared to traditional LSTMs due to the bidirectional processing. Training Bi-LSTM models may require more computational resources and longer training times.

In conclusion, Bi-LSTM is an extension of the LSTM architecture that incorpo-

rates bidirectional processing to capture both past and future context in sequential data. By considering information from both directions, Bi-LSTM can capture a broader context and dependencies, leading to improved performance in tasks requiring a comprehensive understanding of sequential data.

1.3.3 Gated Recurrent Unit

GRU is a kind of RNN architecture which addresses the limitations of traditional RNNs in capturing long-term dependencies and mitigates the vanishing gradient problem. It consists of two primary gates: the reset gate and the update gate. These gates enable GRU to adaptively learn the importance of different inputs and the relevance of past hidden states.

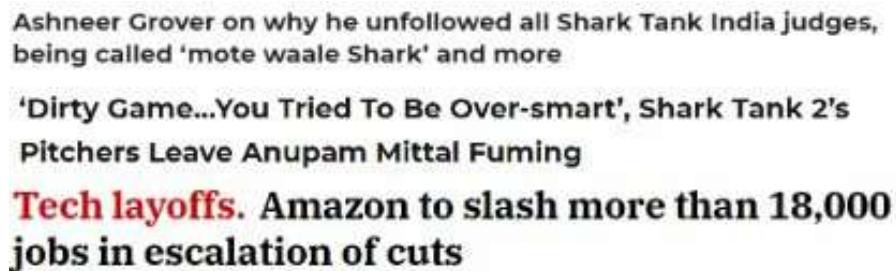
GRU has a simpler design, making it computationally efficient and easier to train. It doesn't utilize a separate memory cell but instead employs the hidden state to capture and store information. GRU has shown promising results in various tasks involving sequential data. In natural language processing, GRU has been successfully applied to translation and modelling applications. In speech recognition, it has been utilized for converting speech to text. In time series analysis, GRU has been employed for forecasting, anomaly detection, and signal processing.

One of the advantages of GRU is its capability to capture dependencies over longer sequences compared to traditional RNNs. By employing gating mechanisms, GRU can selectively remember or forget information based on its relevance.

In conclusion, the GRU is a recurrent neural network architecture that overcomes some limitations of traditional RNNs. With its gating mechanisms and simplified design, GRU efficiently captures dependencies in sequential data, making it suitable for different applications.

1.4 Introduction to Clickbait and fake news

In today's world, with all the advancements in technology and a significant increase in the number of users accessing the internet and online resources, there is also an increase in vulnerable and clickbait data on the internet. Clickbait is simply poor content with no meaning or value-added data, which is fed to the user by exciting them with catchy textual and non-textual data. The purpose of clickbait is to get more views on their sites, which will lead to more revenue via advertisements and other sources. These clickbait articles or news feeds create an irrelevant experience for humans and sometimes distract them to favour a particular agenda. Clickbaits can be categorized into eight types: teasing, ambiguous, formatting, inflammatory, wrong, graphic, exaggeration, and bait and switch [23].



Ashneer Grover on why he unfollowed all Shark Tank India judges, being called 'mote waale Shark' and more
'Dirty Game...You Tried To Be Over-smart', Shark Tank 2's Pitchers Leave Anupam Mittal Fuming
Tech layoffs. Amazon to slash more than 18,000 jobs in escalation of cuts

Figure 1.2: Some News Headlines

In Fig. 1.2, some headlines were given, and they were taken from popular news websites. As we can see, the headlines seem so promising and exciting that there is a high probability of clicking the headlines to read the complete news. In recent years, clickbait has become extremely popular. People in this domain want early success, and for that, they do whatever they feel will increase their business. A lot of research has been already done to detect clickbait, and Facebook in 2014 took action to remove the clickbaits as per ElArini and Tang [24]. Reis et al. [25] took a dataset of 69,000 headlines and observed the polarity of sentiments found in these headlines and extremities in the gained popularity. Headlines are the first impressions of the news and decide whether the user will read it or not, as per Digirolamo and Hintzman [26]. Loewenstein [27] also explained the information gap

theory, according to him, what we know and what we wish to know vary, and this difference has emotional consequences. Users become curious about these gaps. Not knowing makes us uncomfortable. Natural Language Processing (NLP) has gained more popularity to find context and is very frequently used in the below models. To understand the context better, convolutional neural networks (CNN) have been used in many models to improve accuracy, as per Kim [28].

Pothast et al. [29] were one of the first, to work on detecting clickbait on Twitter, but the problem was not limited to Twitter, the same problem exists with other social media platforms as well. Gianotto and Alison [30] came up with approaches like "down-worthy," which transforms the headlines into something more garbageish after applying a predefined set of words to identify clickbait. Using the dataset created by Chakraborty et al. [2], recurrent neural networks were used in an experiment by Anand et al. [19] to recognise clickbait news. They adopted a bi-directional RNN. Fig 1.3 depicts the model architecture and the output of the model.

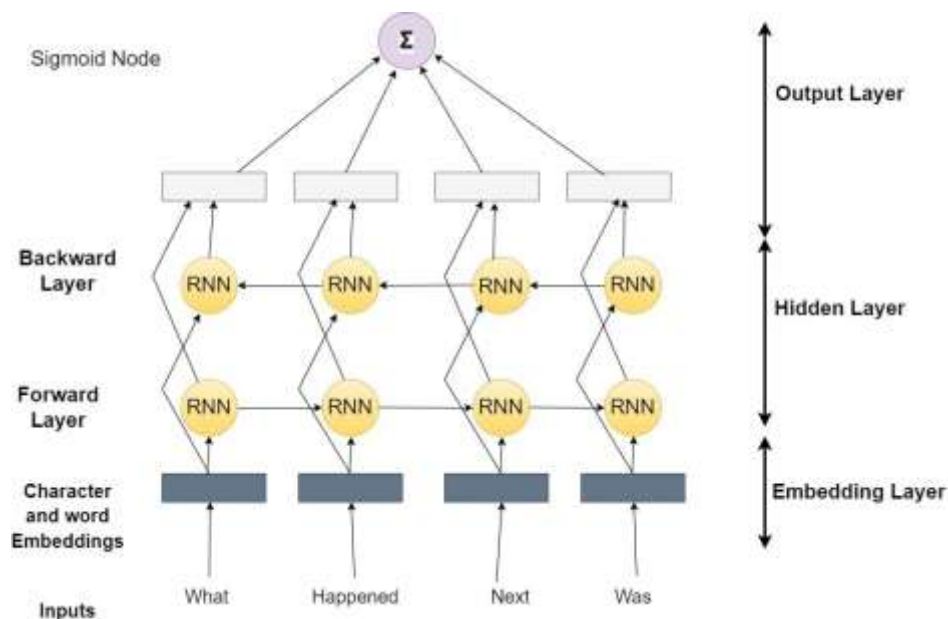


Figure 1.3: Bi-Directional RNN architecture for clickbait detection

Rony et al. [31] analysed the millions of Facebook posts from several organizations. Deep learning approaches have recently achieved success in text categorization, so they built their algorithms utilising distributed subword embeddings rather than bags of words and trained their systems using this dataset. They were

able to achieve 98.3% accuracy. Lie et al. [32] utilised Integer Linear Programming, the addition of the relationship between the images and the text in the technique is what distinguishes their study apart from the competition.

Clickbait is a form of providing false content, intentionally to attract a variety of users and get engagement and monetary benefits. It makes users curious to click the link and follow the content in various format like audio, video, text, images. As the online user base is getting bigger and bigger and more and more users are coming online, the unusual activities, scam and clickbait is becoming more common. Social media platforms like Youtube, Facebook, Twitter gives user a freedom to post their content in text, pictures, videos format.

CHAPTER 2

LITERATURE REVIEW

To identify clickbait, Amol Agrawal [1] has presented a deep learning-based methodology. He used Convolution Neural Networks(CNN) with one layer of convolution, the first layer was used for word embedding, During his experiments, he used both word embeddings created from scratch as well as word embeddings explored using an unsupervised neural model [12][13]. Word Vectors were then trained on billion words of google ¹ news[14].In the next layer, convolutions over word vectors are produced using filters of various sizes (3, 4, and 5). A new feature is produced by each operation. A feature map is created using all the newly generated features. The feature map is subjected to a max-over-time pooling process [15], and the feature chosen for that particular feature map is the one having the highest value. These generated features from the filters constitute the subsequent layer, which is the penultimate layer. After that, a fully linked softmax layer receives these features and outputs the probability distribution over labels. He has used 5-fold cross-validation in his experiments. He compares both the scratch and the non-static word2vec models,and the click-word2vec beats the scratch model. He was able to get the highest F1-Score of 0.86, with precision 0.85, 0.90 accuracy, and recall 0.88.

In [2], A. Chakraborty et al. proposed a method by selecting N-gram features, Word Patterns, Sentence Structure, and Clickbait Languages. They used two approaches, one based on topical similarity and another based on linguistic patterns. They have done experiments using three prediction models - Decision Trees, Random Forests, and Support Vector Machines. They have done 10-fold cross-validation, out

¹<https://code.google.com/p/word2vec/>

of all 3 prediction models, SVM performs the best, with the F1-score of 0.93, recall of 0.90, precision of 0.95, accuracy of 0.93, and ROC-AUC of 0.97 for all features.

Table 2.1: Performance of SVM classifier.

SVM					
Features Used	ROC-AUC	Acc.	F1-Score	Prec.	Rec.
Sentence Structure	0.84	0.77	0.77	0.79	0.75
N-gram Features	0.9	0.89	0.89	0.92	0.85
Clickbait Language	0.88	0.86	0.85	0.88	0.82
Word Patterns	0.91	0.84	0.84	0.84	0.84
All features	0.97	0.93	0.93	0.95	0.90

They performed pattern-based, topic-based, and hybrid blocking approaches, and pattern-based approaches gave better results, so they implemented them into the browser extension. They offered a feature where users could examine the websites that the extension had blocked and provide feedback, whether it was clickbait or not. On average, around 89% of the time, the extension correctly blocked the clickbait sites.

Table 2.2: Performance of blocking approaches.

Approach	F1-Score	Acc.	Prec.	Rec.
Topic Based	0.74	0.75	0.769	0.74
Pattern Based	0.79	0.81	0.834	0.76
Hybrid	0.72	0.72	0.766	0.682

In [3], A. Geçkil et al. demonstrated the model by referencing the Potthast et al. [17] model. In [17], the authors selected 215 elements from among three groups: meta information, linked web pages, and teaser messages. Out of 215 features from the Potthast et al. [17] model, the last four features related to the metadata and the first 19 features of the teaser message were chosen. They classified the data according to the confidence index and decided whether the given headline was clickbait or not. When compared to the non-Clickbait word list, the frequency consistency of the news headline should be greater than 0.08 and less than 0.02, then the headline is considered a Clickbait headline. Using the TextRank algorithm, an unsupervised technique put forth by Rada Mihalcea and Paul Tarau [18], a summary of the titles

has been generated. They were able to get a precision of 0.899, a F-Score of 0.920, a recall of 0.941, and an accuracy of 0.865.

In [4], S. Chawda et al. utilized the context of the titles and preferred Recurrent Convolutional Neural Network (RCNN) for text classification. They further enhanced the proposed model using LSTM and Gated Recurrent Unit (GRU) for better accuracy than the previous models. They used the pre-trained word2vec and randomly initiated word embeddings as input for RCNN. GRU and LSTM are further used to capture long-term dependencies. SVM in [2] gives an accuracy of 0.93, whereas using RCNN, the accuracy went to 0.950. Now, for further improvements, using LSTM with RCNN, the accuracy increased a bit and reached 0.9586. RCNN, when used with GRU, gave an accuracy of 0.9667, and finally, the proposed method of RCNN + GRU + Word2Vec gave an optimum accuracy of 0.9776.

S. Kaur et al. [5] proposed a method for textual as well as non-textual texts. They experimented using a two-phase hybrid CNN-LSTM biterm model and performed it on three different datasets. The first dataset was obtained from A. Chakraborty et al. [2], which contains 32,000 headlines from various news articles. The second dataset was taken from Khater et al. [20], which contains 12,000 headlines. The third dataset was created by the authors themselves and primarily contains the headlines from Reddit and Facebook pages. Word embedding is done using GloVe and Word2Vec. LSTM was used to capture the long-term sequences. CNN is used to extract high-level sequences of word features. Data pre-processing is done on non-textual data, where the image is converted into grayscale, noise removal, text extraction, and auto-correction are done, and the pre-processed data is fed into the proposed model. The performance of the proposed model is shown in Table 2.3

In [6], P. Rajapraksha et al. demonstrated a model using transfer learning models like XLNET, BERT, and RoBERTa to detect clickbait. They have taken the training dataset from Webis Clickbait Corpus 2017² and the testing dataset from Kaggle

²<https://webis.de/data/webis-clickbait-17.html>

Table 2.3: Performance of Applied Algorithms.

Approach	Dataset	ROC-AUC	Acc	Prec.	F1-Score	Rec.
Without Pre-trained vectors	Dataset 1	0.87	0.81	0.82	0.81	0.79
	Dataset 2	0.81	0.78	0.80	0.79	0.78
	Dataset 3	0.87	0.84	0.84	0.82	0.80
With Word2vec pre-trained vectors	Dataset 1	0.97	0.93	0.92	0.85	0.80
	Dataset 2	0.89	0.84	0.90	0.85	0.81
	Dataset 3	0.94	0.89	0.91	0.88	0.85
With GloVe pre-trained vectors	Dataset 1	0.99	0.95	0.91	0.88	0.85
	Dataset 2	0.94	0.89	0.95	0.92	0.89
	Dataset 3	0.98	0.94	0.95	0.93	0.91

'Train a clickbait detector'³. They used these three models with different cases by fine-tuning them to categorize whether it was clickbait or not. They used 3 fine tuning strategies, model generalization, model expansion, and model compression. According to their experiments, the results for RoBERTa were far better than BERT and XLNet in many cases.

In [7], B. Gamage et al. used a combination of deep learning models. Their approach is inspired by Zannettou et al. [21]. The idea of the authors is to use the different features of a YouTube video to classify it, and Zannettou et al. [21] focused more on user features such as titles, tags, statistics, and comments, whereas B. Gamage et al. considered an audio transcript along with the above features for evaluation. They used the same dataset as Zannettou et al. [21], which consists of 14000 videos, where 5,049 videos weren't clickbait and 8,591 videos were clickbait. The division of the dataset is done in 81% in training, 9% in validation, and 10% in testing. A multi-model deep learning architecture is built to classify whether the video is clickbait or not. An audio transcript is generated from the YouTube videos using the youtube-transcript API [22]. A separate model is applied to each of the features (title, thumbnail, comments, audio transcript, tags, statistics) and later combined into a single one. After combining, transfer learning is applied to the model, where dense layers were trained and fine-tuning was done throughout the entire model. Their model was successful in obtaining an accuracy of 92.4%.

³<https://www.tira.io/task/clickbait-detection/dataset/clickbait17-test-170720/>

but it cannot be considered the same with different datasets, as a result, the model was not able to give more than 85% of accuracy for different datasets.

In [8], D. Varshney et al. proposed a similar method to B. Gamage et al. [7], but they categorized the features into three categories – Video Content based features, user-profile-based features, and human consensus-based features. Video based features contain speech-title similarity, dislike-like ratio, number of likes, and number of dislikes. In speech-to-title similarity, first, the audio is extracted from the video and an audio transcript is generated, later, the audio transcript and the title are compared using the cosine similarity formula, to find the similarity between them. Human consensus-based features include the number of comments, positive polarity, negative polarity, Fake comment count ratio, etc. User profile features consist of the total number of videos, subscribers-to-age ratio, subscriber count, channel views, and registration age. Various classifiers like SVM, Random Forest, Decision-Tree, SVM, and Logistic Regression were used to experiment, and the Random Forest outperformed them all.

A. A. Balan et al. [9] presented a method using deep learning models, especially recurrent neural networks like Long short-term memory (LSTM). They used 32,000 headlines, including clickbait as well as non-clickbait news. The data was taken from popular websites like 'ViralNova', 'BuzzFeed', 'Scoopwhoop', 'Thatscoop' etc. Preprocessing is done where text is converted to lowercase, punctuation is removed, and stop words are removed. Word2vec word embedding was used to convert text into word vectors. 20% of the dataset was used for testing, and the remaining 80% was used for training. LSTM was trained multiple times to improve efficiency. The most accurate LSTM units were 50 and 50. Different classifiers were compared with LSTM, like Naive Bayes, and LSTM performed better than Naive Bayes.

In [10], S. Regina, K. Purwandari, et al used Natural Language Processing (NLP) methods along with supervised-learning methods like K-Nearest Neighbor, Decision trees, and Bidirectional Long Short-Term Memory.

The dataset was taken from DATA INDONESIA.xlsx. Text preprocessing is done,

and steps like stemming, folding, and tokenization have been performed on the data. Word2Vec is used for text embedding. The dataset was divided into 3:1:1 for training, validation, and testing. When a word or phrase may have many interpretations, bi-LSTM is more beneficial. The performance of Bi-LSTM, KNN, and decision trees is shown in Table 2.4. Out of all the classifiers, Bi-LSTM performed better than others.

Table 2.4: Comparison of Bi-LSTM with Machine Learning Algorithms.

Approach	Acc.	F1-Score	Prec.	Rec.
Bi-LSTM	0.71	0.65	0.67	0.69
KNN	0.577	0.388	0.51	0.313
Decision Tree	0.56	0.49	0.49	0.49

In [11], Y.-W. Ma et al. proposed an approach based on feature engineering and artificial intelligence. They had done data collection, feature extraction, text preprocessing, feature evaluation, etc. Their model used 18 format-based or lexicon features including title starts with number, title of exclamation, Number of inputs, Number of tags, etc. They have taken a dataset containing 6,000 non-clickbait and 6,000 clickbait headlines. A hybrid model is built using a neural network schema. CNN and LSTM are used to improve accuracy. Softmax activation function was used in the model, and an input layer and 100-D vectors of the title text are used in the embedding process. ANOVA is used for feature selection and extraction. They were able to get to a precision of 93.25% and an accuracy of 88.5%.

Table 2.5: Summary of existing models

Author Name	Approach	Result
Amol Agrawal et al.	Word2vec and CNN with one layer of convolution	Accuracy = 0.90, F1-Score = 0.86 Recall = 0.88, ROC-AUC = 0.90 Precision = 0.85,
A. Chakraborty et al.	Features Selected - Sentence Structure, Clickbait Language, N-gram features, Word Patterns, Models used - SVM, Decision Tree, Random Forest	Accuracy = 0.93, ROC-AUC = 0.97 Precision = 0.92, F1-Score = 0.934 Recall = 0.95,
A. Geçkil et al.	Features Selected - Meta Information, Teaser Message(N-grams,bag of words) Models Used - Naive Bayes, Random Forest, Logistic Regression	F1-Score = 0.920 Precision = 0.899, Accuracy = 0.865, Recall = 0.941,
S. Chawda et al.	Recurrent Convolution Neural Network (RCNN) + Gated Recurrent Unit (GRU) + Word2Vec	Accuracy = 0.9776
S. Kaur et al.	A Hybrid CNN-LSTM Biterm model with Glove and Word2vec pre-trained vectors and also without pre-trained vectors	Table - 2.3
P. Rajapaksha et al.	Transfer Learning Models like BERT, RoBERTa and XLNet with fine-tuning with different cases	Precision = 0.73, Accuracy = 0.85, F1-Score = 0.69 Recall = 0.87,
B. Gamage et al.	Features selected - Title, comments, audio transcript, Tags, Statistics Deep Learning Models - LSTM, CNN	Accuracy = 0.85
D. Varshney et al.	Features selected - Title, comments, audio transcript, Tags, Statistics Model Used - SVM, Random Forest, Decision-Tree, logistic-Regression	Precision = 0.84, F1-Score = 0.77 Recall = 0.78,
A. A. Balan et al.	Word Embedding using Word2Vec, Model Used - LSTM, Naive Bayes	Precision = 0.96, Accuracy = 0.96, F1-Score = 0.96 Recall = 0.96,
S. Regina et al.	Word Embedding using Word2Vec, Models Used- Bi-LSTM, KNN, Decision Tree	Table - 2.4
Y.-W. Ma et al.	Lexicon and format-based features Model Used - CNN-LSTM	Accuracy = 0.88, F1-Score = 0.98 Recall = 0.98, Precision = 0.93,

CHAPTER 3

METHODOLOGY

The methodology for determining whether the headlines are clickbait or not consists of four phases, firstly the dataset creation, secondly the data preprocessing, then text processing, and applying our different machine learning and deep learning models, out of all Bi-LSTM performed well.

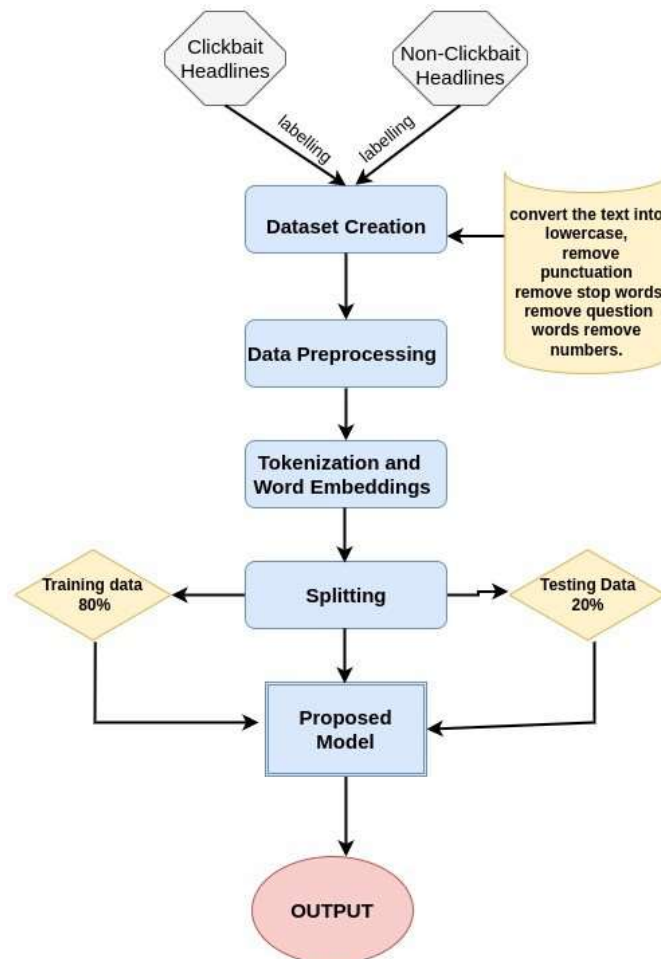


Figure 3.1: Proposed Model Flow

3.1 Dataset Creation

The Dataset contains almost 52000 headlines containing 27000 clickbait, whereas 25000 non-clickbait headlines are from multiple websites and the headlines are taken from 2007 till March 2023. For the 2007-2016 timeline, the dataset is taken from Kaggle[18], the Kaggle dataset contains almost 30,000 headlines both clickbait and non-clickbait.

For the 2019-2023 timeline, the headlines are scrapped from multiple sources like Twitter and other News publications. Clickbait headlines are scrapped from Twitter handles like BuzzFeed, ViralNova, Thatscoop, and The Odyssey. Non-Clickbait headlines are scrapped from online news apis like The Guardian ¹, BBC News, Bloomberg, Reuters ², The Washington Post ³, NY Times ⁴.

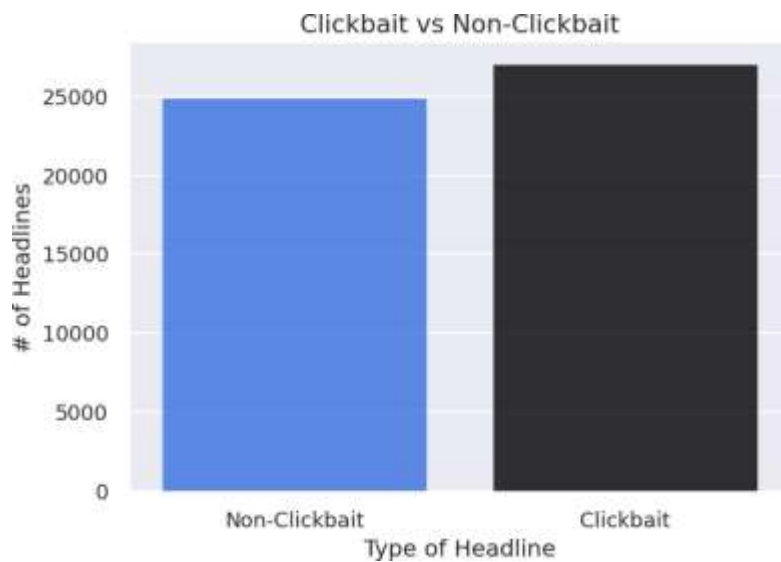


Figure 3.2: Histogram of Created Dataset: Clickbait and Non-Clickbait Headlines

¹<https://open-platform.theguardian.com/>

²<https://newsapi.org/>

³<https://www.washingtonpost.com>

⁴<https://api.nytimes.com/>



Figure 3.3: Count of Number of Words in a Headline

3.2 Data Preprocessing

After collecting headlines from all the sources, we will merge all the headlines into one CSV file, The dataset contains columns like text and labels. Preprocessing is done before actually processing the data, so in preprocessing, we will convert the text into lowercase, remove punctuation, remove words containing numbers, remove text in square brackets, remove question words, remove stop words, and remove numbers.

```
#make text lowercase
df['text']=df['text'].apply(lambda x: x.lower())
```

Figure 3.4: Converting Text into Lowercase

```

#function to remove punctuation and non-alphabetical characters and links
import re
def clean_text_round1(text):
    '''Make text lowercase, remove text in square brackets, remove punctuation and remove words containing numbers.'''
    text = text.lower()
    #text = re.sub('[\w\d]+', '', text)
    text = re.sub('\n', '', text)
    text = re.sub(' ', '', text)
    text = re.sub(r'^https?:\/\/.*[\r\n]*', '', text, flags=re.MULTILINE)
    text = re.sub(r'\w+:\/\/{2}[\d\w-]+\.[\d\w-]+(?:\.[\d\w-]+)+$', '', text)
    text = re.sub('[\.\*\?]', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub("'", '', text)
    text = re.sub('"', '', text)
    text = re.sub('`', '', text)
    text = re.sub('-', '', text)
    text = re.sub('_', '', text)
    return text

```

Figure 3.5: Data Preprocessing

3.3 Text Processing

In Text Processing, We have done tokenization of Data, where each word is represented as a token and to perform Exploratory data analysis (EDA), we have done featured engineering by finding the most clickbait words, most non-clickbait words, number of unique words in each class, headlines start with a number or not, headlines contain exclamation mark. For modeling, TFIDF scores were accessed for each unigram and bigram.

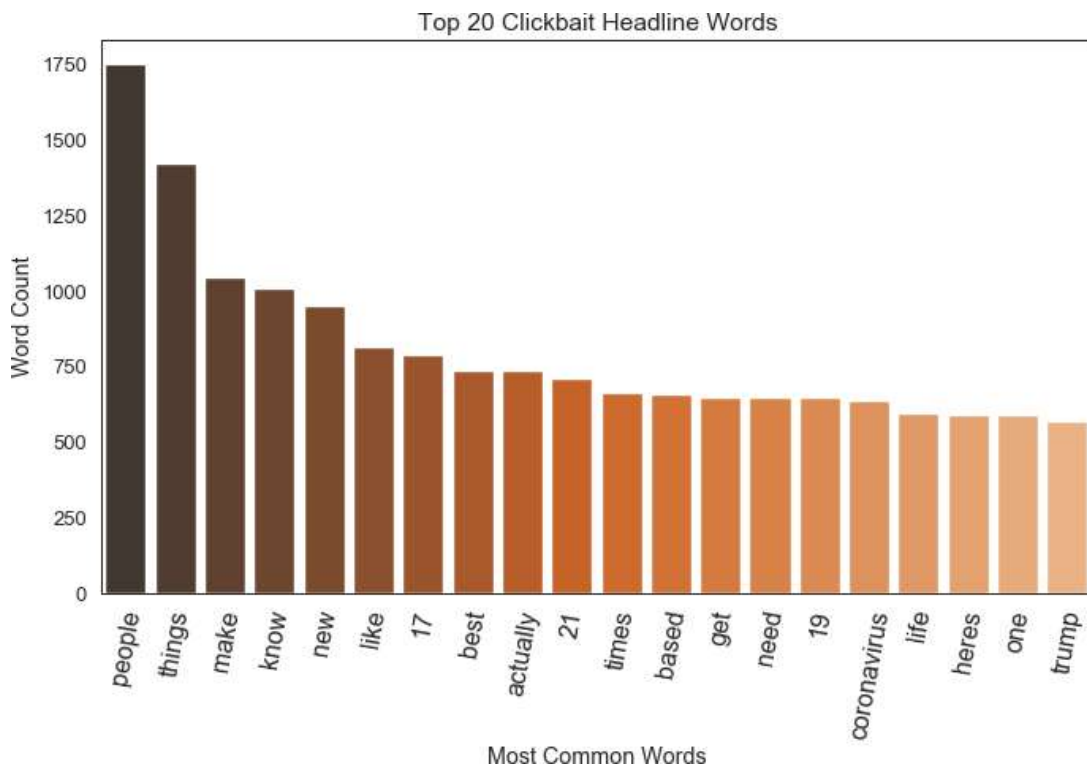


Figure 3.6: Top 20 Clickbait Headline Words

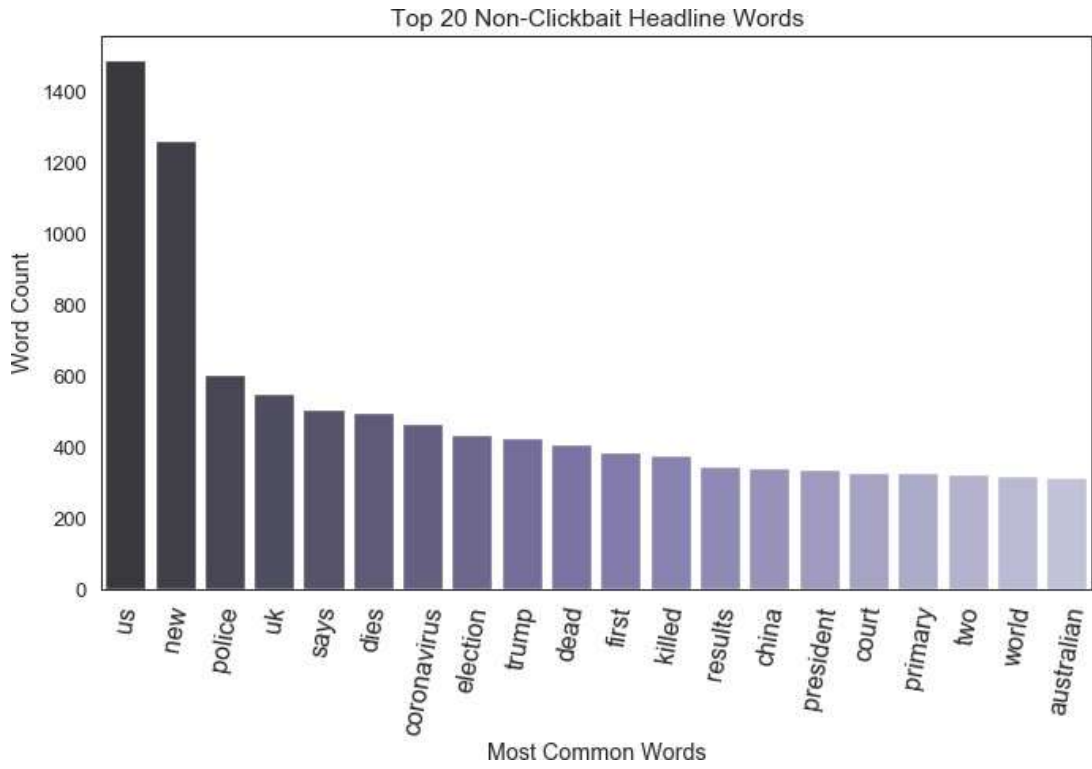


Figure 3.7: Top 20 Non-Clickbait Headline Words

3.4 Machine Learning Models

3.4.1 Naive Bayes

```
#Naive Bayes Classifier
nb_classifier = MultinomialNB(alpha = .05)

nb_classifier.fit(X_train, y_train)

nb_train_preds = nb_classifier.predict(X_train)
nb_test_preds = nb_classifier.predict(X_test)

print(train_results(nb_train_preds))
print(test_results(nb_test_preds))

('Training Accuracy:', 0.9986699066376774, ' Training Recall:', 0.9992099155597255)
('Testing Accuracy:', 0.9300184162062615, ' Testing Recall:', 0.9416336706261915)
```

Figure 3.8: Naive Bayes Classifier

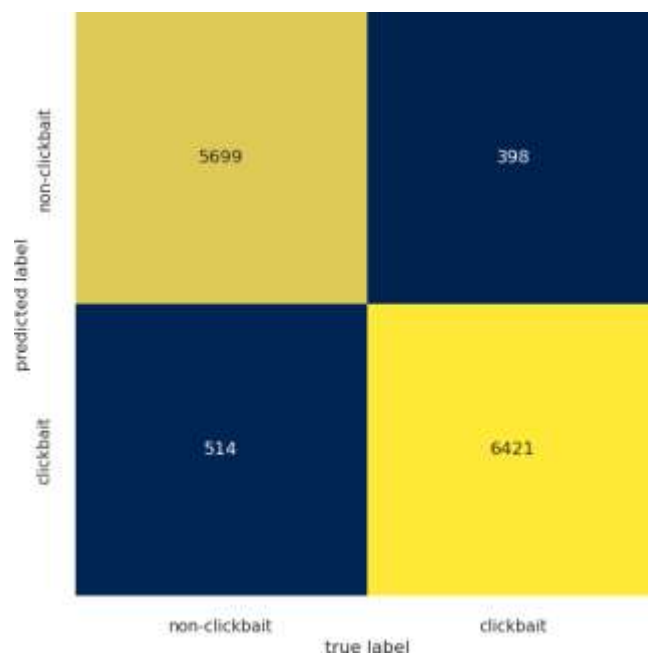


Figure 3.9: Confusion Matrix of Naive Bayes Classifier

3.4.2 Random Forest

```
#Random Forest
rf_classifier = RandomForestClassifier(class_weight = 'balanced', n_estimators=900, )
rf_classifier.fit(X_train, y_train)

rf_test_preds = rf_classifier.predict(X_test)
rf_train_preds = rf_classifier.predict(X_train)

print(train_results(rf_train_preds))
print(test_results(rf_test_preds))

('Training Accuracy:', 1.0, ' Training Recall:', 1.0)
('Testing Accuracy:', 0.9070748925721301, ' Testing Recall:', 0.9338612699809357)
```

Figure 3.10: Random Forest Classifier

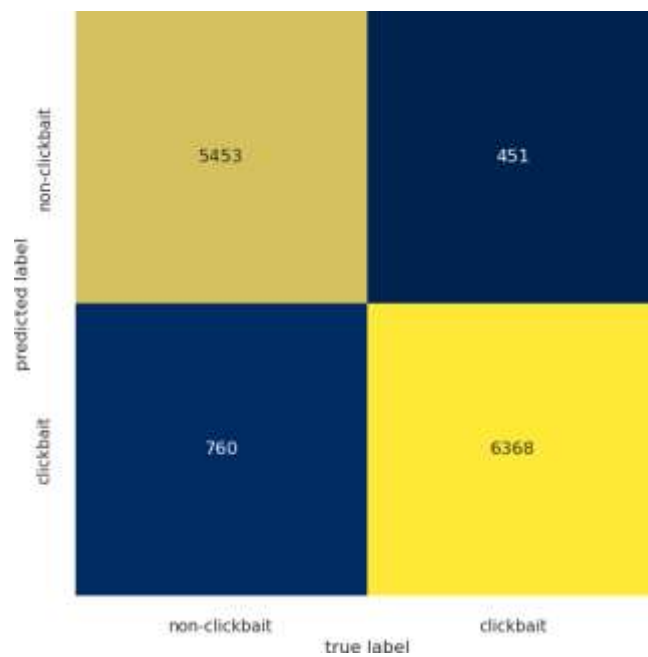


Figure 3.11: Confusion Matrix of Random Forest

3.4.3 Support Vector Machine (SVM)

```
#SVM Model

svm_classifier = LinearSVC(class_weight='balanced', C=10, max_iter = 1500 )
svm_classifier.fit(X_train, y_train)

svm_test_preds = svm_classifier.predict(X_test)
svm_train_preds = svm_classifier.predict(X_train)

print(train_results(svm_train_preds))
print(test_results(svm_test_preds))

('Training Accuracy:', 0.999872106407469, ' Training Recall:', 0.9997530986124142)
('Testing Accuracy:', 0.9323204419889503, ' Testing Recall:', 0.9260888693356797)
```

Figure 3.12: Support Vector Machine

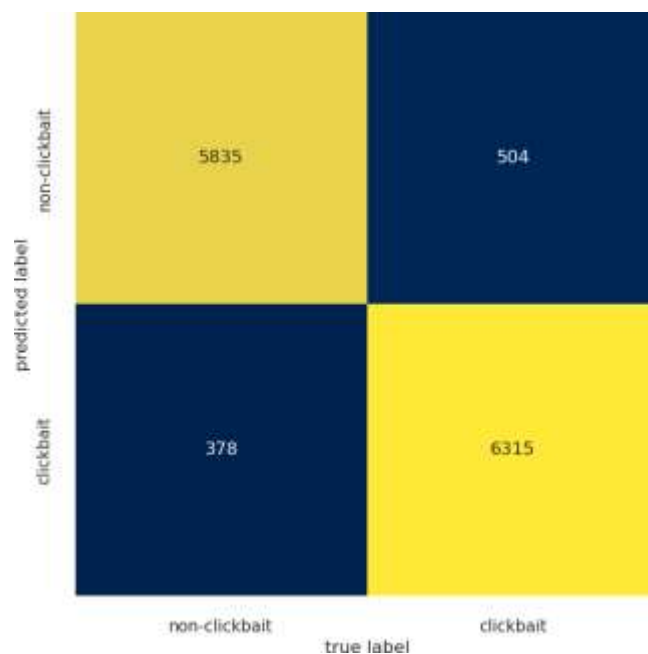


Figure 3.13: Confusion Matrix of Support Vector Machine

3.4.4 Logistic Regression

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=500, class_weight = 'balanced', solver = 'liblinear', tol=0.0001)

lr.fit(X_train,y_train)

lr_train_preds = lr.predict(X_train)
lr_test_preds = lr.predict(X_test)

print(train_results(lr_train_preds))
print(test_results(lr_test_preds))

#plot confusion matrix on test set lr Classifier
sns.set()

cm_dc = confusion_matrix(y_test, lr_test_preds)
sns.heatmap(cm_dc.T, square=True, annot=True, fmt='d', cbar=False,cmap="cividis",
            xticklabels=['non-clickbait','clickbait'],yticklabels=['non-clickbait','clickbait']
            )
plt.xlabel('true label')
plt.ylabel('predicted label');
plt.savefig('lr_cm')
```

Figure 3.14: Logistic Regression

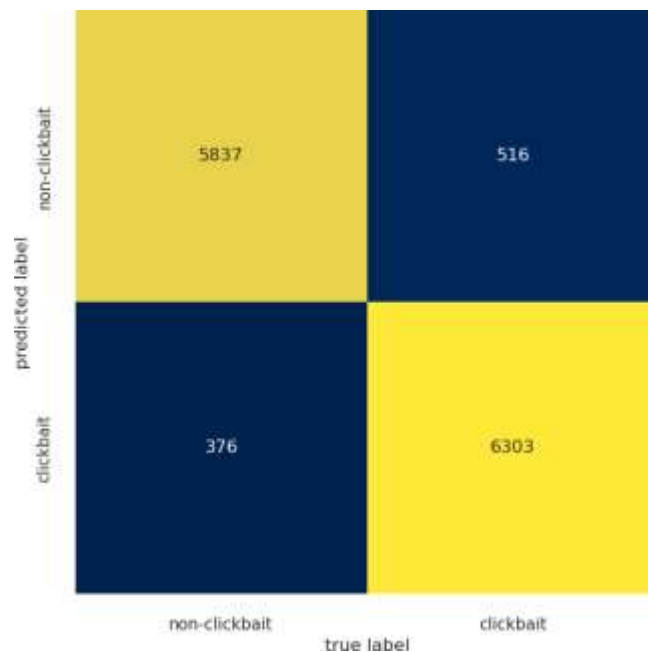


Figure 3.15: Confusion Matrix of Logistic Regression

3.4.5 XGBoost

```
from xgboost import XGBClassifier

xgb_clf = XGBClassifier()
xgb_clf.fit(X_train, y_train)
xgb_test_preds = xgb_clf.predict(X_test)
xgb_train_preds = xgb_clf.predict(X_train)

print(test_results(xgb_test_preds))
print(train_results(xgb_train_preds))

#plot confusion matrix on test set xgboost Classifier
sns.set()

cm_dc = confusion_matrix(y_test, xgb_test_preds)
sns.heatmap(cm_dc.T, square=True, annot=True, fmt='d', cbar=False, cmap="cividis",
            xticklabels=['non-clickbait', 'clickbait'], yticklabels=['non-clickbait', 'clickbait']
            )
plt.xlabel('true label')
plt.ylabel('predicted label');

('Testing Accuracy:', 0.8566605279312461, ' Testing Recall:', 0.8558439653908197)
('Training Accuracy:', 0.8834122010487274, ' Training Recall:', 0.8654387437657399)
```

Figure 3.16: XGBoost

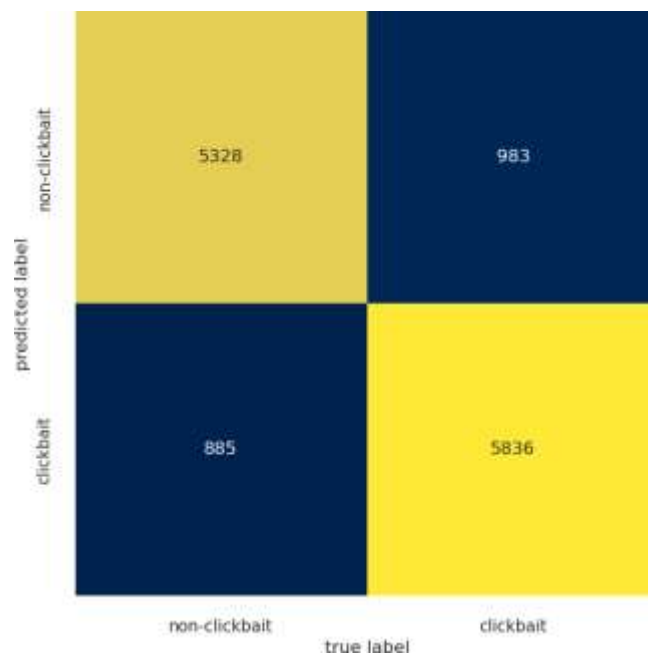
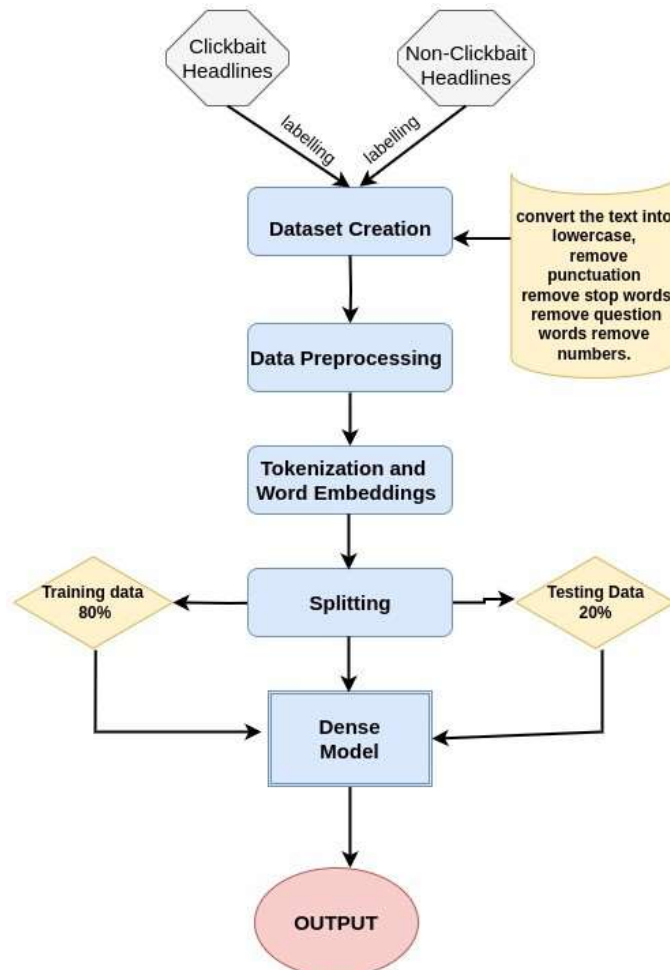


Figure 3.17: Confusion Matrix of XGBoost

3.5 Deep Learning Models

3.5.1 Dense Model



```
# Define parameter
embedding_dim = 16
drop_value = 0.2

# Define Dense Model Architecture
model = Sequential()
model.add(Embedding(vocab_size,
                    embedding_dim,
                    input_length = max_len))
model.add(GlobalAveragePooling1D())
model.add(Dense(24, activation='relu'))
model.add(Dropout(drop_value))
model.add(Dense(1, activation='sigmoid'))
```

Figure 3.18: Dense Model

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 100, 16)	8000
global_average_pooling1d (GlobalAveragePooling1D)	(None, 16)	0
dense (Dense)	(None, 24)	408
dropout (Dropout)	(None, 24)	0
dense_1 (Dense)	(None, 1)	25

Total params: 8,433
 Trainable params: 8,433
 Non-trainable params: 0

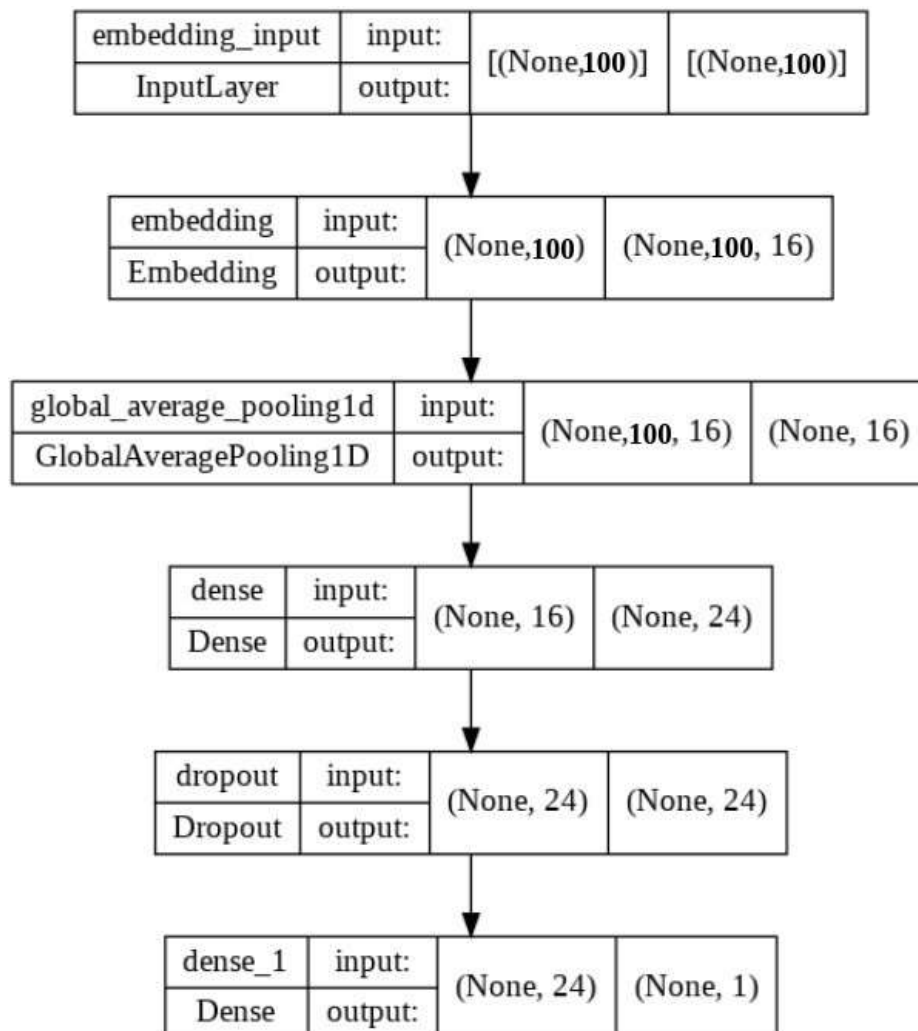


Figure 3.19: Summary of the Dense model

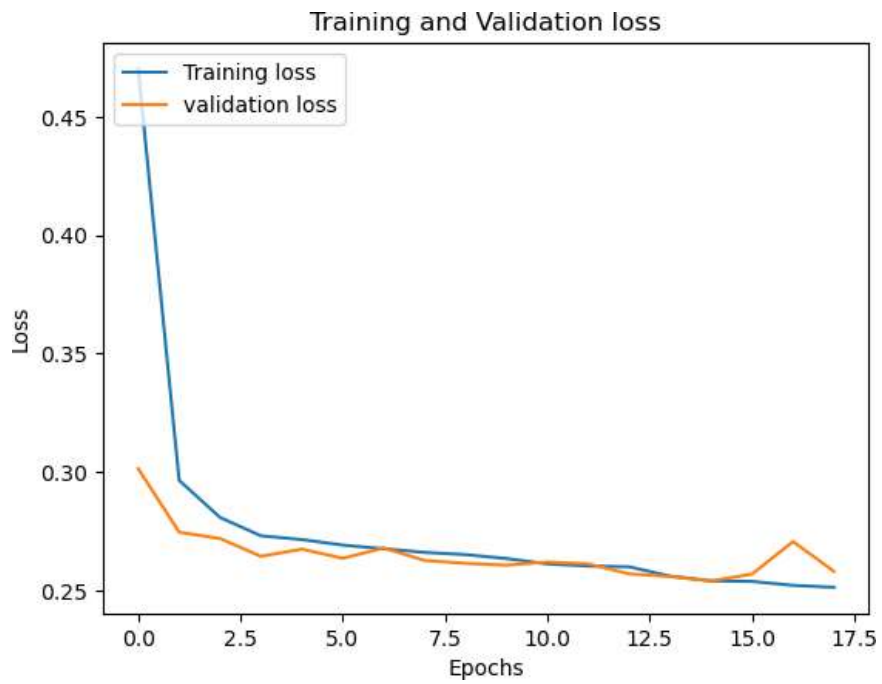


Figure 3.20: Training and Validation loss of Dense Model

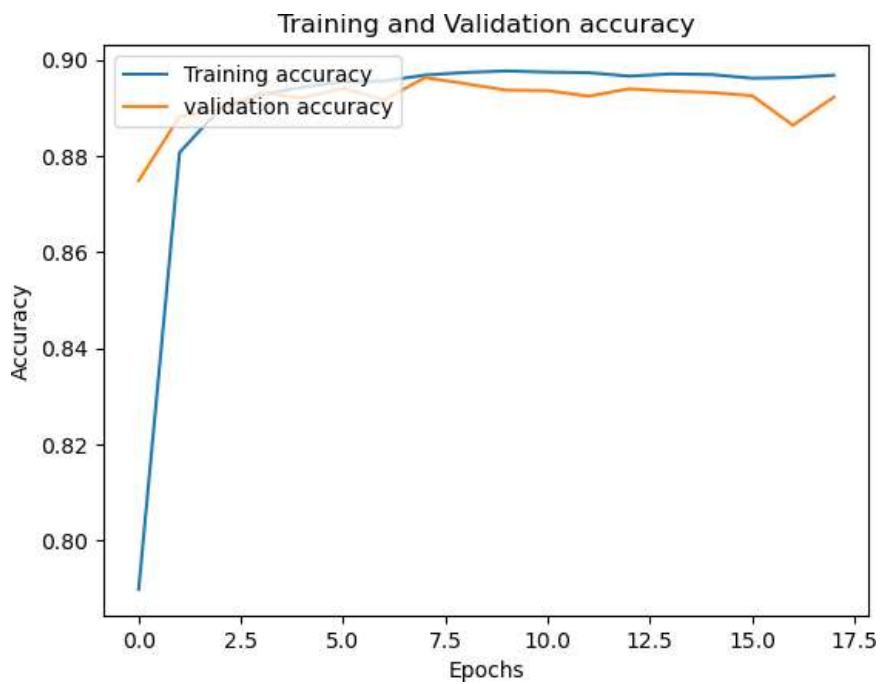


Figure 3.21: Training and Validation accuracy of Dense Model

3.5.2 Long Short Term Memory

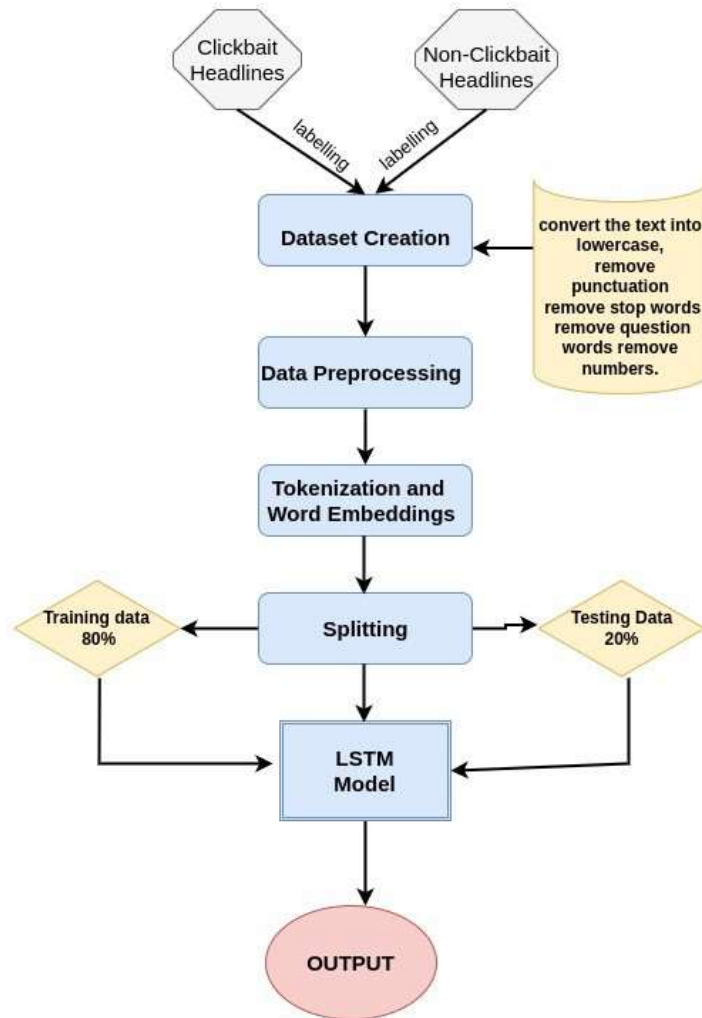


Figure 3.22: Long Short Term Memory Model Flow

```
# Define parameter
n_lstm = 128
drop_lstm = 0.2
# Define LSTM Model
model1 = Sequential()
model1.add(Embedding(vocab_size, embedding_dim, input_length=max_len))
model1.add(SpatialDropout1D(drop_lstm))
model1.add(LSTM(n_lstm, return_sequences=False))
model1.add(Dropout(drop_lstm))
model1.add(Dense(1, activation='sigmoid'))
```

Figure 3.23: Long Short Term Memory Model

Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 100, 16)	8000
spatial_dropout1d (SpatialD ropout1D)	(None, 100, 16)	0
lstm (LSTM)	(None, 128)	74240
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 1)	129

=====
 Total params: 82,369
 Trainable params: 82,369
 Non-trainable params: 0
 =====

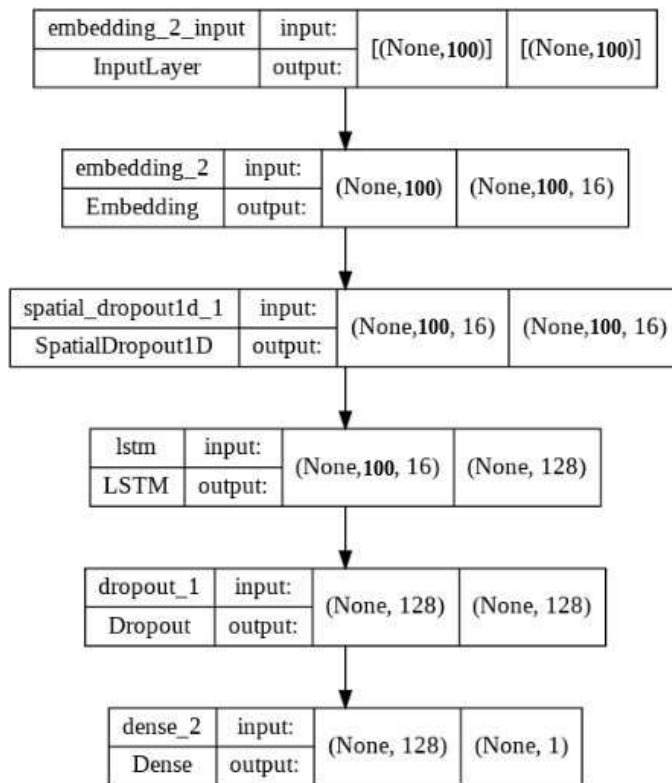


Figure 3.24: Summary of the Long Short Term Memory Model

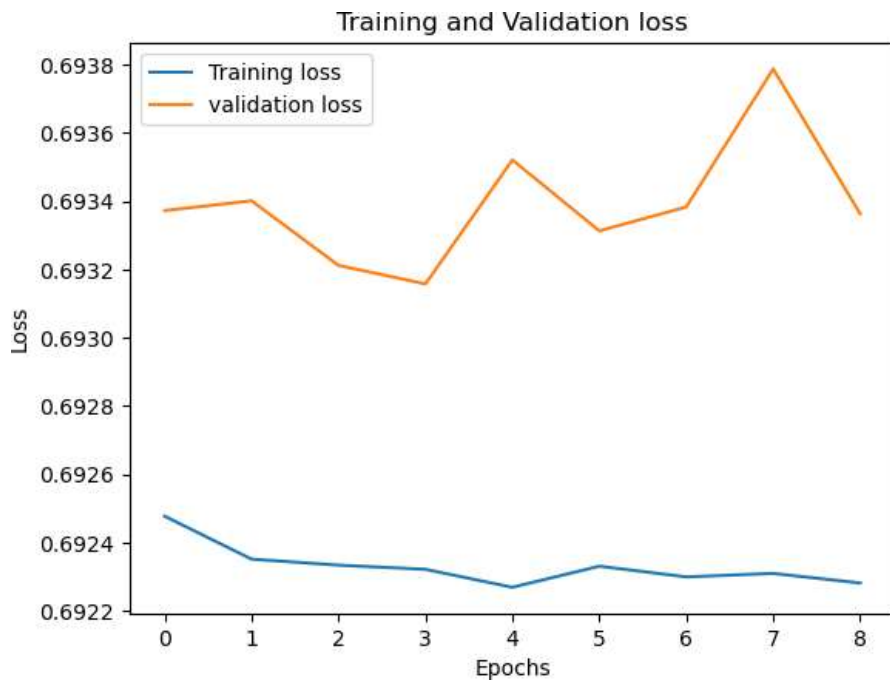


Figure 3.25: Training and Validation loss of LSTM

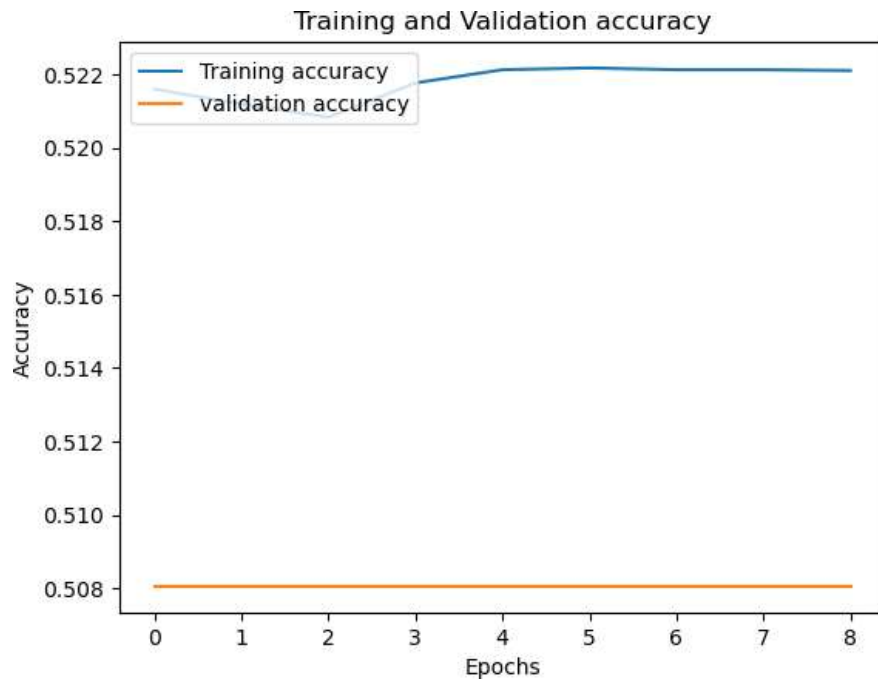


Figure 3.26: Training and Validation accuracy of LSTM

3.5.3 Gated Recurrent Unit

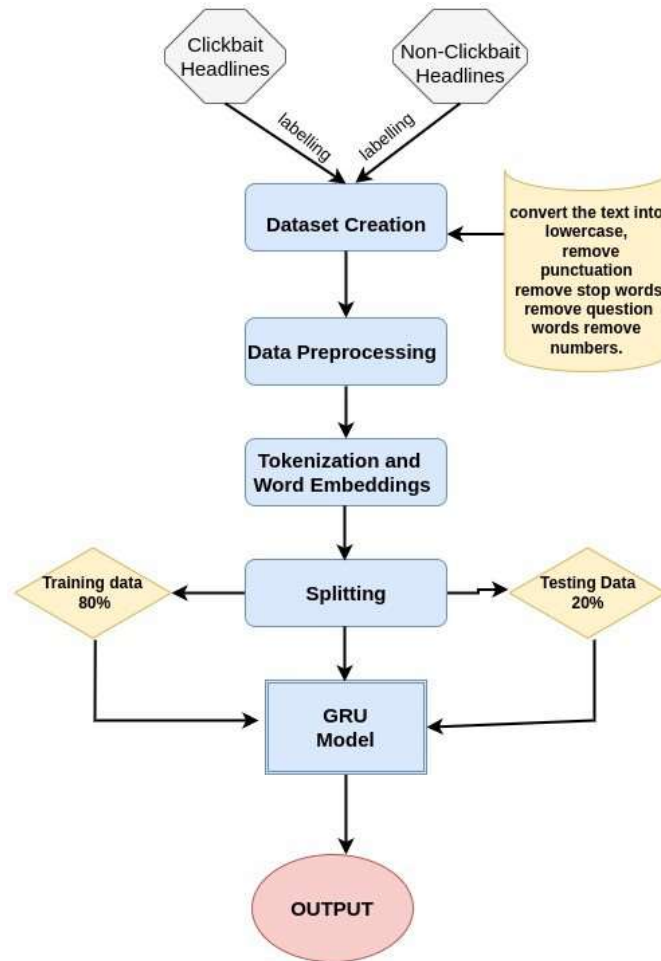


Figure 3.27: Gated Recurrent Unit Model Flow

```
#GRU
model3 = Sequential()
model3.add(Embedding(vocab_size,
                    embedding_dim,
                    input_length = max_len))
model3.add(SpatialDropout1D(0.2))
model3.add(GRU(128, return_sequences = False))
model3.add(Dropout(0.2))
model3.add(Dense(1, activation = 'sigmoid'))
```

Figure 3.28: Gated Recurrent Unit Model

Model: "sequential_10"

Layer (type)	Output Shape	Param #
embedding_10 (Embedding)	(None, 100, 16)	8000
spatial_dropout1d_2 (SpatialDropout1D)	(None, 100, 16)	0
gru_1 (GRU)	(None, 128)	56064
dropout_10 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 1)	129

=====
 Total params: 64,193
 Trainable params: 64,193
 Non-trainable params: 0
 =====

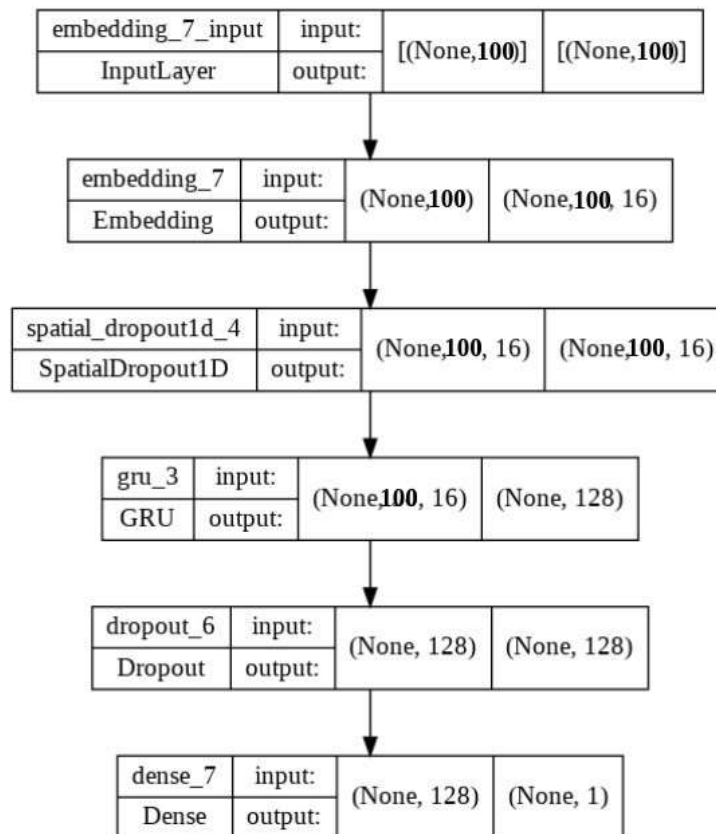


Figure 3.29: Summary of the Gated Recurrent Unit Model



Figure 3.30: Training and Validation loss of GRU

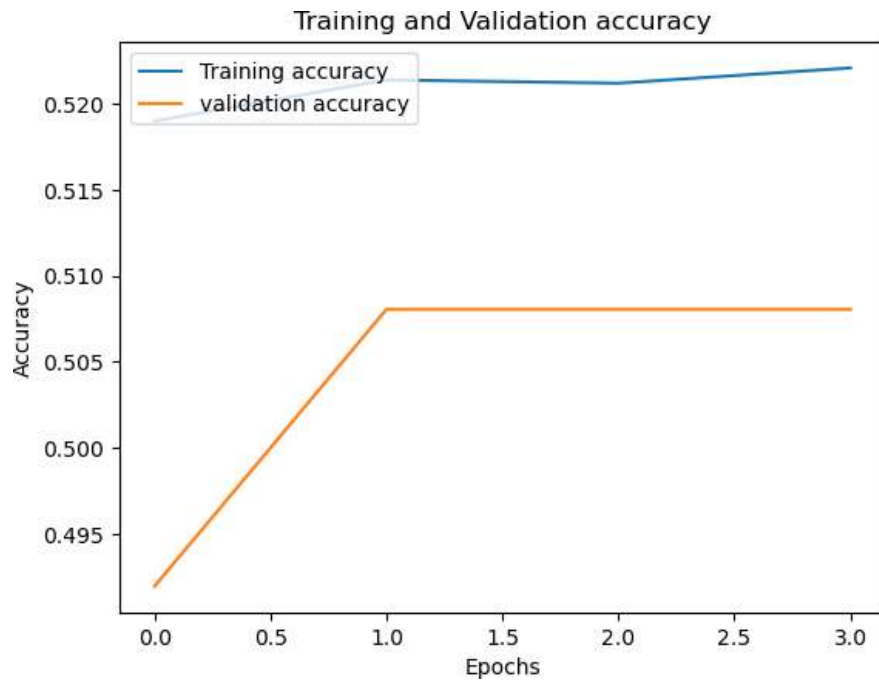


Figure 3.31: Training and Validation accuracy of GRU

3.5.4 Bidirectional Long Short Term Memory Model:

The Bi-LSTM Model consists of 4 layers: The embedding layer, which is the first layer of the Bi-LSTM Model, takes a sentence as input and produces as output a vector representation of the sentence where each word is replaced by a vector of numbers. This vector of numbers indicates the context of the word and how closely it is related to or not related to some other words. The Bi-LSTM layer, the second layer of the model, takes as input a vector-of-vectors representation of a sentence and produces as output a vector of numbers. The dropout layer, the third layer in the model, works towards reducing or delaying overfitting in a neural network by randomly turning off some percentage of units during the training phase of the deep learning model. The final layer of the neural network is the dense layer which takes as input a vector of numbers and produces as output the predicted class for the given input sentence.

Table 3.1: Implementation Details and Layer-wise Parameters of Bi-LSTM Model

Layer	Parameter Name	Parameter Value
Embedding Layer	Vocab Size	500
	Max Length	100
	Embedding Dim	16
Bi-LSTM Layer	Units	128
Dropout Layer	Drop Rate	20%
Dense Layer	Units	1
	Activation Function	Sigmoid

In Table- 3.1, the Implementation details of Bi-LSTM are given, we have used sigmoid as Activation Function.

In Table 3.2, all the parameters we have used while training is given with their values. We are using Adam Optimizer. We are doing early stopping for validation loss.

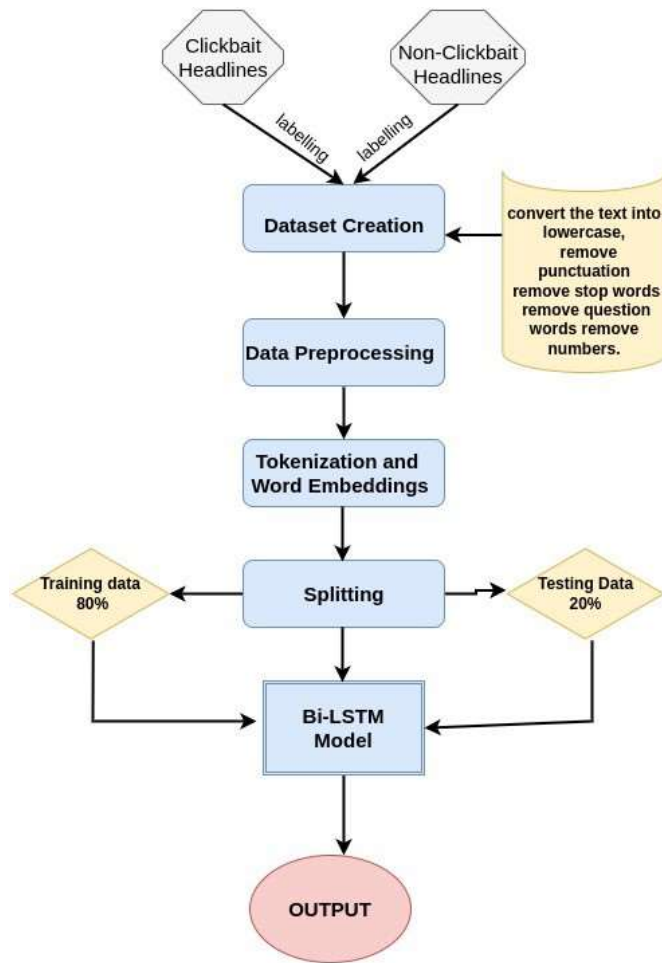


Figure 3.32: Bidirectional Long Short Term Memory Model Flow

```

#Bi-directional LSTM
model2 = Sequential()
model2.add(Embedding(vocab_size,
                    embedding_dim,
                    input_length = max_len))
model2.add(Bidirectional(LSTM(n_lstm,
                             return_sequences = False)))
model2.add(Dropout(drop_lstm))
model2.add(Dense(1, activation='sigmoid'))
  
```

Figure 3.33: Bidirectional Long Short Term Memory Model

Model: "sequential_9"

Layer (type)	Output Shape	Param #
embedding_9 (Embedding)	(None, 100, 16)	8000
bidirectional_6 (Bidirectional)	(None, 256)	148480
dropout_9 (Dropout)	(None, 256)	0
dense_10 (Dense)	(None, 1)	257

=====
 Total params: 156,737
 Trainable params: 156,737
 Non-trainable params: 0

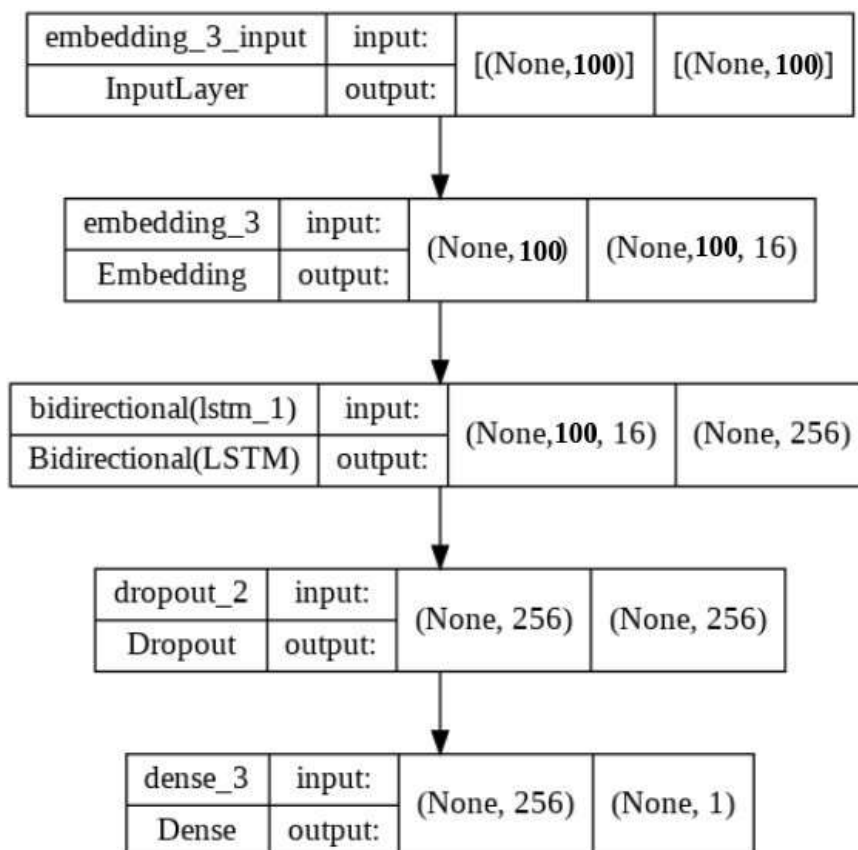


Figure 3.34: Summary of the Bidirectional Long Short Term Memory Model



Figure 3.35: Training and Validation loss of Bi-LSTM

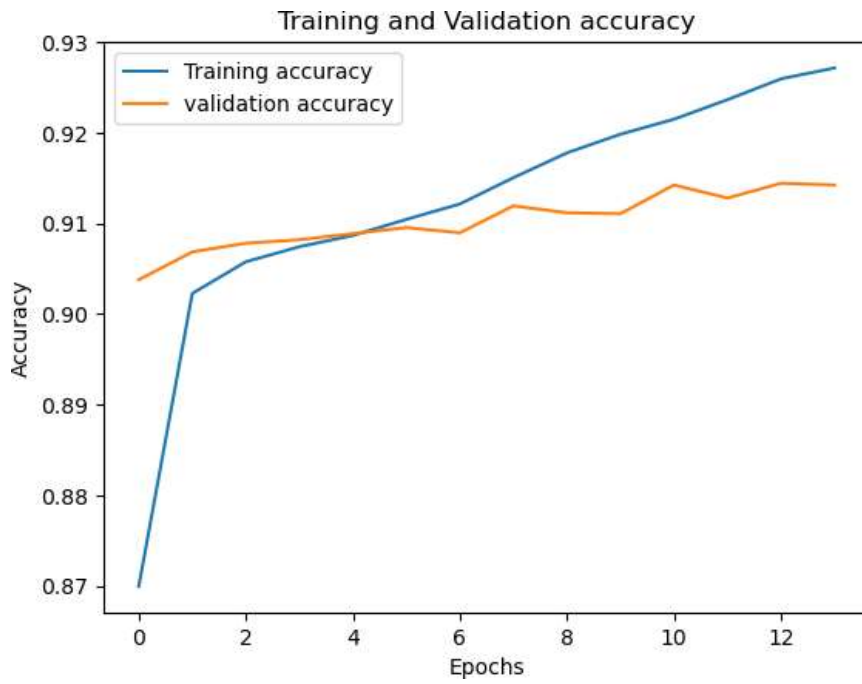


Figure 3.36: Training and Validation accuracy of Bi-LSTM

Table 3.2: Parameters used to Train the model

Parameter Name	Parameter Value
Loss Function	Binary Cross entropy
Optimizer	Adam
Metrics	F1 Score, Precision Recall, Accuracy
Number of Epochs	500
Early Stopping Pa- tience	3

In Table - 3.3, the parameters used for data processing are given. We are splitting the data in 80:20, 80% for the training set, and 20% for testing.

Table 3.3: Parameters for Data Preprocessing

Parameter Name	Parameter Value
Train Test Split	80:20:00
Max Length	100
Padding Type	Post
Truncation Type	Post
Vocab Size	500
Number of Unique Words	30757

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Experimental Setup

The models are implemented in Python, codes are written in jupyter notebook and run on a system having an Intel i5 11th gen processor with RAM of 16 GB and an operating system of Red Hat Linux.

4.2 Evaluation Metrics

In machine learning, evaluation metrics are used to determine a model's performance. To determine the performance of our models, we utilize classification metrics like F1 score, recall, precision, and accuracy. We can measure the performance using a confusion matrix. It is a matrix of 2 * 2 table, for 2 class classifiers.

		Actual class	
		1	0
Predicted class	1	True Positive	False Positive
	0	False Negative	True Negative

Figure 4.1: Confusion Matrix

- True Positive (TP): Our model predicted class ‘clickbait’ and the actual class is ‘clickbait’.
- True Negative (TN): Our model predicted class ‘non-clickbait’ and the actual class is ‘non-clickbait’.
- False Positive (FP): Our model predicted class ‘clickbait’ but the actual class is ‘non-clickbait’.
- False Negative (FN): Our model predicted class ‘non-clickbait’, but the actual class is ‘clickbait’.

4.3 Training and Validation Performance Curves

We have plotted the curves of Training and Validation accuracy and loss of our Bi-direction LSTM model. Training loss evaluates how well our model matches the training set of data, and Validation loss evaluates how well it performs on the validation set. The purpose of this curve is to understand, which part needs tuning, there can be underfitting or overfitting problems, which can arise in our model. Usually in Bidirectional LSTM, an Overfitting problem arises. In Overfitting, our models perform better on training data but perform poorly on validation data, So validation loss starts increasing.

4.4 Result Analysis

Our Model aims to classify the clickbait headlines using Bidirectional LSTM. The results of BI-LSTM along with other Deep Learning Models like the Dense Model, LSTM Model, and GRU is shown in Table 4.2, Out of all learning models applied BI-LSTM outperforms with an accuracy of 93%. We have also applied machine learning algorithms to our created dataset. In Machine Learning algorithms, SVM performed the best with an accuracy of 93%, but the recall value of BI-LSTM is

better than SVM.

Table 4.1: Performance of Machine Learning Approaches.

Approach	Acc.	F1-Score	Prec.	Rec.
Naive Bayes	0.930	0.934	0.926	0.942
Logistic Regression	0.931	0.934	0.944	0.924
SVM	0.932	0.935	0.944	0.926
Random Forest	0.906	0.912	0.893	0.933
XGBoost	0.857	0.862	0.868	0.856

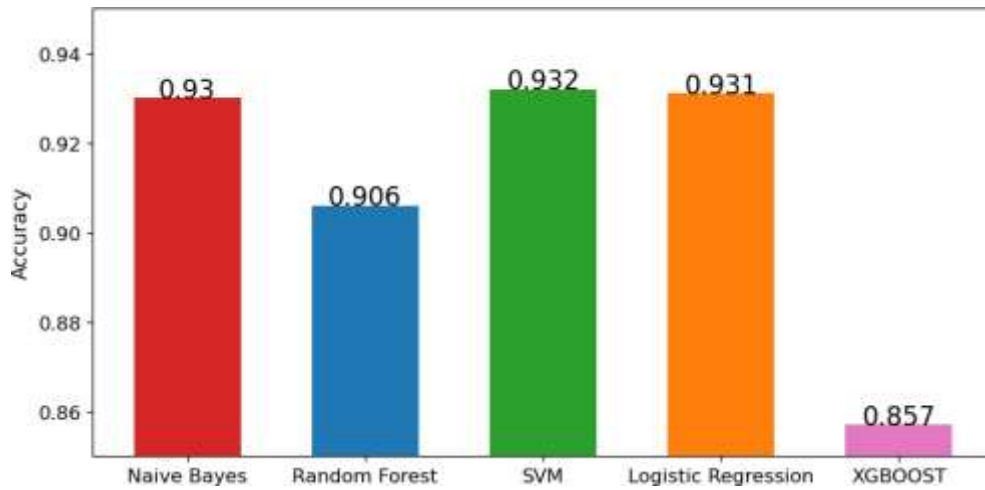


Figure 4.2: Comparison of Machine Learning Models

As We can see in Table 4.1, SVM has a higher F1-score, precision, and accuracy than other classifiers, and have an equivalent Recall value as others. Naive Bayes have maximum recall and accuracy. Support Vector Machine (SVM) performs better than Random Forest, the accuracy of SVM is slightly better than Naive Bayes but recall is slightly worse. Random Forest overfitting with training dataset and recall and accuracy of Random Forest were lower than Naive Bayes. The performance of Logistic Regression is very close to SVM, but recall is slightly lower than SVM, and At last, XGBoost Performs worst of all models. The accuracy of each machine learning model used is represented by a bar graph in Figure 4.2.

In Table 4.2, We can observe the outcomes of using deep learning models, Dense Model has good accuracy of 90%, and The binary cross-entropy has been utilized

Table 4.2: Performance of Deep learning approaches.

Approach	Acc.	F1-Score	Prec.	Rec.
Gated Recurrent Unit (GRU)	0.521	0.678	0.522	1
Long Short Term Memory	0.522	0.681	0.522	1
Dense Model	0.908	0.908	0.926	1
Bidirectional Long Short Term Memory (Bi-LSTM)	0.931	0.931	0.942	1

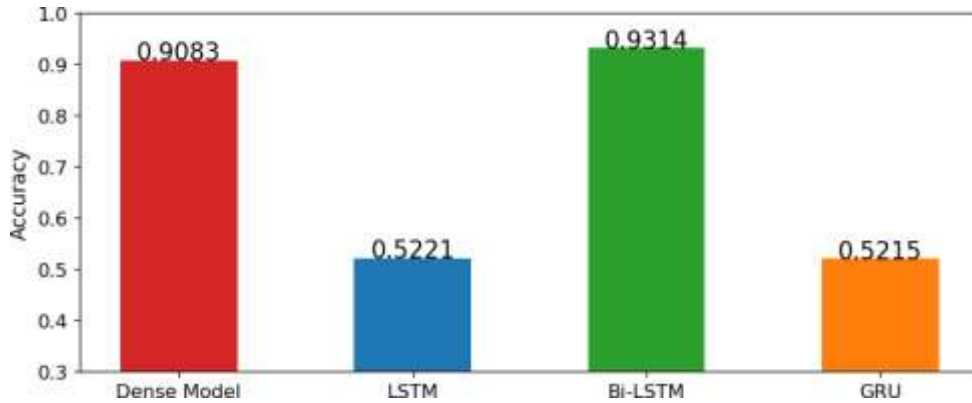


Figure 4.3: Comparison of Deep Learning Models

as a loss function. We have utilized Adam as an optimizer, which uses momentum to prevent local minima. The F1-Score, recall, and precision of the Dense model was good. The performance of LSTM was not good in our experiment, the accuracy of LSTM was just 52%, and other metrics were also not up to the mark. GRU's performance was not good. Out of all deep learning models, BI-LSTM outperformed with an accuracy of 93%, recall of 1, precision of 94%, and F1-score of 93%.

4.5 Handling Overfitting

There can be multiple ways to avoid Overfitting: increasing the training dataset, cross-validation, Adding dropout layers, and early stopping. In this model, we have used early stopping. Early stopping is a halt in the training of the data when it seems like the model will not learn anything new. We have used validation loss as a parameter for early stopping, and patience with the limit 3, i.e. it will continue until the next 3 epochs if the validation loss stops decreasing.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

Clickbait detection is a really difficult task, as in a world full of the internet, a lot of news headlines float from here to there, and sometimes it's very difficult to find the origin of the news as it takes time to actually detect whether it is confirmed news or not. That's why there is no fixed dataset used in the various models included in our study because news and headlines keep on changing, and publishers are getting more and more intelligent after every such detection model, so we need to keep improvising our models with a different dataset. In this study, we reviewed the existing models in the domain of clickbait detection and observed that machine learning algorithms have a very significant role in clickbait detection and that NLP tools are really important to understand the context of the data. In the future, clickbait detection can be identified by browser extensions as well as mobile applications. Until now, the experiment was performed on datasets with only a few languages; in the future, larger datasets with multiple languages will be collected by news websites and social media sites. Future work also includes (1) finding important features that are more useful, (2) using the latest word embedding techniques, and (3) clickbait detection for thumbnails and video frames for video clickbait detection.

REFERENCES

- [1] A. Agrawal, ‘Clickbait detection using deep learning’, in 2016 2nd international conference on next generation computing technologies (NGCT), 2016, pp. 268–272.
- [2] A. Chakraborty, B. Paranjape, S. Kakarla, and N. Ganguly, ‘Stop clickbait: Detecting and preventing clickbaits in online news media’, in 2016 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM), 2016, pp. 9–16.
- [3] A. Geçkil, A. A. Müngen, E. Gündogan, and M. Kaya, ‘A clickbait detection method on news sites’, in 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2018, pp. 932–937.
- [4] S. Chawda, A. Patil, A. Singh, and A. Save, ‘A novel approach for clickbait detection’, in 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 2019, pp. 1318–1321.
- [5] S. Kaur, P. Kumar, and P. Kumaraguru, ‘Detecting clickbaits using two-phase hybrid CNN-LSTM biterm model’, *Expert Systems with Applications*, vol. 151, p. 113350, 2020.
- [6] P. Rajapaksha, R. Farahbakhsh, and N. Crespi, ‘BERT, XLNet or RoBERTa: The Best Transfer Learning Model to Detect Clickbaits’, *IEEE Access*, vol. 9, pp. 154704–154716, 2021.
- [7] B. Gamage, A. Labib, A. Joomun, C. H. Lim, and K. Wong, ‘Baitradar: A Multi-Model Clickbait Detection Algorithm Using Deep Learning’, in ICASSP

- 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 2665–2669.
- [8] D. Varshney and D. K. Vishwakarma, ‘A unified approach for detection of Clickbait videos on YouTube using cognitive evidences’, *Applied Intelligence*, vol. 51, no. 7, pp. 4214–4235, 2021.
- [9] A. A. Balan, P. Anoop, and A. S. Mahesh, ‘Clickbait Detection Using Long short-term memory’, in *2022 Second International Conference on Interdisciplinary Cyber Physical Systems (ICPS)*, 2022, pp. 159–163.
- [10] S. Regina, K. Purwandari, F. I. Kurniadi, and Others, ‘Clickbait Headline Detection Using Supervised Learning Method’, in *2022 IEEE International Conference on Internet of Things and Intelligence Systems (IoT&IS)*, 2022, pp. 408–412.
- [11] Y.-W. Ma, J.-L. Chen, L.-D. Chen, and Y.-M. Huang, ‘Intelligent Clickbait News Detection System Based on Artificial Intelligence and Feature Engineering’, *IEEE Transactions on Engineering Management*, 2022.
- [12] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, ‘Natural language processing (almost) from scratch’, *Journal of machine learning research*, vol. 12, no. ARTICLE, pp. 2493–2537, 2011.
- [13] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, ‘Semi-supervised recursive autoencoders for predicting sentiment distributions’, in *Proceedings of the 2011 conference on empirical methods in natural language processing*, 2011, pp. 151–161.
- [14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, ‘Efficient estimation of word representations in vector space’, *arXiv preprint arXiv:1301.3781*, 2013.
- [15] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, ‘Natural language processing (almost) from scratch’, *Journal of machine learning research*, vol. 12, no. ARTICLE, pp. 2493–2537, 2011.

- [16] V. Piek et al., “Newsreader: How semantic web helps natural language processing helps semantic web,” Special Issue Knowledge Based Systems, Elsevier.
- [17] M. Potthast, S. Köpsel, B. Stein, and M. Hagen, ‘Clickbait detection’, in European conference on information retrieval, 2016, pp. 810–817.
- [18] R. Mihalcea and P. Tarau, ‘Textrank: Bringing order into text’, in Proceedings of the 2004 conference on empirical methods in natural language processing, 2004, pp. 404–411.
- [19] A. Anand, T. Chakraborty, and N. Park, ‘We used neural networks to detect clickbaits: You won’t believe what happened next!’, in European Conference on Information Retrieval, 2017, pp. 541–547.
- [20] S. R. Khater, O. H. Alsahlee, D. M. Daoud, and M. S. A. ElSeoud, ‘Clickbait detection’, in Proceedings of the 7th international conference on software and information engineering, 2018, pp. 111–115.
- [21] S. Zannettou, S. Chatzis, K. Papadamou, and M. Sirivianos, ‘The good, the bad and the bait: Detecting and characterizing clickbait on youtube’, in 2018 IEEE Security and Privacy Workshops (SPW), 2018, pp. 63–69.
- [22] Jonas Depoix, youtube-transcript-api : YouTube Transcript/Subtitle API (including automatically generated subtitles and subtitle translations), 2020,
Available at : <https://pypi.org/project/youtube-transcript-api/>
- [23] P. Biyani, K. Tsioutsoulouklis, and J. Blackmer, ““ 8 amazing secrets for getting more clicks”: detecting clickbaits in news streams using article informality’, in Thirtieth AAAI conference on artificial intelligence, 2016.
- [24] K. El-Arini and J. Tang, ‘News feed fyi: Clickbaiting’, Facebook Newsroom, 2014.

- [25] J. Rieis, F. de Souza, P. V. de Melo, R. Prates, H. Kwak, and J. An, ‘Breaking the news: First impressions matter on online news’, in Proceedings of the international AAAI conference on web and social media, 2015, vol. 9, pp. 357–366.
- [26] G. J. Digirolamo and D. L. Hintzman, ‘First impressions are lasting impressions: A primacy effect in memory for repetitions’, *Psychonomic Bulletin & Review*, vol. 4, no. 1, pp. 121–124, 1997.
- [27] G. Loewenstein, ‘The psychology of curiosity: A review and reinterpretation’, *Psychological bulletin*, vol. 116, no. 1, p. 75, 1994.
- [28] K. Yoon, ‘Convolutional Neural Networks for Sentence Classification [OL]’, arXiv Preprint, 2014.
- [29] M. Potthast, S. Köpsel, B. Stein, and M. Hagen, ‘Clickbait detection’, in European conference on information retrieval, 2016, pp. 810–817.
- [30] A. Gianotto, ‘Downworthy: A browser plugin to turn hyperbolic viral headlines into what they really mean’, downworthy. snipe. net, 2014.
- [31] M. M. U. Rony, N. Hassan, and M. Yousuf, ‘Diving deep into clickbaits: Who use them to what extents in which topics with what effects?’, in Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017, 2017, pp. 232–239.
- [32] W. Li and H. Zhuge, ‘Summarising news with texts and pictures’, in 2014 10th International Conference on Semantics, Knowledge and Grids, 2014, pp. 100–107.

LIST OF PUBLICATIONS

[1] Nipun Bansal and Kapil Kumar Yadav, “**A Comparative Study on Clickbait Detection using Machine Learning Based Methods**”, communicated and accepted at International Conference on Disruptive Technologies (ICDT-2023), IEEE Conference Record No.57929 11th - 12th May 2023, Greater Noida, India

[2] Nipun Bansal and Kapil Kumar Yadav, “**Clickbait Detection Using Bi- LSTM Model**”, communicated and accepted at First International Conference On Computational Intelligence For Information, Security And Communication Applications (CIISCA-2023), 22nd - 23rd June 2023, Bengaluru, India



International Conference on Disruptive Technologies (ICDT-2023)

IEEE Conference Record No: #57929

11th to 12th May, 2023

Certificate of Presentation

This is to certify that Mr./Ms./Dr. KAPIL KUMAR YADAV
of DELHI TECHNOLOGICAL UNIVERSITY
has presented / contributed paper entitled A COMPARATIVE STUDY ON
CLICKBAIT DETECTION USING MACHINE LEARNING BASED METHODS
in International Conference on Disruptive Technologies (ICDT-2023) dated 11th - 12th May,
2023 organized by Department of Computer Science & Engineering, GL Bajaj Institute of
Technology & Management, Greater Noida, (U.P.), India.


Dr. Sansar Singh Chauhan
Conference Chair
ICDT-2023


Dr. Shashank Awasthi
Conference Secretary
ICDT-2023


Prof. Manas Kumar Mishra
General Chair
ICDT-2023



Greater Noida

International Conference on Disruptive Technologies (ICDT-2023)

IEEE Conference Record No: #57929

11th to 12th May, 2023

Certificate of Best Paper

In recognition of the research paper quality, originality, and significance the paper titled *"A Comparative Study on Clickbait Detection using Machine Learning Based Methods"* Presented by Mr./Ms./Dr. Kapil Kumar Yadav of DTU, New Delhi is selected as **Best Paper** in International Conference on Disruptive Technologies (ICDT-2023) dated 11th - 12th May, 2023 organized by Department of Computer Science & Engineering, GL Bajaj Institute of Technology & Management, Greater Noida, (U.P.), India.

Dr. Sansar Singh Chauhan
Conference Chair
ICDT-2023

Dr. Shashank Awasthi
Conference Secretary
ICDT-2023

Prof. Manas Kumar Mishra
General Chair
ICDT-2023

CIISCA 2023 notification for paper 8519

1 message

CIISCA 2023 <ciisca2023@easychair.org>
To: Kapil Kumar Yadav <singhkapil347@gmail.com>

Thu, May 25, 2023 at 10:53 AM

Dear Kapil Kumar Yadav,

Congratulations!

On behalf of the CIISCA-2023 Technical Program Committee, we are pleased to inform you that your submitted paper ID: 8519, entitled "Clickbait Detection Using Bi-LSTM Model" has been accepted for presentation in CIISCA-2023 to be held in Global Academy of Technology, Bangalore, Karnataka, India, during June 22– 23, 2023 online and in-person mode.

Paper ID: PID-8519

You are requested to use Paper ID for further communication regarding this paper.

Thanks for your support to CIISCA-2023.

Please note that the accepted papers will be indexed in IEEE Xplore database if

- The final/Camera Ready paper uses the proper format according to the template provided by IEEE.
 - The final/Camera Ready paper passes through IEEE compliance in terms of PDF Express.
 - The final/Camera Ready paper passes IEEE Plagiarism test (<20%).
 - The final/Camera Ready paper (both word and pdf format) must be mailed on or before 27th May 2023 to ciisca@gat.ac.in.
- Please write the Title of the mail as per the following format: PID-8519_Camera ready paper
- The IEEE electronic Copyright Form is filled and ECF link will be shared with separate mail after completing registration and camera-ready paper is received.
 - One of the Author must register and present the paper in the conference.
 - The reviewer's comments (if any, in the bottom of this email) must be addressed properly in the final/Camera Ready of the paper.

Note:

- The registration link: <https://forms.gle/GQASDNR3qwat2iiDA>
- Payment details:

Account Holder Name: Industry Institute Interaction Cell- GAT

Account Number: 143510100047637

Bank and Branch: Union Bank of India, RR Nagar Branch

IFSC Code: UBIN0814351

- Complete the registration by 27th May 2023.
- If you want plagiarism report then write to us to ciisca@gat.ac.in Please write the Title of the mail as per the following format: PID-8519_plagiarism report required

We also like to extend our sincere thanks to the reviewers for handling review of this paper. Thanks for submitting your paper to CIISCA-2023. We look forward to see you during the Conference.

Best regards,

Dr. Galiveeti Hemakumar Reddy

Technical Program Committee Chair(s), CIISCA-2023

SUBMISSION: 8519

TITLE: Clickbait Detection Using Bi-LSTM Model

----- REVIEW 1 -----

SUBMISSION: 8519

TITLE: Clickbait Detection Using Bi-LSTM Model

AUTHORS: Kapil Kumar Yadav and Nipun Bansal

----- Overall evaluation -----

SCORE: 2 (accept)

---- TEXT:

The paper is well drafted.

However the authors are strictly informed to follow the IEEE template and reduce the plagiarism.

The authors may add the recent literature in the survey section. The figures quality can be enhanced.

The result section is good.

----- Recommendation for best paper awards -----

SELECTION: no



GLOBAL ACADEMY OF TECHNOLOGY
GROWING AHEAD OF TIME....
Autonomous Institute, Affiliated To VTU

[Home](#) [Committee](#) [Keynote](#) [Call For Paper](#) [Registration](#) [Contact Us](#) [More](#)

FIRST INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCE FOR INFORMATIONS, SECURITY AND COMMUNICATION APPLICATIONS

CIISCA - 2023

 17 22nd & 23rd JUNE 2023-BENGALURU

**ORGANIZED BY
DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE
GLOBAL ACADEMY OF TECHNOLOGY**

[SUBMIT PAPER HERE](#)

“NOTE: All the CIISCA 2023 presented papers will be submitted to IEEE xplore and submitted for indexing to Scopus”