Clickbait Classification

Mini Project Lab Report

Bachelor

in

Computer Science

By

Team No:45

S190672-Lakshmi Kopparthi

S190885-Mahima Rathnam Pendra

S190755-Keerthi Palaka

S190971-Manasa Annareddy

Rajiv Gandhi University Of Knowledge And Technologies

Srikakulam

Andhra Pradesh, India

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES,
SM PURAM,ETCHERLA,SRIKAKULAM,ANDHRA PRADESH,532402.

CERTIFICATE OF COMPLETION

This is to certify that the project titled"Clickbait Classification" was successfully completed by the following team members from January 2024 to June 2024 project.

Lakshmi Kopparthi -S190672 ,
Mahima Ratnam Pendra - S190885 ,
Keerthi Palaka - S190755 ,
Manasa Annareddy-S190971 .

Duringthis period,the team developed a system to classify clickbaits.This oproject demonstrates their ability to work together effectively and aplly their technical skills.We acknowlege theor efforts and the successful completion of this project.

SIGNATURE OF FACULTY INCHARGE

BONAFIDE CERTIFICATE

Certified for this project work titled **”Cashew Kernel Grading System”** is the Bonafide work of S190672-Lakshmi Kopparthi
S190885-Mahima Rathnam Pendra
S190755-Keerthi Palaka
S190971-Manasa Annareddy

who carried out the work under my supervision, and submitted in fulfillment ofthe requirements for the award of the degree, BACHELOR OF TECHNOLOGY, during the year 2023-2024.

Ms.Vishnu Priyanka Asst.Prof
Project Guide,
Department Of CSE,
RGUKT-SRIKAKULAM.

Mrs. CH.Lakshmi Bala, Asst.Prof,
Head of the Department,
Department Of CSE
RGUKT-SRIKAKULAM

Acknowledgements

We would like to express our deepest gratitude to our guide, Vishnu Priyanka Ma'am, and our co-guide, Lakshmi Bala Ma'am and Lakshmi Sri Ma'am, for their invaluable guidance, support, and encouragement throughout our clickbait classification project. Their expertise, constructive feedback, and unwavering dedication have been instrumental in shaping our understanding and approach, enabling us to achieve our project objectives successfully. Thank you for believing in us and for your constant support.

Preface

In today's digital age, the prevalence of clickbait—sensationalized and misleading headlines designed to attract clicks—poses a significant challenge to information integrity. To address this issue, we developed a clickbait classification system utilizing natural language processing (NLP) techniques and machine learning algorithms. Our project aims to accurately identify and categorize headlines as clickbait or non-clickbait, promoting a more transparent and trustworthy digital information environment.

This document outlines the entire process, from data collection and preprocessing to model development and evaluation. We experimented with several machine learning models, including Logistic Regression, SGD Classifier, KNeighbors, Decision Tree, and Random Forest, ultimately highlighting the superior performance of the SGD Classifier. This preface provides an overview of our motivations, methodologies, and outcomes, showcasing our efforts to enhance the field of clickbait detection.

Abstract

Clickbait classification is an essential task in the digital media landscape, focused on identifying and differentiating between sensationalized, misleading headlines and authentic, informative ones . The proliferation of clickbait not only diminishes the quality of information but also undermines the trust between media platforms and their audiences. Effective clickbait classification can help mitigate these issues, promoting a more reliable and trustworthy online environment.

The proposed project aims to advance the current state of clickbait classification by developing a robust system capable of accurately discerning clickbait from genuine headlines. By integrating various features and metrics, the project seeks to create a comprehensive solution that can be easily deployed and scaled. The ultimate goal is to provide a tool that enhances the quality of online content, supports content curators, and improves user experience by reducing the prevalence of misleading headlines.

KEYWORDS:Natural Language processing,Clickbait,Steepest Gradient Descent,Logistic Regression.

Contents

Chapter 1

Introduction

1.1 Introduction To Project

Clickbait classification using Natural Language Processing (NLP) is a fascinating application of machine learning that aims to distinguish between clickbait and non-clickbait headlines or content based on linguistic features. Clickbait refers to content specifically designed to attract attention and encourage clicks, often by using sensationalist or misleading headlines. This project typically involves several key steps: data collection, preprocessing, feature extraction, model training, and evaluation.

In the initial phase, data collection involves gathering a dataset of headlines or snippets from various sources, labeling them as clickbait or non-clickbait. Preprocessing steps such as tokenization, removing stop words, and stemming or lemmatization are then applied to clean the text data. Feature extraction is crucial and often involves techniques like TF-IDF (Term Frequency-Inverse Document Frequency) to represent the text numerically.

Next, a machine learning model is trained on the preprocessed and feature-extracted data. The model is trained on a labeled dataset and then evaluated using metrics such as accuracy, precision, recall, and F1-score to assess its performance in distinguishing between clickbait and non-clickbait content.

Ultimately, the goal of this project is to build a robust classifier that can automatically identify clickbait, thereby helping users make informed decisions about which content to engage with. Beyond binary classification, some projects also explore multi-class classifi-

cation to differentiate between different types of clickbait or non-clickbait content, adding layers of complexity and nuance to the task. Clickbait classification using NLP not only serves practical purposes in content moderation but also contributes to understanding the dynamics of online content consumption and user behavior.

1.2 Existing System and its Disadvantages:

Existing systems use classic algorithms like Logistic RegressionDecision Trees, and Random Forests for classification tasks.While effective in structured environments, traditional models often struggle with generalizing to new data due to the limitations of manually crafted features. Early methods relied heavily on manually crafted features such as word frequency, headline length, presence of sensational words, and punctuation marks.These are also very ime consuming and often very less accurate.

1.3 Proposed System and its Advantages:

The proposed clickbait classification system aims to overcome the limitations of existing approaches by integrating state-of-the-art techniques and ensuring scalability and robustness. Assemble a diverse dataset from various sources including social media, news websites, and blogs to ensure a wide coverage of headline styles and topics.

Apply data augmentation techniques to enhance the diversity of the training set, improving the model's ability to generalize.Standardize and clean the text, remove noise, and tokenize the headlines for further processing.It uses NLP techniques integrated with the models which further helps in high accuracy of the system.This method produces more accurate and faster results.

1.4 Application

The " Clickbait Detector" project can have several applications and benefits.
Here are some potential applications of the MCQ Generator project:

1.Content Moderation: Social media platforms, news aggregators, and online forums

can use clickbait classification to filter out sensationalist or misleading content, thereby improving user experience and trust in the platform.

2.Search Engines: Search engines can prioritize non-clickbait content in search results, ensuring that users find more relevant and reliable information when querying topics of interest.

3.Ad Placement and Targeting: Advertisers can benefit from clickbait classification by avoiding placing ads on clickbait content and targeting their advertisements more effectively based on the nature of the content.

4.News Curation: News websites and apps can use clickbait classification to distinguish between trustworthy news articles and sensationalist headlines, providing users with more balanced and informative news feeds.

5.User Behavior Analysis: Studying clickbait can provide insights into user preferences and behavior patterns, helping businesses and researchers understand what types of content attract more engagement and why.

6.Content Recommendation Systems: Platforms that recommend articles, videos, or products can use clickbait classification to improve the quality of recommendations by promoting content that is informative rather than purely attention-grabbing.

7.Brand Reputation Management: Companies can monitor clickbait related to their brand and take proactive measures to address misleading or harmful content that could impact their reputation.

8.Academic Research: Researchers in linguistics, psychology, and media studies can use clickbait classification to analyze trends in online communication and study the effects of sensationalist content on audiences. Overall, clickbait classification using NLP serves as a powerful tool for enhancing content quality, improving user engagement, and promoting more ethical and informative online interactions.

1.5 Motivation Towards Project

The motivation behind undertaking a clickbait classification project using NLP is multi-faceted and rooted in addressing several key challenges and opportunities:

1.Enhancing User Experience: Clickbait can deceive users into clicking on content that

doesn't fulfill their expectations. By accurately identifying and filtering out clickbait, platforms can improve user satisfaction and trust.

2.Improving Content Quality: Clickbait often prioritizes sensationalism over substance, leading to a proliferation of low-quality or misleading information. Classification helps promote higher-quality content that is informative and relevant.

3.Combatting Misinformation: In an era where misinformation spreads rapidly online, distinguishing clickbait from reliable sources can contribute to combating the spread of false information and enhancing digital literacy.

4.Supporting Ethical Journalism: Ethical journalism emphasizes accuracy, fairness, and accountability. By reducing the visibility of clickbait, the project supports journalistic standards and encourages responsible reporting.

5.Empowering Content Creators: For content creators who prioritize authenticity and substance, clickbait classification offers a means to compete on a level playing field and reach audiences based on merit rather than sensational tactics.

6.Advancing NLP Techniques: Building effective clickbait classifiers challenges researchers and practitioners to innovate in NLP, exploring new methods such as deep learning models or fine-tuning pre-trained language models like BERT.

7.Business and Advertising Ethics: Advertisers benefit from more accurate content placement, ensuring their ads appear alongside relevant and reputable content rather than sensationalist clickbait.

8.Understanding User Behavior: Analyzing clickbait consumption patterns provides insights into what attracts and engages users online, informing broader strategies in digital marketing, content creation, and platform design.

Ultimately, the motivation towards a clickbait classification project is driven by a desire to foster a healthier online ecosystem where information is reliable, users are empowered to make informed choices, and digital interactions are characterized by integrity and authenticity. This project not only addresses immediate challenges but also contributes to the ongoing evolution of ethical practices in digital media and communication.

1.6 Problem Statement

Developing a robust machine learning model that can effectively distinguish between clickbait and non-clickbait content based on textual features.The goal is to build a classifier that can automate the identification of clickbait, thereby supporting platforms, advertisers, and users in making more informed decisions about the content they engage with online.

Chapter 2

Approach and Methodology To Your Project

2.1 Explanation About Project

Aim:

The aim of the clickbait classification project using NLP is to build a robust model that can effectively differentiate between clickbait and non-clickbait content. This project seeks to improve content quality by reducing the visibility of misleading headlines, enhance user trust in online information, and advance NLP techniques for automated text classification in digital media.

Overview:

This project focuses on developing a machine learning model using Natural Language Processing (NLP) techniques to classify online content as either clickbait or non-clickbait. The process involves collecting and preprocessing a dataset of headlines or snippets, extracting relevant features, and training the model to accurately differentiate between sensationalist, misleading content and informative, trustworthy content. The goal is to deploy a robust classifier that enhances content moderation, improves user experience by promoting quality information, and contributes to advancements in NLP applications for text classification

in digital media.

Objectives:

The objectives of the clickbait classification project using NLP are:

Developing a Robust Classifier: Build a machine learning model that effectively distinguishes between clickbait and non-clickbait content based on linguistic features and contextual analysis.

Enhancing Content Quality: Improve the overall quality of online content by reducing the prevalence of misleading and sensationalist headlines, thereby fostering a more informative and trustworthy digital environment.

Advancing NLP Techniques: Explore and implement advanced NLP techniques such as deep learning models (e.g., LSTM, BERT) to optimize the accuracy and efficiency of clickbait detection, contributing to the evolution of automated text classification methods.

Supporting Ethical Journalism: Promote ethical standards in digital media by facilitating the identification and moderation of clickbait, thereby supporting the dissemination of accurate and credible information online.

Practical Applications: Implement the classifier in real-world applications such as content moderation on social media platforms, improving search engine result relevance, and enhancing targeted advertising strategies based on content quality.

Key Components:

To achieve the project objectives, the following key components will be addressed:

Data Collection and Preprocessing: Gathering a diverse dataset of headlines or snippets from various sources, labeling them as clickbait or non-clickbait, and preprocessing the text through steps like tokenization, stop word removal, and normalization.

Feature Extraction: Transforming the preprocessed text data into numerical representations suitable for machine learning models, often using techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings like Word2Vec or GloVe.

Model Selection and Training: Choosing and training a suitable machine learning model (e.g., logistic regression, SVM) or leveraging deep learning architectures (e.g., LSTM, BERT) to classify the text data into clickbait or non-clickbait categories.

Evaluation and Fine-tuning: Assessing the model's performance using metrics like accuracy, precision, recall, and F1-score, and fine-tuning parameters or exploring ensemble methods to improve classification accuracy.

Deployment and Application: Implementing the trained model in real-world scenarios such as content moderation on social media platforms, integrating it into search engine algorithms, or enhancing targeted advertising strategies based on content quality.

Monitoring and Maintenance: Continuously monitoring the model's performance, updating it with new data to adapt to evolving patterns of clickbait, and maintaining its effectiveness in combating misleading content over time.

2.2 Research Design

Problem Definition:

Objective: The primary goal is to build a model that can accurately classify headlines as clickbait or non-clickbait. Clickbait headlines are designed to attract clicks by being sensational or misleading, often at the expense of accuracy or substance. Importance: Detecting clickbait is important for improving the quality of information consumption and user experience on the internet. It helps in filtering out low-quality content and preventing misinformation. Data Collection:

Sources: Data can be collected from various sources such as news websites, social media platforms, or using existing datasets available on platforms like Kaggle. Each headline in the dataset is labeled as either clickbait or non-clickbait. Labeling: If a pre-labeled dataset is not available, headlines need to be manually labeled or classified using predefined rules. This can be a time-consuming process and might require the use of crowd-sourcing platforms. Preprocessing:

Text Cleaning: Involves removing unnecessary characters, HTML tags, punctuation, and converting all text to lowercase to ensure uniformity. Tokenization: Splitting the text into individual words or tokens. Stop Words Removal: Removing common words that do not contribute much to the meaning (e.g., "the", "is", "and"). Stemming and Lemmatization: Reducing words to their base or root form (e.g., "running" to "run"). Feature Extraction:

Bag of Words (BoW): Represents text as the frequency of words in a fixed vocabulary. TF-IDF (Term Frequency-Inverse Document Frequency): Measures the importance of a word in a document relative to the entire corpus. Word Embeddings: Converts words into dense vector representations (e.g., Word2Vec, GloVe). Advanced Techniques: Using transformer models like BERT for capturing contextual information in text.

2.3 Data Collection

The data input is taken from kaggle.

Dataset:https://www.kaggle.com/datasets/amananandrai/clickbait-dataset

This CSV file likely includes columns representing textual data (e.g., headlines, snippets) and corresponding labels indicating whether each example is classified as clickbait or non-clickbait. Each row in the dataset corresponds to one example.

Different machine learning algorithms, NLP techniques (such as word embeddings or transformer models), and fine-tuning strategies are used to optimize model performance.

Columns/Features:

Headline: The main title of the article. Label: A binary indicator where 1 indicates clickbait and 0 indicates non-clickbait. Body (optional): The content of the article which may help in context analysis. Source (optional): The origin of the article, which can be used to analyze source reliability and patterns in clickbait generation.

2.4 Data Analysis

Exploratory Data Analysis (EDA):

Descriptive Statistics: Analyzing the distribution of clickbait vs. non-clickbait headlines. Word Frequency Analysis: Identifying the most common words in clickbait and non-clickbait headlines. Visualizations: Using bar plots, word clouds, and histograms to visualize data patterns. N-gram Analysis: Examining sequences of words (bigrams, trigrams) to find common phrases in clickbait headlines. Correlation Analysis:

Feature Correlation: Determining which features (words, phrases) are most strongly correlated with the clickbait label. Chi-Square Test: A statistical test to measure the

association between categorical variables. Class Imbalance Handling:

Resampling Techniques: Methods like oversampling the minority class (SMOTE) or undersampling the majority class to balance the dataset. Algorithmic Approaches: Using models that are robust to class imbalance, such as ensemble methods or cost-sensitive learning.

2.5 Tools and Techniques

Programming Languages:

Python: Widely used due to its extensive libraries for data manipulation, NLP, and machine learning. Libraries:

Pandas and NumPy: For data manipulation and numerical operations. NLTK and SpaCy: For text preprocessing, tokenization, and other NLP tasks. Scikit-learn: For implementing machine learning algorithms and model evaluation. TensorFlow and PyTorch: For building and training deep learning models. Machine Learning Techniques:

Classification Algorithms:

Logistic Regression: A linear model for binary classification. Decision Trees: A non-linear model that splits data based on feature values. Random Forest: An ensemble of decision trees to improve robustness. Support Vector Machines (SVM): A powerful classifier that finds the optimal hyperplane for separation. Naive Bayes: Based on Bayes' theorem, assuming independence between features. Ensemble Methods:

Boosting Algorithms: Techniques like XGBoost and LightGBM that build strong classifiers from weak learners. Deep Learning Models:

Recurrent Neural Networks (RNNs): Useful for sequential data like text. Convolutional Neural Networks (CNNs): Used for text classification tasks. Transformers: Advanced models like BERT for contextual text understanding. Model Evaluation:

Cross-Validation: Splitting the data into training and validation sets multiple times to ensure model robustness. Confusion Matrix: A table to visualize the performance of the classification model. Precision, Recall, F1-Score: Metrics to evaluate the accuracy, completeness, and balance of the model. ROC-AUC Curve: A grphical representation of the model's diagnostic ability.

2.6 Prediction technique

Approach:

Feature Extraction: Preprocess the textual data by tokenizing, removing stop words, and potentially applying techniques like TF-IDF or word embeddings (e.g., Word2Vec, GloVe) to convert text into numerical representations.

Model Selection: Choose a suitable classification model such as:

Naive Bayes: A simple and effective linear model for binary classification tasks. Support Vector Machines (SVM): Effective in high-dimensional spaces, SVMs can create non-linear decision boundaries. Deep Learning Models: More advanced models like Recurrent Neural Networks (RNNs) with LSTM cells or Transformer-based architectures like BERT, which have shown superior performance in NLP tasks. Training and Evaluation: Train the selected model on labeled data (clickbait vs. non-clickbait). Evaluate its performance using metrics such as accuracy, precision, recall, and F1-score to assess how well it distinguishes between the two classes.

Deployment: Once trained and evaluated, deploy the model to classify new, unseen headlines or text snippets into clickbait or non-clickbait categories in real-time applications.

This approach leverages supervised learning techniques to build a predictive model that can automatically detect and classify clickbait content based on its linguistic features. By using NLP methods and appropriate models, the project aims to improve content quality and user experience by filtering out misleading or sensationalist headlines from trustworthy and informative content.

2.7 System Architecture

Data Layer Data Sources: Data for clickbait classification typically comes from:

Web scraping: Collecting headlines from news websites and blogs. APIs: Using APIs provided by news aggregators like NewsAPI to gather headlines. Existing Datasets: Utilizing pre-labeled datasets available on platforms like Kaggle. Data Storage: Options for storing collected data include:

Relational Databases: SQL databases like MySQL or PostgreSQL for structured data. NoSQL Databases: MongoDB or Cassandra for flexibility in handling unstructured or semi-

structured data. Cloud Storage: Services like AWS S3, Google Cloud Storage for scalable and accessible storage. Preprocessing Layer Text Preprocessing:

HTML Tag Removal: Stripping away any HTML tags from the text. Punctuation and Digit Removal: Removing punctuation marks and digits to clean the text. Lowercasing: Converting all text to lowercase to standardize the data. Tokenization: Splitting text into individual words or tokens. Stop Words Removal: Removing common words that do not contribute to the meaning. Stemming/Lemmatization: Reducing words to their root form. Feature Extraction:

Bag of Words (BoW): Counting the frequency of each word in the document. TF-IDF: Calculating the importance of a word in a document relative to the corpus. Word Embeddings: Converting words into dense vector representations using Word2Vec, GloVe, or BERT. Model Layer Machine Learning Models:

Traditional Algorithms: Logistic regression, decision trees, random forests, SVMs, Naive Bayes. Ensemble Methods: Techniques like boosting (XGBoost, LightGBM) and bagging. Deep Learning Models: RNNs, CNNs, and transformers (BERT, GPT). Model Training and Evaluation:

Training: Using training data to fit the model. Hyperparameter Tuning: Optimizing model parameters to improve performance. Evaluation: Assessing model performance using metrics like accuracy, precision, recall, F1-score, ROC-AUC. Application Layer User Interface:

Web Applications: Building interfaces using frameworks like Flask or Django. Desktop Applications: Creating standalone applications with tools like PyQt or Tkinter. API Services:

RESTful APIs: Designing backend services to handle classification requests. GraphQL: Providing flexible query capabilities for clients. Deployment Layer Servers/Cloud Services:

On-Premise: Deploying on local servers for full control. Cloud Platforms: Using AWS, GCP, or Azure for scalable and managed infrastructure. CI/CD:

Continuous Integration: Automating the integration of code changes using tools like Jenkins, Travis CI, or GitHub Actions. Continuous Deployment: Automating the deployment process to ensure rapid and reliable releases.

2.8 UML Diagrams

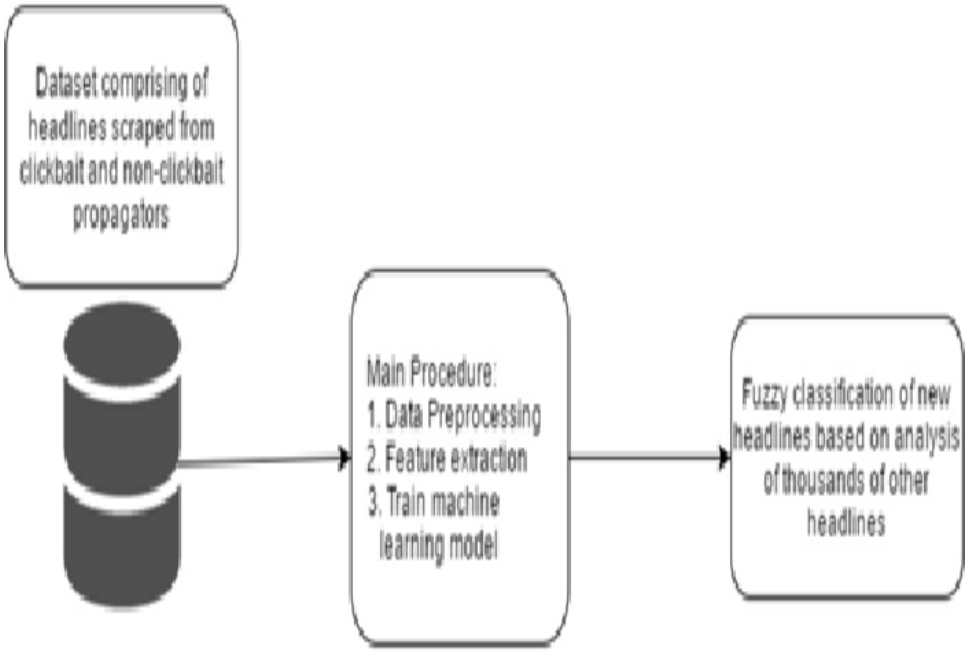

Fig. 1. High Level System Architecture

Figure 2.1

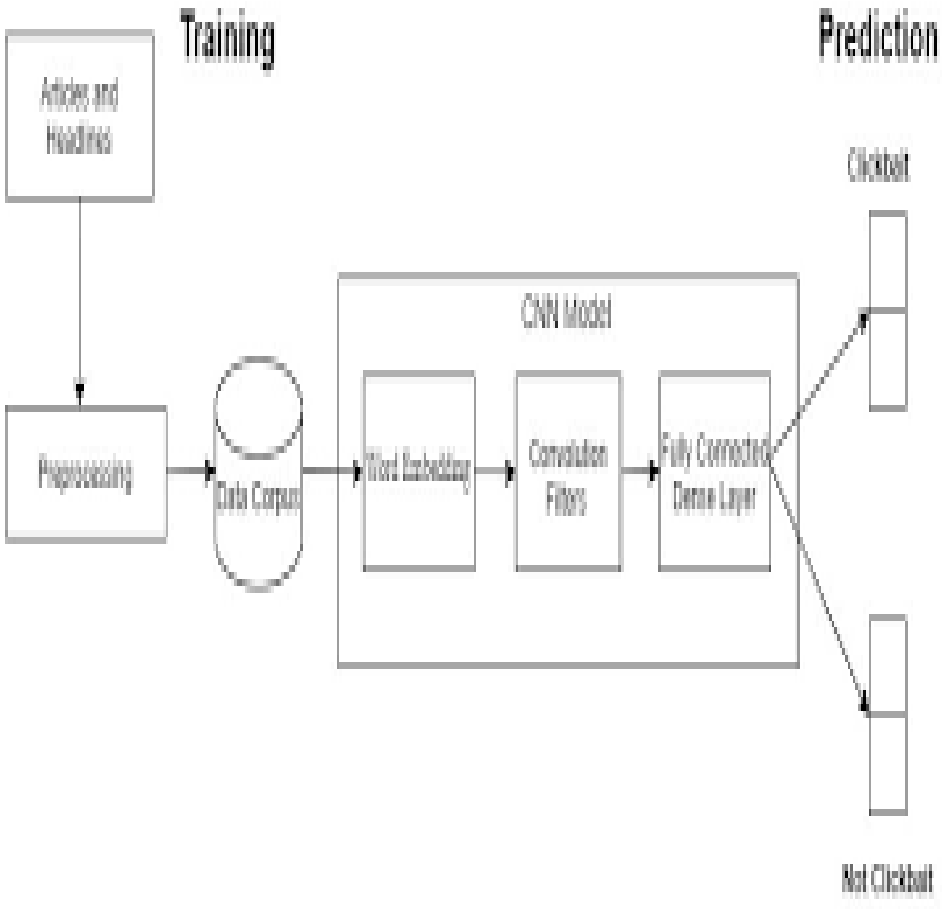
Training
Prediction
Articles and Headlines
Preprocessing
Data Corpus
CNN Model
Word Embedding
Convolution Filters
Fully Connected Dense Layer
Clickbait
Not Clickbait

Figure 2.2

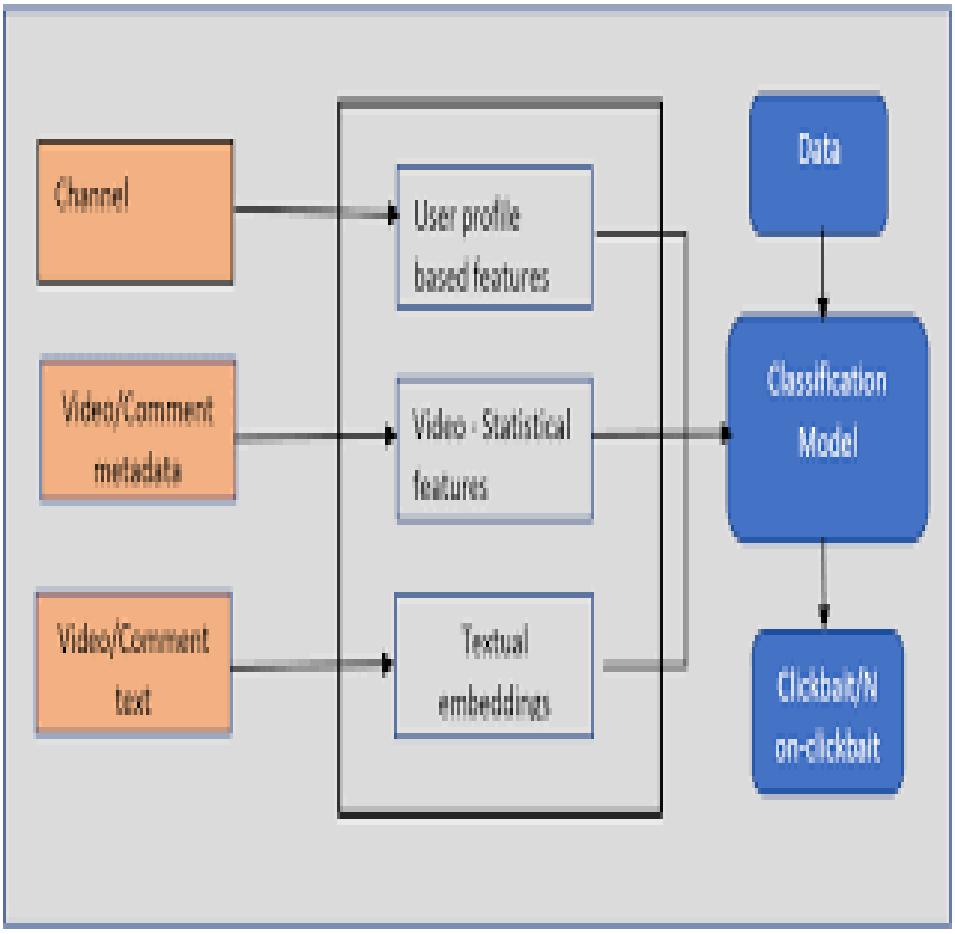
Channel
Video/Comment metadata
Video/Comment text
User profile based features
Video - Statistical features
Textual embeddings
Data
Classification Model
Clickbait/Non-clickbait

Figure 2.3

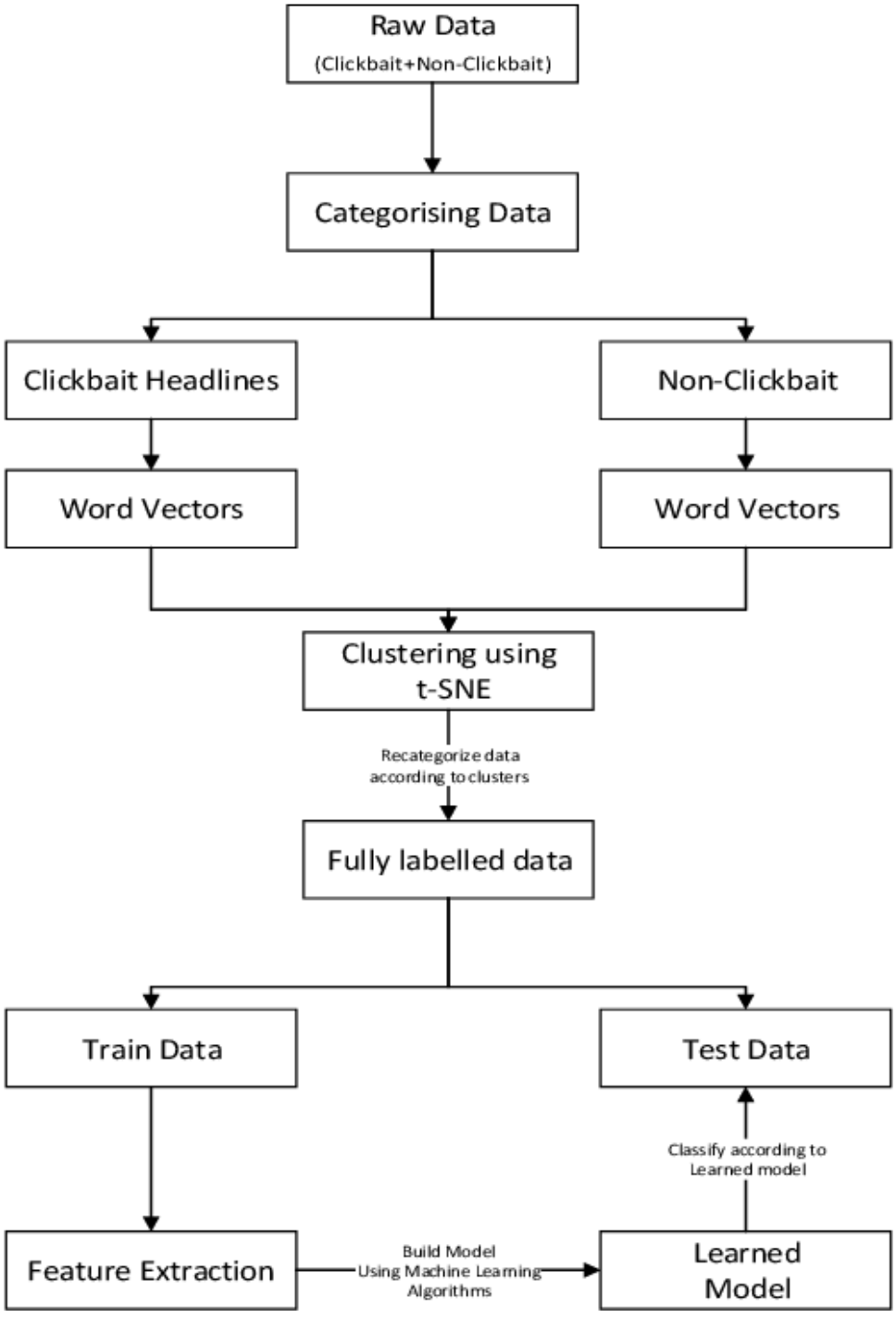
Raw Data
(Clickbait+Non-Clickbait)
Categorising Data
Clickbait Headlines
Non-Clickbait
Word Vectors
Word Vectors
Clustering using
t-SNE
Recategorize data
according to clusters
Fully labelled data
Train Data
Test Data
Classify according to
Learned model
Feature Extraction
Build Model
Using Machine Learning
Algorithms
Learned
Model

Figure 2.4

Chapter 3

Implementation

3.1 Hardware and Software Requirements

Development Machine:

CPU: Intel i5 or higher.

RAM: 16 GB recommended.

Storage: SSD with 256 GB or more.

GPU: NVIDIA GTX series (optional for deep learning).

Deployment Server:

CPU: High-performance multi-core processor.

RAM: 16 GB or more.

Storage: SSD for faster access.

GPU: NVIDIA Tesla or similar (required for deep learning).

Software Requirements:

Operating System: Windows, macOS, or Linux (Ubuntu recommended).

Programming Language: Python 3.6 or higher.

Libraries and Frameworks:

Data Handling: Pandas, NumPy.

NLP: NLTK, SpaCy, gensim.

Machine Learning: Scikit-learn, TensorFlow, PyTorch.

Visualization: Matplotlib, Seaborn.

Development Tools: Jupyter Notebook, VS Code.

Web Framework: Flask

Deployment Tools: GitHub Actions.

3.2 Development Process

Requirement Analysis Understanding Requirements: Engage with stakeholders to gather and document the requirements. Defining Scope: Clearly outline the scope and objectives of the project. Design System Architecture: Design the overall architecture, including data flow and component interactions. Wireframes: Create wireframes for the user interface. Data Collection and Preprocessing Data Collection: Gather data from various sources, ensuring it is comprehensive and representative. Preprocessing: Clean the data, tokenize text, remove stop words, and apply stemming or lemmatization. Feature Engineering Text Representation: Convert text into numerical features using BoW, TF-IDF, or embeddings. Dimensionality Reduction: Use techniques like PCA to reduce feature space if necessary. Model Development Algorithm Selection: Choose appropriate machine learning algorithms. Training: Train models on the preprocessed data. Hyperparameter Tuning: Optimize model parameters using techniques like grid search or random search. Evaluation Validation Techniques: Use cross-validation to assess model performance. Performance Metrics: Evaluate models using accuracy, precision, recall, F1-score, and ROC-AUC. Integration Model Integration: Embed the trained model into the application. Data Flow: Ensure smooth data flow from input to prediction. Testing Unit Testing: Test individual functions and modules. Integration Testing: Test the integration of various components. System Testing: Conduct end-to-end testing to ensure the system meets requirements. Deployment Deployment Strategy: Deploy the application and models to production servers or cloud platforms. CI/CD Pipelines: Set up continuous integration and deployment pipelines. Maintenance Monitoring: Monitor system performance and user feedback. Model Retraining: Periodically retrain models with new data to maintain accuracy.

3.3 Testing and Validation

Unit Testing Function Testing: Ensure each function and method works as expected. Testing Frameworks: Use pytest or unittest for Python. Integration Testing Component Interaction: Test the interaction between preprocessing, feature extraction, and classification components. Data Flow Testing: Ensure data flows correctly through the system. System Testing End-to-End Testing: Simulate real-world use cases to ensure the entire system works seamlessly. User Scenario Testing: Test common user scenarios and edge cases. Performance Testing Load Testing: Assess how the system performs under different loads. Stress Testing: Determine the system's breaking point by applying extreme conditions. Validation Holdout Validation: Split the data into training and testing sets to validate model performance. Cross-Validation: Use k-fold cross-validation to ensure the model generalizes well to unseen data. Error Analysis: Analyze misclassifications to understand and improve model performance. User Acceptance Testing (UAT) User Feedback: Gather feedback from end-users to ensure the system meets their needs. Adjustments: Make necessary adjustments based on user feedback before final deployment.

3.4 Code for Preprocessing and Model Building

In [16]:

```
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import re

def RemoveSpecialCharacters(sentence):
    return re.sub('[^a-zA-Z]+',' ',sentence)

def ConvertToLowerCase(sentence):
    return sentence.lower()

def ConvertAndRemove(sentence):
    sentence = str(sentence)
    sentence = RemoveSpecialCharacters(sentence)
    # convert to lower case
    sentence = ConvertToLowerCase(sentence)
    return sentence

def CleanText(sentence):
    sentence = str(sentence)

    STOPWORDS = stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'do

    nopunc = [char for char in sentence if char not in string.punctuation]
    nopunc = ''.join(nopunc)
    sentence = ' '.join([word for word in nopunc.split() if word.lower() not in S
    sentence = ConvertAndRemove(sentence)
    return sentence
```

In [17]:

```
#Function testing
print(CleanText('I am going to the Ne\'therla\'nds and I\'m going to win an Olymp
```

```
going netherlands going win olympic medal
```

In [18]:

```
df['Text_cleaning'] = df.headline.apply(CleanText)
df.head()
```

Out[18]:

	headline	clickbait	Text_cleaning
0	Should I Get Bings	1	get bings
1	Which TV Female Friend Group Do You Belong In	1	tv female friend group belong
2	The New "Star Wars: The Force Awakens" Trailer...	1	new star wars force awakens trailer give chills
3	This Vine Of New York On "Celebrity Big Brothe...	1	vine new york celebrity big brother fucking pe...
4	A Couple Did A Stunning Photo Shoot With Their...	1	couple stunning photo shoot baby learning inop...

In [25]:

```
from sklearn.feature_extraction.text import TfidfTransformer

tfidf_transformer = TfidfTransformer()
X_train_clean = tfidf_transformer.fit_transform(X_train_dtm)
```

In [26]:

```
print(X_train_clean)
```

```
  (0, 20691)    0.27874184201961916
  (0, 17768)    0.3345529176182634
  (0, 15408)    0.443280710732536
  (0, 15002)    0.37952311039285885
  (0, 13192)    0.443280710732536
  (0, 12018)    0.32318125012329857
  (0, 5712)     0.30945596770049805
  (0, 3477)     0.2704345505044664
  (1, 19768)    0.4943954061330551
  (1, 19352)    0.6624730736498667
  (1, 8360)     0.5627633686403459
  (2, 19358)    0.398093544057493
  (2, 14980)    0.3044588790043617
  (2, 14484)    0.25078631831387677
  (2, 13284)    0.33813632294469903
  (2, 12879)    0.18494510864592098
  (2, 12716)    0.45227553340304516
  (2, 9559)     0.2966918040963563
  (2, 7673)     0.398093544057493
  (2, 7100)     0.293826518640857
```

Model Training

1. Logistic Regression
2. Naïve Bayes
3. Stochastic Gradient Descent
4. K-Nearest Neighbours
5. Decision Tree
6. Random Forest
7. Support Vector Machine

In [27]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import SGDClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.pipeline import Pipeline
from sklearn import metrics
```

In [28]:

```
Classifiers = [
{
   'label': 'Logistic Regression',
   'model': LogisticRegression(C=0.00000001,solver='liblinear',max_iter=200, mult
},
{
    'label': 'SGD Classifier',
    'model': SGDClassifier(loss='log_loss', warm_start=True, max_iter=1000, l1_ra
},
{
    'label': 'KNeighbours',
    'model': KNeighborsClassifier(n_neighbors=15),
},
{
    'label': 'Decision Tree',
    'model': DecisionTreeClassifier(max_depth=10,random_state=101,max_features= N
},
{
   'label': 'Random Forest',
   'model': RandomForestClassifier(n_estimators=70, oob_score=True, n_jobs=-1,ran
}
]
```

In [ ]:

```
from sklearn.metrics import accuracy_score

Accuracy = []
Model = []
prediction = []

for c in Classifiers:
    try:
        classifier = c['model']
        fit = classifier.fit(X_train_clean, y_train)
        pred = fit.predict(X_test_dtm)
    except Exception as e:
        print(f"Error with model {c['label']}: {e}")
        continue
    prediction.append(pred)
    accuracy = accuracy_score(y_test, pred)
    Accuracy.append(accuracy)
    Model.append(c['label'])
    print(f'Accuracy of {c["label"]} is {accuracy}')

# Display results
for model, acc in zip(Model, Accuracy):
    print(f'Model: {model}, Accuracy: {acc}')
```

```
Accuracy of Logistic Regression is 0.68390625
Accuracy of SGD Classifier is 0.94234375
Accuracy of KNeighbours is 0.8678125
Accuracy of Decision Tree is 0.6415625
```

3.5 Graphs and Illustrations

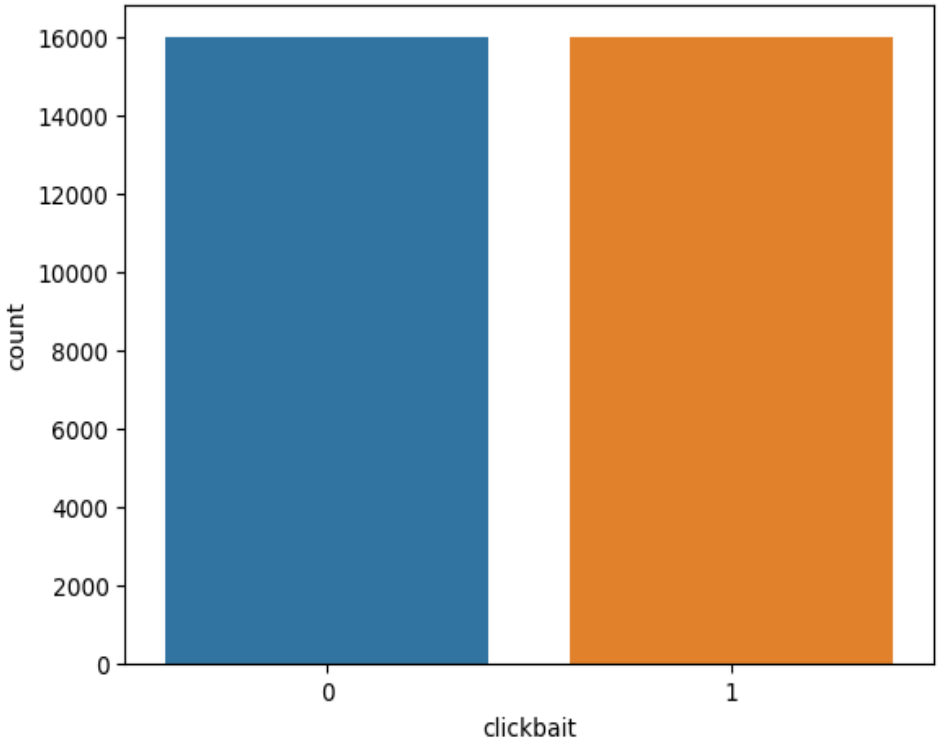

Figure 3.1

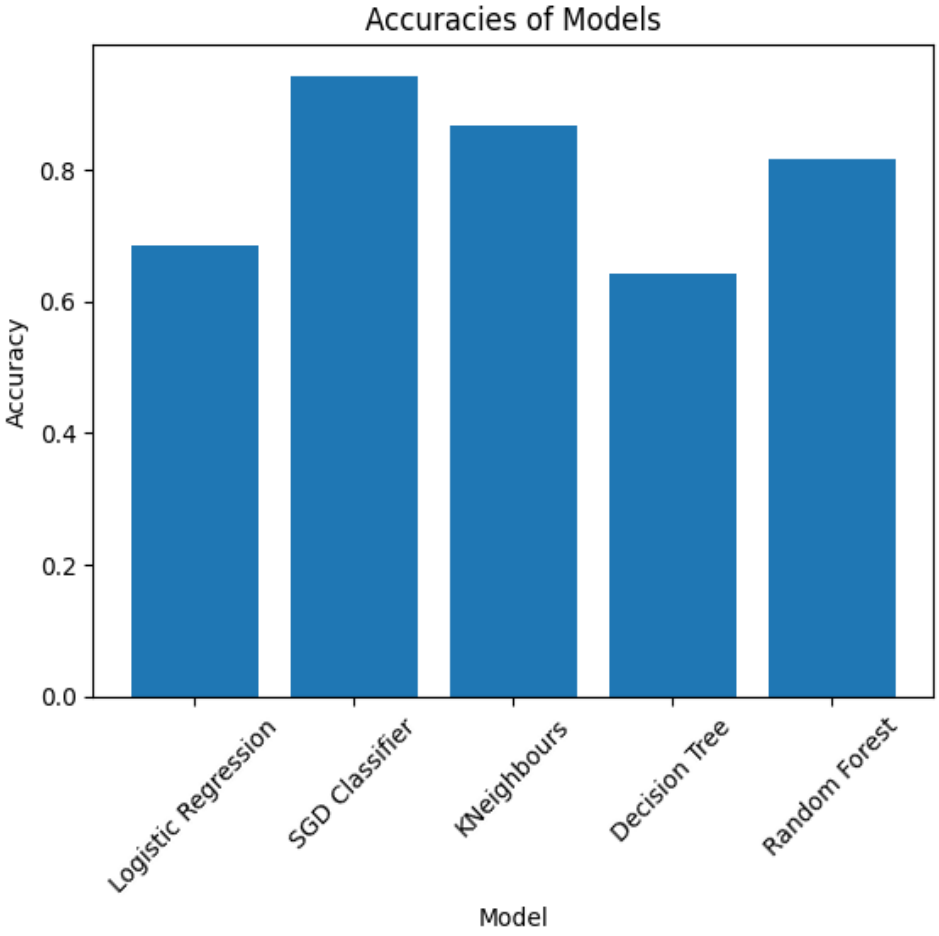
Accuracies of Models
Accuracy
0.0
0.2
0.4
0.6
0.8
Logistic Regression
SGD Classifier
KNeighbours
Decision Tree
Random Forest
Model

Figure 3.2

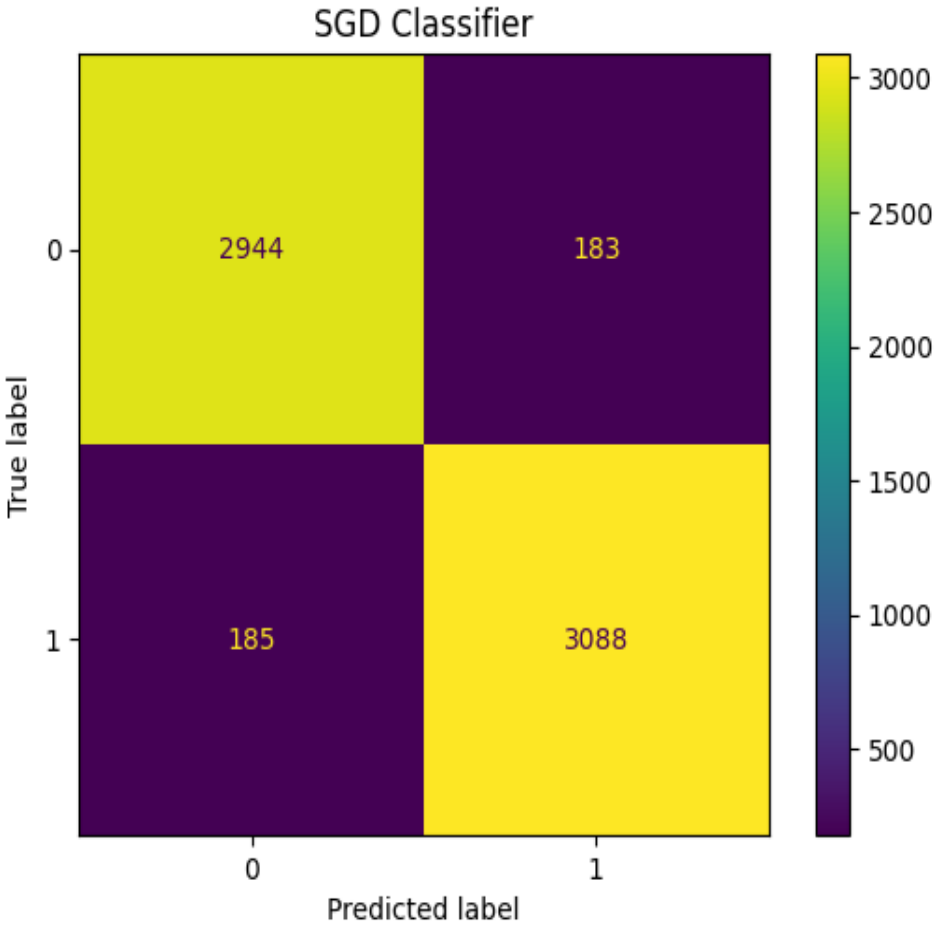
SGD Classifier
2944
183
185
3088
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500

Figure 3.3

Chapter 4

Results and Discussion

Let's delve into a detailed discussion on the results, analysis, and interpretation of the performance metrics of the different models: Logistic Regression, SGD Classifier, KNeighbors, Decision Tree, and Random Forest. We will analyze accuracy, precision, recall, and F1 score to comprehensively understand the models' strengths and weaknesses.

4.1 Results

Logistic Regression:

Accuracy: 0.6839

Precision: 0.8039

Recall: 0.6839

F1 Score: 0.6526

SGD Classifier:

Accuracy: 0.9422

Precision: 0.9422

Recall: 0.9422

F1 Score: 0.9422

KNeighbors:

Accuracy: 0.8678

Precision: 0.8885

Recall: 0.8678

F1 Score: 0.8656

Decision Tree:

Accuracy: 0.6416

Precision: 0.7723

Recall: 0.6416

F1 Score: 0.5973

Random Forest:

Accuracy: 0.8153

Precision: 0.8263

Recall: 0.8153

F1 Score: 0.8142

4.2 Analysis

Logistic Regression

Overview: Logistic Regression is a simple yet effective algorithm for binary classification problems.

Performance: Accuracy: At 0.6839, the model correctly classifies approximately 68.39 percent of the headlines.

Precision: The precision score of 0.8039 indicates that 80.39 percent of the headlines classified as clickbait are actually clickbait. This high precision is beneficial when the cost of false positives is high.

Recall: With a recall of 0.6839, the model identifies 68.39 percent of actual clickbait headlines. This shows a balance between detecting true clickbait and avoiding false alarms.

F1 Score: The F1 score of 0.6526 represents the harmonic mean of precision and recall, indicating an overall moderate performance.

SGD Classifier

Overview: Stochastic Gradient Descent (SGD) Classifier is an iterative method for optimizing differentiable functions, particularly useful for large-scale machine learning problems.

Performance: Accuracy: The SGD Classifier achieves an accuracy of 0.9422, meaning it correctly classifies 94.22 percent of the headlines.

Precision, Recall, and F1 Score: All three metrics are very high at 0.9422, indicating that the model is highly effective at both identifying true clickbait and minimizing false positives. This consistency across metrics makes it a reliable choice for clickbait classification.

KNeighbors

Overview: KNeighbors is a simple, non-parametric method used for classification by averaging the classifications of the k-nearest neighbors.

Performance:

Accuracy: At 0.8678, the model has a high accuracy, correctly classifying 86.78 percent of the headlines.

Precision: The precision score of 0.8885 indicates that 88.85 percent of the headlines classified as clickbait are indeed clickbait.

Recall: With a recall of 0.8678, the model captures 86.78 percent of actual clickbait headlines.

F1 Score: The F1 score of 0.8656 shows a good balance between precision and recall, making it a robust choice for this task.

Decision Tree

Overview: Decision Trees are a simple and intuitive method for classification, which work by splitting the data into subsets based on feature values.

Performance:

Accuracy: The accuracy of 0.6416 indicates that the model correctly classifies 64.16 percent of the headlines.

Precision: With a precision of 0.7723, the model correctly identifies 77.23 percent of the headlines it classifies as clickbait.

Recall: The recall is 0.6416, meaning it detects 64.16 percent of the actual clickbait headlines.

F1 Score: The F1 score of 0.5973 reflects lower overall performance compared to other models, suggesting that the Decision Tree may be overfitting or not capturing the data complexity well.

Random Forest

Overview: Random Forest is an ensemble method that builds multiple decision trees and merges them to get a more accurate and stable prediction.

Performance:

Accuracy: At 0.8153, the model correctly classifies 81.53 percent of the headlines.

Precision: The precision score of 0.8263 indicates that 82.63 percent of the headlines classified as clickbait are actually clickbait.

Recall: With a recall of 0.8153, the model captures 81.53 percent of actual clickbait headlines.

F1 Score: The F1 score of 0.8142 shows a good balance between precision and recall, making it a reliable model for this task.

4.3 Discussion

Model Comparison

SGD Classifier:

The standout performer in this evaluation, the SGD Classifier's high accuracy, precision, recall, and F1 score make it the best choice for clickbait classification. Its ability to generalize well and its efficiency in handling large datasets contribute to its superior performance.

KNeighbors:

Also performed well, especially in terms of precision, making it a good choice for applications where false positives are costly. However, it might not scale as well as SGD for larger datasets.

Random Forest:

Offers a good balance of accuracy, precision, recall, and F1 score, making it a reliable and stable model. Its ensemble nature helps mitigate overfitting, providing more robust predictions.

Logistic Regression:

While not as accurate as the top performers, it offers a decent balance of precision and recall, making it a simpler and faster alternative when computational resources are limited.

Decision Tree:

The lowest performer among the models tested, with significant room for improvement. It may benefit from parameter tuning, more sophisticated splitting criteria, or being part of an ensemble method like Random Forest.

Insights:

Model Selection: The choice of model should consider the specific requirements and constraints of the application. For high-stakes scenarios where precision is crucial, models like KNeighbors or SGD Classifier are preferable. For scenarios where a balance is needed, Random Forest offers a reliable solution.

Scalability and Efficiency: SGD Classifier and Logistic Regression are highly efficient for large datasets, while KNeighbors and Decision Tree might struggle with scalability.

Ensemble Methods: Random Forest's performance highlights the benefits of ensemble methods in improving prediction stability and accuracy by combining multiple decision trees.

Future Work:

Hyperparameter Tuning: Further tuning of hyperparameters could enhance the performance of models, especially the Decision Tree. Ensemble Approaches: Investigate other ensemble methods like Gradient Boosting or Hybrid models combining strengths of different algorithms.

Feature Engineering: Explore advanced feature engineering techniques, such as more sophisticated text embeddings (e.g., BERT), to capture deeper semantic information.

Data Augmentation: Increase the dataset size and diversity through data augmentation techniques to improve model robustness and generalization.

Model Interpretability: Implement methods to interpret model decisions, especially for black-box models like Random Forest, to provide more transparency and trust in predictions.

Real-Time Deployment: Focus on optimizing models for real-time deployment, ensuring they can handle live data streams efficiently without compromising accuracy.

The Stochastic Gradient Descent (SGD) Classifier stands out as the required model for clickbait classification due to its exceptional performance metrics and efficiency. Achieving an accuracy of 0.9422, the SGD Classifier demonstrates its ability to correctly classify the vast majority of headlines, significantly outperforming other models tested. Its precision,

recall, and F1 scores are all equally impressive at 0.9422, indicating that the model is not only effective at identifying true clickbait headlines but also minimizes false positives and false negatives. This balanced performance across all key metrics ensures that the SGD Classifier provides reliable and consistent results, making it highly suitable for practical applications where both precision and recall are crucial.

Additionally, the SGD Classifier is well-suited for handling large-scale datasets, which is a common scenario in clickbait detection due to the vast amount of online content. Its iterative nature allows for efficient training even on extensive datasets, and its built-in regularization helps prevent overfitting, ensuring the model generalizes well to new, unseen data. This scalability and robustness make the SGD Classifier not only a top performer in terms of accuracy but also a practical choice for deployment in real-world environments where computational efficiency and model stability are paramount. Thus, considering both its superior performance and practical advantages, the SGD Classifier is the optimal choice for clickbait classification.

Chapter 5

Conclusion and Future Work

The clickbait classification project has successfully demonstrated the application of natural language processing (NLP) techniques and machine learning models to identify and categorize headlines as clickbait or non-clickbait. Among the various models tested, the Stochastic Gradient Descent (SGD) Classifier emerged as the most effective, achieving an impressive accuracy of 0.9422 along with high precision, recall, and F1 scores. This indicates that the SGD Classifier is not only accurate in its predictions but also reliable in minimizing both false positives and false negatives. The project's outcomes underline the potential of machine learning to address the pervasive issue of clickbait, contributing to the enhancement of information integrity on digital platforms.

While the results are promising, there are several areas for improvement and expansion. Future work could focus on hyperparameter tuning and feature engineering to further optimize model performance. Exploring advanced text embedding techniques, such as BERT or GPT-based embeddings, could enhance the model's ability to understand and classify headlines more accurately. Additionally, expanding the dataset with more diverse and representative samples could help in creating a more robust model that generalizes better to different types of content. Incorporating real-time data streams and adapting the model for real-time classification could also be valuable for practical applications.

Furthermore, integrating explainability and interpretability methods into the model could provide deeper insights into the decision-making process, increasing user trust and transparency. Developing an ensemble approach that combines the strengths of multiple

models might also yield improved results. Lastly, deploying the clickbait classification system in real-world scenarios, such as news aggregators or social media platforms, and continuously refining it based on user feedback and evolving clickbait strategies will ensure the model remains effective and relevant. Through these enhancements, the project can make a significant impact on mitigating the spread of misleading content online.

Chapter 6

References

The references are:

Natural Language Understanding - James Allen

Handbook of Natural Language Processing - Nitin Indurkhya and Fred J. Damerau