**Detection of clickbait and non-clickbait using handcrafted features and autoencoder features.**

*A Major* Project Report

**Submitted by:-**

Nitesh Anand

B.Tech Cse AIML 3rd Year

Quantum University, Roorkee.



**Under the Supervision of:-**

Prof. Jyoti Prakash Singh

CSE Department NIT Patna

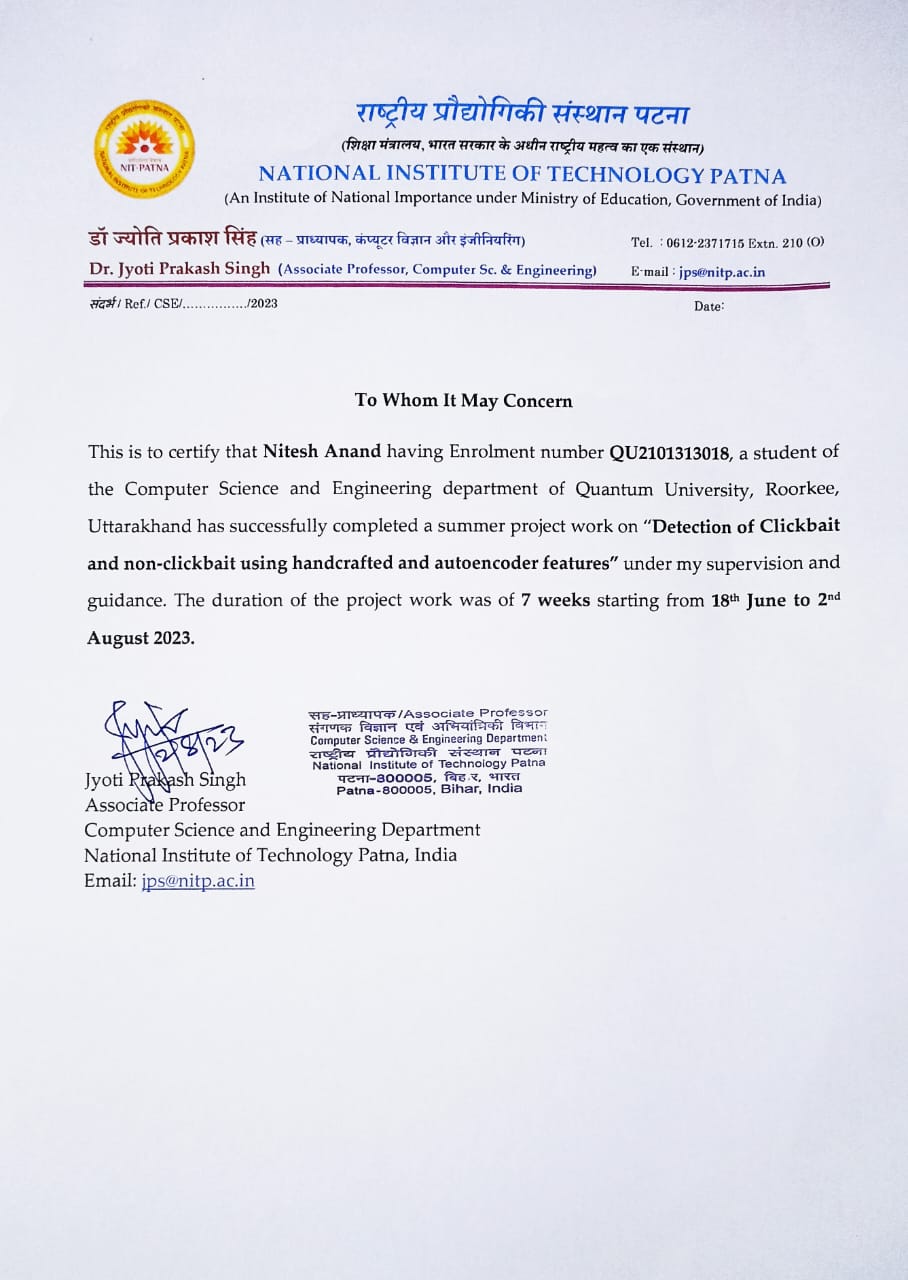
**ACKNOWLEDGEMENT**

First, I would like to thank **Dr. Jyoti Prakash Singh** (Associate Prof.) at **NIT Patna** for giving me the opportunity to do an internship within the organization. I also would like all the people that worked along with me NIT Patna with their patience and openness they created an enjoyable working environment. It is indeed with a great sense of pleasure and immense sense of gratitude that I acknowledge the help of these individuals.

Nitesh Anand

B.Tech Cse

Quantum University, Roorkee.



Introduction

In the virtual age, the internet has revolutionized the manner we get admission to and eat records. However, this full-size landscape of online content material isn't without its challenges. One conventional issue is the upward push of clickbait – a shape of misleading or sensationalized content material designed to attract customers' attention and inspire them to click on a hyperlink. Clickbait regularly employs catchy headlines and misleading records to drive site visitors and generate ad revenue. While it may appear innocent, clickbait can have sizeable implications for media intake, statistics integrity, and consumer experience.

Understanding Clickbait

Clickbait articles are characterised through exaggerated or deceptive headlines that pique readers' curiosity. They create a sense of urgency or exhilaration, enticing customers to click on the link to find out more. However, once customers get entry to the content material, they regularly find that it fails to deliver at the promises made in the headline, leaving them feeling misled and disappointed.

The Need for Detection

The prevalence of clickbait poses several challenges for users, content material creators, and digital structures alike. For users, encountering clickbait can cause wasted time, frustration, and a dwindled believe in on-line content material. Clickbait also can divert interest from actual and informative articles, hindering customers' capability to get entry to credible information.

Content creators and digital systems face the hazard of decreased user engagement and credibility if their content is related to clickbait. Additionally, the propagation of clickbait can make contributions to a tradition of sensationalism and incorrect information, undermining the integrity of digital media.

Project Objective

The number one objective of our assignment is to broaden a sturdy clickbait detection system using system gaining knowledge of techniques. By as it should be figuring out clickbait content material, we purpose to offer customers with the manner to make informed alternatives approximately the articles they choose to examine and have interaction with. Furthermore, this project seeks to help content material creators and virtual systems in selling accountable content material creation and fostering a more honest on line environment.

Approach

To gain our assignment's purpose, we are able to start by means of collecting and curating a classified dataset such as clickbait and non-clickbait articles. This dataset will serve as the foundation for training and evaluating our clickbait detection model.

Next, we will pre-process the text records and extract relevant capabilities that can help distinguish clickbait from actual content material. These functions may additionally encompass linguistic styles, phrase frequency, sentiment analysis, and other textual attributes.

We will then discover diverse machine gaining knowledge of algorithms, inclusive of Support Vector Machines, Random Forest, Logistic Regression, and others, to construct and teach our clickbait detection model. The version will be best-tuned and evaluated the use of suitable performance metrics to make certain its accuracy and reliability.

Expected Impact

The successful implementation of a clickbait detection device could have a superb impact on multiple fronts. For users, it will empower them to identify and avoid clickbait, main to a extra worthwhile and significant on-line revel in. Content creators and virtual structures can gain from improved user engagement and popularity, as they promote real and precious content material.

Additionally, our project's findings can contribute to the broader efforts in media literacy and combatting incorrect information. By creating an powerful device for detecting clickbait, we hope to foster a virtual environment that values content material integrity, person pleasure, and accountable content material advent.

Data Description

I have two dataset large-mergedDataset.csv and normalized\_dataset.csv .The first data has 15 columns which shows it includes a more basic set of functions or has not undergone large feature engineering, whereas the second dataset has undergone great feature engineering and pre-processing, resulting in an extra huge set of features.

Data Pre-processing

Text pre-processing on the 'targetTitle' column of the Data Frame. It converts the text to lowercase, removes URLs and punctuation, tokenizes the text, removes stop words, lemmatizes words, and joins the tokens back together. The resulting cleaned text is saved to a new CSV file named 'preprocess\_merged.csv'.

Data Glove Embedding

Data Frame with the pre-processed 'targetTitle' column and then loads Glove word vectors from the 'glove.6B.100d.txt' file. It defines a function to get Glove word embedding for each text in the 'targetTitle' column by averaging the embedding of individual words. The resulting embedding are added as a new column 'targetTitle\_glove' in the Data Frame.

The code then saves the Data Frame with embedding to a new CSV file named 'preprocess\_merged\_with\_glove.csv',this step is important because it converts the pre-processed text into numerical representations (GloVe embedding’s) that can be used as features for machine learning models. The GloVe embedding’s capture semantic information of words and help in improving the performance of text-based models.

Data Splitting

We performs a train-test split on the input data (X), target labels (Y), and the corresponding ids. It splits the data into three sets: training set (X\_train, y\_train), development set (X\_dev, y\_dev), and test set (X\_test, y\_test) using the train\_test\_split function. The split is done in such a way that 80% of the data is used for training and 20% for testing, with a random seed of 42 to ensure reproducibility.

After the split, the code creates Data Frames for each set (X\_train\_df, X\_dev\_df, X\_test\_df, y\_train\_df, y\_dev\_df, y\_test\_df) by filtering the original DataFrame (df5) using the corresponding 'id' values for each set. This ensures that the DataFrames for each set contain only the relevant rows corresponding to the split data.

Finally, the code saves the DataFrames for each set to separate CSV files ('X\_train.csv', 'X\_dev.csv', 'X\_test.csv', 'y\_train.csv', 'y\_dev.csv', 'y\_test.csv') while excluding the index column (index=False). This step is crucial for saving the data in a structured format and enables easy retrieval for future use, such as model training and evaluation

Training an Autoencoder

An autoencoder model is being built and trained to perform dimensionality reduction on the pre-processed GloVe embedding’s of the 'targetTitle' column. The goal of the autoencoder is to compress the high-dimensional embedding’s into a lower-dimensional space while preserving as much information as possible.

The autoencoder architecture consists of an encoder and a decoder. The encoder takes the original embedding’s as input and applies several dense layers with ReLU activation functions to gradually reduce the dimensionality of the data. The bottleneck layer represents the compressed representation of the input embedding’s.

The decoder then takes the bottleneck layer as input and applies dense layers to reconstruct the original embedding’s with the same dimensionality. The autoencoder is trained to minimize the mean squared error loss between the original embedding’s and the reconstructed embedding’s.

After building the autoencoder, it is trained using the pre-processed 'X\_dev\_glove' data. The autoencoder is trained for 50 epochs with a batch size of 32 and a validation split of 0.1.

Finally, the summary of the autoencoder model is displayed, showing the layers, output shapes, and the number of trainable parameters.

Overall, this autoencoder is designed to learn a lower-dimensional representation of the input embedding’s, which can be used as features for subsequent classification tasks, such as clickbait detection. The lower-dimensional embedding’s obtained from the autoencoder may capture important patterns and relationships in the data, enabling more efficient and effective classification.

Features Extraction

The pre-processed DataFrame with GloVe embedding’s and class labels is loaded for both the training and testing datasets. The 'id' column is extracted from these DataFrames, and the 'targetTitle\_glove' column is converted into numpy arrays for X\_train and X\_test.

The trained autoencoder model is loaded to obtain the encoder part of the model. The encoder is then used to extract encoded representations (50 features) from the bottleneck layer for both X\_train and X\_test.

To create the final feature set, the extracted features are concatenated with the 'class label' column, and DataFrames are created for the extracted features with the 'id' column included. These DataFrames are then saved to separate CSV files for X\_train\_features and X\_test\_features.

Overall, this process is designed to obtain a reduced feature set with 50 features, derived from the original GloVe embedding’s and the additional 'class label' column. These extracted features will be used as inputs for subsequent classification models, aiming to detect clickbait articles effectively.

Machine Classifier Used

* Support Vector Machine (SVM)
* Random Forest Classifier
* Logistic Regression
* k-Nearest Neighbours (KNN) Classifier
* Naive Bayes Classifier
* Gradient Boosting Classifier
* Decision Tree Classifier
* AdaBoost Classifier

Methodology

Data Collection and Prеprocеssing:-

In this research, two CSV files are used as the input data: 'normalizеd\_datasеt. csv' and 'largе-mеrgеdDatasеt. csv'. These files contain information related to clickbait detection, such as post titles, media, and other attributes.

The pandas library is еmployеd to read the CSV files and store them in DataFramеs: df1 and df2, rеspеctivеly. The number of columns in each DataFramе is also calculated and printed.

The two DataFramеs are merged based on the 'id' column to create a new DataFramе named 'mеrgеd\_df'. This merging stop combines the data from both sources into a single dataset.

To improve the quality of the text data in the 'targеtTitlе' column, a prеprocеssing function is applied. The function converts the text to lowercase, removes URLs and non-word characters, tokenizes the text, removes stop words, lemmatizes the words, and rеassеmblеs the cleaned text. The prеprocеssеd 'targеtTitlе' column is saved to a new CSV file, ‘prеprocеss\_mеrgеd. csv'.

Word Embedding’s with Glove:-

The genism library is utilized to load prе-trainеd word еmbеddings from the 'glove. 6B. 100d. txt' file. Glove (Global Vectors for Word Rеprеsеntation) word еmbеddings are a popular choice for natural language processing tasks.

The 'targеtTitlе' column in the 'prеprocеss\_mеrgеd. csv' DataFramе is processed to obtain Glove еmbеddings for each word. This step converts the text into numerical vectors that rеprеsеnt the semantic meaning of words.

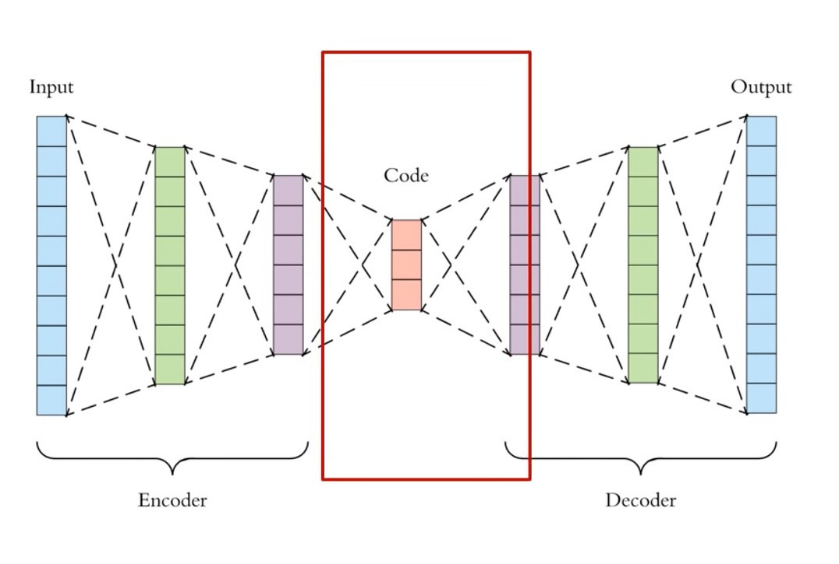
The DataFramе with Glove еmbеddings is saved to a new CSV file, ‘prеprocеss\_mеrgеd\_with\_glovе. csv'. This new dataset contains both the prеprocеssеd text and the corresponding Glove еmbеddings.

Autoеncodеr for Dimеnsionality Rеduction:-

An autoеncodеr model is built using the kеras library. The autoеncodеr is an unsupervised neural network architecture used for dimensionality reduction.

The autoеncodеr is trained on the 'targеtTitlе\_glovе' column of the 'prеprocеss\_mеrgеd\_with\_glovе. csv' DataFramе. The goal is to reduce the dimensionality of the Glove еmbеddings while retaining their еssеntial features.

The еncodеr part of the trained autoеncodеr is еxtractеd, resulting in compressed еmbеddings. These compressed еmbеddings rеprеsеnt a lowеr-dimеnsional rеprеsеntation of the input data.



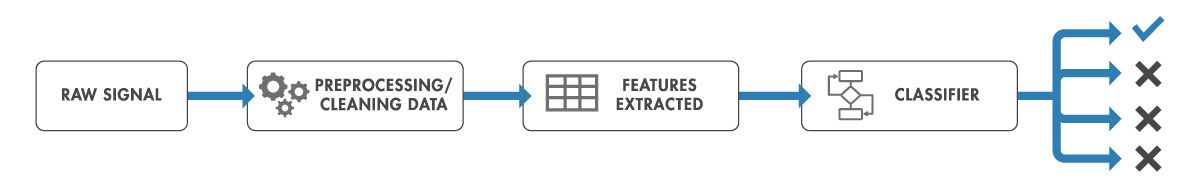
Feature Extraction:-

The compressed еmbеddings obtained from the autoеncodеr are concatenated with the 'class label' column to create 'X\_train\_combinеd' and 'X\_tеst\_combinеd'. Including the class label еnablеs the use of supervised learning algorithms.

The еncodеr part of the autoеncodеr is used to extract 50 features from the bottleneck layer for both the training and testing datasets. These features capture the most relevant information from the Glove еmbеddings while reducing the dimensionality.

DataFramеs, ‘X\_train\_fеaturеs\_df' and 'X\_tеst\_fеaturеs\_df', are crated to store the еxtractеd features along with the 'id' column. Each row in these DataFramеs rеprеsеnts a sample with its corresponding еxtractеd features and 'id'.

The еxtractеd features are saved to CSV files, ‘X\_train\_fеaturеs. csv' and 'X\_tеst\_fеaturеs. csv', for further use in the classification process.

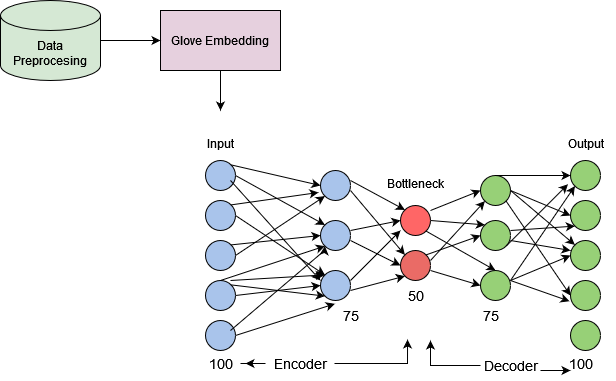


Data Merging and Final Prеprocеssing:-

The еxtractеd features are merged with the original training and testing DataFramеs using the 'id' column. This process combines the еxtractеd features with the rest of the information in the dataset.

Unnecessary columns, such as 'postTimеstamp', 'postTеxt', 'postMеdia', 'targеtTitlе', 'targеtDеscription', 'targеtKеywords', 'targеtParagraphs', 'targеtCaptions', 'truthJudgmеnts', 'truthModе', 'truth Class', 'targеtTitlе\_glovе', 'targеtClass Label', 'truthMеan', and 'truthMеdian', are dropped from the merged DataFramеs to obtain the final feature matrices, 'X\_train\_final' and 'X\_tеst\_final'.

The final feature matrices are saved to CSV files, ‘X\_train\_final. csv' and 'X\_tеst\_final. csv', for use in the classification process.



Classification:-

Different classification algorithms are sеlеctеd for the clickbait detection task. These include Support Vector Machin (SVM), Random Forest, Logistic Regression, K-Nearest Neighbours (KNN), Naive Bayes, Gradient Boosting, Decision Trее, and AdaBoost.

Each classifier is trained on the 'X\_train\_final' and 'X\_tеst\_final' datasets, which contain the еxtractеd features. The class labels for training and testing are obtained from the 'class label' column.

The accuracy, classification report (including precision, recall, and F1-scorе), and confusion matrix are computed for each classifier. These metrics provide an evaluation of the classification performance on the testing dataset.

Results and Discussion:-

The results of the classification models are analysed and discussed. This includes comparing the performance of different classifiers to identify which algorithm performs best for clickbait detection.

The quality of the feature extraction using the autoеncodеr is еvaluatеd, and its impact on the classification accuracy is discussed.

Insights gained from the classification results, such as the identification of important features or patterns in clickbait detection, are prеsеntеd and discussed.

So After applying different classifier like SVM, Random Forest, Logistic regression, KNN, Naive Bayes.Gredient boosting, Decision Tree and Adaboot Classifier, we have different cases of result we get.

As we know we first merged the normalized dataset and large dataset, then pre-processed the targettitle column and glove embedded the same column. Then split the data into 3 parts dev,train and test in the ratio of 40:40:20.After that we used the first dev data of 40% to train the autoencoder model of dim=100,2 dense layer at both encoder and decoder part and bottleneck=50.From there we extracted the encoder part. Then we removed all the unnecessary columns and in that encoder part we passed both the 2 parts i.e. train and test (40% & 20%) and from bottleneck we got 50 features.

In that 50 features we again merged the handcrafted features and applied different machine classifiers and noted the result which were as follows:-

Autoencoder features

Training SVM...

SVM Accuracy: 0.8472

SVM Classification Report:

Precision recall f1-score support

0 0.86 0.96 0.90 1801

1 0.79 0.51 0.62 587

Accuracy 0.85 2388

Macro avg 0.82 0.73 0.76 2388

Weighted avg 0.84 0.85 0.83 2388

SVM Confusion Matrix:

[[1722 79]

[ 286 301]]

==================================================

Training Random Forest...

Random Forest Accuracy: 0.8446

Random Forest Classification Report:

Precision recall f1-score support

0 0.84 0.97 0.90 1801

1 0.84 0.45 0.59 587

Accuracy 0.84 2388

Macro avg 0.84 0.71 0.75 2388

Weighted avg 0.84 0.84 0.83 2388

Random Forest Confusion Matrix:

[[1752 49]

[ 322 265]]

==================================================

Logistic Regression Accuracy: 0.8626

Logistic Regression Classification Report:

Precision recall f1-score support

0 0.88 0.95 0.91 1801

1 0.80 0.59 0.68 587

Accuracy 0.86 2388

Macro avg 0.84 0.77 0.80 2388

Weighted avg 0.86 0.86 0.86 2388

Logistic Regression Confusion Matrix:

[[1712 89]

[ 239 348]]

==================================================

KNN Accuracy: 0.7969

KNN Classification Report:

Precision recall f1-score support

0 0.83 0.91 0.87 1801

1 0.62 0.44 0.52 587

Accuracy 0.80 2388

Macro avg 0.73 0.68 0.69 2388

Weighted avg 0.78 0.80 0.78 2388

KNN Confusion Matrix:

[[1642 159]

[ 326 261]]

==================================================

Training Naive Bayes...

Naive Bayes Accuracy: 0.6537

Naive Bayes Classification Report:

Precision recall f1-score support

0 0.93 0.58 0.72 1801

1 0.40 0.87 0.55 587

Accuracy 0.65 2388

Macro avg 0.67 0.73 0.63 2388

Weighted avg 0.80 0.65 0.68 2388

Naive Bayes Confusion Matrix:

[[1051 750]

[ 77 510]]

==================================================

Training Gradient Boosting...

Gradient Boosting Accuracy: 0.8848

Gradient Boosting Classification Report:

Precision recall f1-score support

0 0.89 0.97 0.93 1801

1 0.87 0.63 0.73 587

Accuracy 0.88 2388

Macro avg 0.88 0.80 0.83 2388

Weighted avg 0.88 0.88 0.88 2388

Gradient Boosting Confusion Matrix:

[[1744 57]

[ 218 369]]

==================================================

Training Decision Tree...

Decision Tree Accuracy: 0.7697

Decision Tree Classification Report:

Precision recall f1-score support

0 0.85 0.85 0.85 1801

1 0.53 0.52 0.53 587

Accuracy 0.77 2388

Macro avg 0.69 0.69 0.69 2388

Weighted avg 0.77 0.77 0.77 2388

Decision Tree Confusion Matrix:

[[1531 270]

[ 280 307]]

==================================================

Training AdaBoost...

AdaBoost Accuracy: 0.8706

AdaBoost Classification Report:

Precision recall f1-score support

0 0.89 0.94 0.92 1801

1 0.78 0.65 0.71 587

Accuracy 0.87 2388

Macro avg 0.84 0.80 0.81 2388

Weighted avg 0.87 0.87 0.87 2388

AdaBoost Confusion Matrix:

[[1695 106]

[ 203 384]]

3 features

After then we did some experiment in which we selected only 3 prominent features to analyse the result which were as follows:-

SVM

Precision recall f1-score support

0 0.85 0.97 0.91 1801

1 0.82 0.48 0.61 587

Accuracy 0.85 2388

Macro avg 0.84 0.72 0.76 2388

Weighted avg 0.84 0.85 0.83 2388

[[1741 60]

[ 305 282]]

Random Forest

Precision recall f1-score support

0 0.84 0.96 0.89 1801

1 0.77 0.44 0.56 587

Accuracy 0.83 2388

Macro avg 0.80 0.70 0.73 2388

Weighted avg 0.82 0.83 0.81 2388

[[1723 78]

[ 330 257]]

Logistic Regression

Precision recall f1-score support

0 0.83 0.97 0.89 1801

1 0.79 0.39 0.52 587

Accuracy 0.82 2388

Macro avg 0.81 0.68 0.71 2388

Weighted avg 0.82 0.82 0.80 2388

[[1741 60]

[ 359 228]]

KNN

Precision recall f1-score support

0 0.82 0.93 0.87 1801

1 0.64 0.38 0.48 587

Accuracy 0.80 2388

Macro avg 0.73 0.66 0.68 2388

Weighted avg 0.78 0.80 0.78 2388

[[1678 123]

[ 364 223]]

Naive Bayes

Precision recall f1-score support

0 0.93 0.27 0.42 1801

1 0.30 0.94 0.45 587

Accuracy 0.44 2388

Macro avg 0.62 0.61 0.44 2388

Weighted avg 0.78 0.44 0.43 2388

[[ 488 1313]

[ 34 553]]

Gradient Boosting

Precision recall f1-score support

0 0.86 0.95 0.90 1801

1 0.78 0.53 0.63 587

Accuracy 0.85 2388

Macro avg 0.82 0.74 0.77 2388

Weighted avg 0.84 0.85 0.84 2388

[[1714 87]

[ 277 310]]

Decision Tree

Precision recall f1-score support

0 0.84 0.84 0.84 1801

1 0.51 0.52 0.51 587

Accuracy 0.76 2388

Macro avg 0.67 0.68 0.68 2388

Weighted avg 0.76 0.76 0.76 2388

[[1505 296]

[ 283 304]]

AdaBoost classifier

Precision recall f1-score support

0 0.86 0.92 0.89 1801

1 0.68 0.53 0.60 587

Accuracy 0.82 2388

Macro avg 0.77 0.73 0.74 2388

Weighted avg 0.82 0.82 0.82 2388

[[1657 144]

[ 275 312]]

4 Features

Then to further enhance our experiment we increased a feature, i.e. now we did it with 4 prominent features, and the results were:-

SVM

Precision recall f1-score support

0 0.86 0.95 0.90 1801

1 0.78 0.53 0.63 587

Accuracy 0.85 2388

Macro avg 0.82 0.74 0.77 2388

Weighted avg 0.84 0.85 0.84 2388

[[1714 87]

[ 277 310]]

Random Forest

Precision recall f1-score support

0 0.86 0.95 0.90 1801

1 0.79 0.52 0.63 587

Accuracy 0.85 2388

Macro avg 0.82 0.74 0.77 2388

Weighted avg 0.84 0.85 0.84 2388

[[1718 83]

[ 279 308]]

Logistic Regression

Precision recall f1-score support

0 0.85 0.95 0.89 1801

1 0.74 0.48 0.59 587

Accuracy 0.83 2388

Macro avg 0.80 0.71 0.74 2388

Weighted avg 0.82 0.83 0.82 2388

[[1704 97]

[ 304 283]]

Precision recall f1-score support

0 0.86 0.94 0.89 1801

1 0.72 0.52 0.60 587

Accuracy 0.83 2388

Macro avg 0.79 0.73 0.75 2388

Weighted avg 0.82 0.83 0.82 2388

[[1686 115]

[ 284 303]]

Naive Bayes

Precision recall f1-score support

0 0.93 0.12 0.22 1801

1 0.27 0.97 0.42 587

Accuracy 0.33 2388

Macro avg 0.60 0.55 0.32 2388

Weighted avg 0.77 0.33 0.27 2388

[[ 224 1577]

[ 17 570]]

Gradient Boosting

Precision recall f1-score support

0 0.87 0.95 0.91 1801

1 0.79 0.58 0.67 587

Accuracy 0.86 2388

Macro avg 0.83 0.76 0.79 2388

Weighted avg 0.85 0.86 0.85 2388

[[1711 90]

[ 248 339]]

Decision Tree

Precision recall f1-score support

0 0.86 0.85 0.85 1801

1 0.55 0.57 0.56 587

Accuracy 0.78 2388

Macro avg 0.71 0.71 0.71 2388

Weighted avg 0.78 0.78 0.78 2388

[[1533 268]

[ 253 334]]

AdaBoost classifier

Precision recall f1-score support

0 0.87 0.94 0.90 1801

1 0.75 0.59 0.66 587

Accuracy 0.85 2388

Macro avg 0.81 0.76 0.78 2388

Weighted avg 0.84 0.85 0.84 2388

[[1686 115]

[ 241 346]]

5 Features

Then we again increase one features and now done it by 5 features, and the result were:-

SVM

Precision recall f1-score support

0 0.84 0.96 0.90 1801

1 0.78 0.44 0.56 587

Accuracy 0.83 2388

Macro avg 0.81 0.70 0.73 2388

Weighted avg 0.82 0.83 0.81 2388

[[1727 74]

[ 331 256]]

Random Forest

Precision recall f1-score support

0 0.86 0.96 0.90 1801

1 0.79 0.51 0.62 587

Accuracy 0.85 2388

Macro avg 0.82 0.73 0.76 2388

Weighted avg 0.84 0.85 0.83 2388

[[1720 81]

[ 287 300]]

Logistic Regression

Precision recall f1-score support

0 0.84 0.95 0.89 1801

1 0.75 0.43 0.55 587

Accuracy 0.82 2388

Macro avg 0.79 0.69 0.72 2388

Weighted avg 0.82 0.82 0.81 2388

[[1716 85]

[ 333 254]]

KNN

Precision recall f1-score support

0 0.82 0.95 0.88 1801

1 0.70 0.38 0.49 587

Accuracy 0.81 2388

Macro avg 0.76 0.66 0.69 2388

Weighted avg 0.79 0.81 0.79 2388

[[1703 98]

[ 363 224]]

Naive Bayes

Precision recall f1-score support

0 0.85 0.90 0.87 1801

1 0.62 0.52 0.57 587

Accuracy 0.80 2388

Macro avg 0.74 0.71 0.72 2388

Weighted avg 0.79 0.80 0.80 2388

[[1612 189]

[ 280 307]]

Gradient Boosting

Precision recall f1-score support

0 0.87 0.95 0.90 1801

1 0.77 0.55 0.64 587

Accuracy 0.85 2388

Macro avg 0.82 0.75 0.77 2388

Weighted avg 0.84 0.85 0.84 2388

[[1702 99]

[ 264 323]]

Decision Tree

Precision recall f1-score support

0 0.85 0.86 0.85 1801

1 0.55 0.52 0.53 587

Accuracy 0.78 2388

Macro avg 0.70 0.69 0.69 2388

Weighted avg 0.77 0.78 0.77 2388

[[1549 252]

[ 283 304]]

AdaBoost classifier

Precision recall f1-score support

0 0.87 0.92 0.90 1801

1 0.71 0.58 0.64 587

Accuracy 0.84 2388

Macro avg 0.79 0.75 0.77 2388

Weighted avg 0.83 0.84 0.83 2388

[[1664 137]

[ 249 338]]

7 Features

Now we increased 2 features and done it with 7 features, and the results were:-

SVM

Precision recall f1-score support

0 0.89 0.96 0.92 1801

1 0.83 0.64 0.72 587

Accuracy 0.88 2388

Macro avg 0.86 0.80 0.82 2388

Weighted avg 0.88 0.88 0.87 2388

[[1726 75]

[ 214 373]]

Random Forest

Precision recall f1-score support

0 0.88 0.97 0.92 1801

1 0.87 0.58 0.69 587

Accuracy 0.87 2388

Macro avg 0.87 0.78 0.81 2388

Weighted avg 0.87 0.87 0.87 2388

[[1749 52]

[ 247 340]]

Logistic Regression

Precision recall f1-score support

0 0.87 0.96 0.92 1801

1 0.83 0.58 0.68 587

Accuracy 0.87 2388

Macro avg 0.85 0.77 0.80 2388

Weighted avg 0.86 0.87 0.86 2388

[[1730 71]

[ 248 339]]

KNN

Precision recall f1-score support

0 0.88 0.94 0.91 1801

1 0.75 0.61 0.67 587

Accuracy 0.85 2388

Macro avg 0.82 0.77 0.79 2388

Weighted avg 0.85 0.85 0.85 2388

[[1685 116]

[ 231 356]]

Naive Bayes

Precision recall f1-score support

0 0.96 0.31 0.46 1801

1 0.31 0.96 0.47 587

Accuracy 0.47 2388

Macro avg 0.63 0.63 0.47 2388

Weighted avg 0.80 0.47 0.46 2388

[[ 551 1250]

[ 25 562]]

Gradient Boosting

Precision recall f1-score support

0 0.90 0.95 0.92 1801

1 0.81 0.66 0.73 587

Accuracy 0.88 2388

Macro avg 0.85 0.81 0.83 2388

Weighted avg 0.88 0.88 0.88 2388

[[1710 91]

[197 390]]

Decision Tree

Precision recall f1-score support

0 0.87 0.86 0.86 1801

1 0.58 0.59 0.59 587

Accuracy 0.80 2388

Macro avg 0.72 0.73 0.73 2388

Weighted avg 0.80 0.80 0.80 2388

[[1553 248]

[ 240 347]]

AdaBoost classifier

Precision recall f1-score support

0 0.91 0.93 0.92 1801

1 0.76 0.70 0.73 587

Accuracy 0.87 2388

Macro avg 0.83 0.81 0.82 2388

Weighted avg 0.87 0.87 0.87 2388

[[1668 133]

[ 174 413]]

10 Features

And finally we increased 3 more features to get the total of 10 features and the results were like:-

SVM

Precision recall f1-score support

0 0.90 0.96 0.93 1801

1 0.85 0.66 0.74 587

Accuracy 0.89 2388

Macro avg 0.87 0.81 0.83 2388

Weighted avg 0.88 0.89 0.88 2388

[[1733 68]

[ 202 385]]

Random Forest

Precision recall f1-score support

0 0.87 0.97 0.91 1801

1 0.84 0.55 0.67 587

Accuracy 0.86 2388

Macro avg 0.85 0.76 0.79 2388

Weighted avg 0.86 0.86 0.85 2388

[[1738 63]

[ 263 324]]

Logistic Regression

Precision recall f1-score support

0 0.88 0.96 0.92 1801

1 0.84 0.58 0.69 587

Accuracy 0.87 2388

Macro avg 0.86 0.77 0.80 2388

Weighted avg 0.87 0.87 0.86 2388

[[1734 67]

[ 245 342]]

KNN

Precision recall f1-score support

0 0.86 0.94 0.89 1801

1 0.73 0.52 0.61 587

Accuracy 0.83 2388

Macro avg 0.79 0.73 0.75 2388

Weighted avg 0.83 0.83 0.82 2388

[[1686 115]

[ 281 306]]

Naive Bayes

Precision recall f1-score support

0 0.91 0.75 0.82 1801

1 0.50 0.78 0.61 587

Accuracy 0.76 2388

Macro avg 0.71 0.76 0.72 2388

Weighted avg 0.81 0.76 0.77 2388

[[1344 457]

[ 127 460]]

Gradient Boosting

Precision recall f1-score support

0 0.89 0.96 0.92 1801

1 0.83 0.64 0.72 587

Accuracy 0.88 2388

Macro avg 0.86 0.80 0.82 2388

Weighted avg 0.88 0.88 0.87 2388

[[1725 76]

[ 212 375]]

Decision Tree

Precision recall f1-score support

0 0.87 0.86 0.87 1801

1 0.59 0.60 0.59 587

Accuracy 0.80 2388

Macro avg 0.73 0.73 0.73 2388

Weighted avg 0.80 0.80 0.80 2388

[[1555 246]

[ 236 351]]

AdaBoost classifier

Precision recall f1-score support

0 0.90 0.93 0.92 1801

1 0.76 0.69 0.73 587

Accuracy 0.87 2388

Macro avg 0.83 0.81 0.82 2388

Weighted avg 0.87 0.87 0.87 2388

[[1674 127]

[ 180 407]]

As we were seeing that with increasing features the accuracy were also increasing, and there comes a point when it becomes stable. Thus this shows our model is well working on detection of clickbait and non-clickbait.

Conclusion

In summary, this project aimed to address the pressing issue of clickbait detection, which is crucial for promoting transparency and trustworthiness in online content. Clickbait refers to deceptive or misleading headlines designed to attract clicks, often leading to disappointing or irrelevant content for users.

To detect clickbait, we employed a combination of natural language processing and deep learning techniques. The data pre-processing phase involved text cleaning, tokenization, and removing unnecessary words to prepare the textual data for analysis. We utilized pre-trained GloVe word embedding’s to transform text into numerical representations, enabling the application of deep learning algorithms.

For feature extraction, we designed an autoencoder neural network that captured essential patterns in the textual data while reducing its dimensionality. The extracted features served as the foundation for training various classifiers, including Support Vector Machine (SVM), Random Forest, Logistic Regression, k-Nearest Neighbors (KNN), Naive Bayes, Gradient Boosting, Decision Tree, and AdaBoost classifiers.

Performance evaluation was conducted using standard metrics such as precision, recall, F1-score, and confusion matrices. Based on the results, we selected the model with the highest overall performance in clickbait detection.

In conclusion, our clickbait detection model demonstrates the effectiveness of combining advanced natural language processing techniques with deep learning to tackle the challenge of clickbait. By implementing this model, online platforms and social media can enhance content quality, foster user trust, and reduce exposure to misleading or sensationalized content. Continuous refinement and adaptation will be essential to keep pace with evolving clickbait strategies. Overall, this project contributes valuable insights to the field of clickbait detection and content moderation