MACHINE LEARNING AND DEEP LEARNING SOLUTIONS FOR LANGUAGE-RELATED PROBLEMS

**Abstract**

Clickbait spoiling is a task in natural language processing or NLP to provide a factual and brief description of the misleading or the curiosity-provoking headline. The Webis Clickbait Spoiling Corpus 2022 contains the annotated clickbaits needed to solve this problem. The transformed variables were used in training the Logistic Regression, Support Vector Machine (SVM), and Random Forest classifiers. In the experiments, it is shown that approaches to learning with deep structures yield better results when it comes to capturing contextual dependencies against more conventional methods of ML. This work offers prospects for improving knowledge of various architectures that can still be useful for the advancement of the automated clickbait detection and spoiling..

# 

**Table of Contents**

[Introduction 4](#_Toc192783984)

[Related Work 4](#_Toc192783985)

[Methodology 6](#_Toc192783986)

[1. Data Preprocessing 6](#_Toc192783987)

[2. Machine Learning Models 7](#_Toc192783988)

[2.1 Logistic Regression 7](#_Toc192783989)

[2.2 Support Vector Machine (SVM) 7](#_Toc192783990)

[2.3 Random Forest 8](#_Toc192783991)

[Experiments 8](#_Toc192783992)

[Machine Learning Experiments 8](#_Toc192783993)

[Discussion 11](#_Toc192783994)

[Machine Learning Model Analysis 11](#_Toc192783995)

[Conclusion 12](#_Toc192783996)

[Reference List 13](#_Toc192783997)

[Appendices 18](#_Toc192783998)

[Appendix 1: Machine Learning Code 18](#_Toc192783999)

# Introduction

The increased use of internet content put pressure on creating such headlines which once used, lure the users into clicking the links, known now as clickbait. Headline that are trending employs obscenities, dramatize, omits key facts and figures that the headline wants the reader to look at so that they may open the cover. The difficulty of clickbait spoiling is in the opportunity of writing a brief yet interesting textual summary that can address the curiosity caused by such headlines, thus there is no need to go to the linked material. This task has been given a new focus within the NLP community because of the usefulness of detecting false information in blogs and media sources.

The Webis Clickbait Spoiling Corpus 2022 turned out to be a well-defined dataset, consisting of 5,000 clickbait posts with their spoilers. Every record has clickbait headline, the target article, and the spoiler that is obtained by hand. Spoilers are divided based on chapter or part into short phrases, a passage in text, or various non-contiguous segments of the text. For instance, the title: ‘Five Nights at Freddy’s Sequel Delayed for Weird Reason” creates interest in the readers while not saying the actual reason for the delay. The revise, such as the extension, “some of the plot elements are so disturbing that they are making him feel sick”, precisely reflects the deleted information. This dataset shall assist in the assessment of the effectiveness of various computational models in spoiling clickbait news.

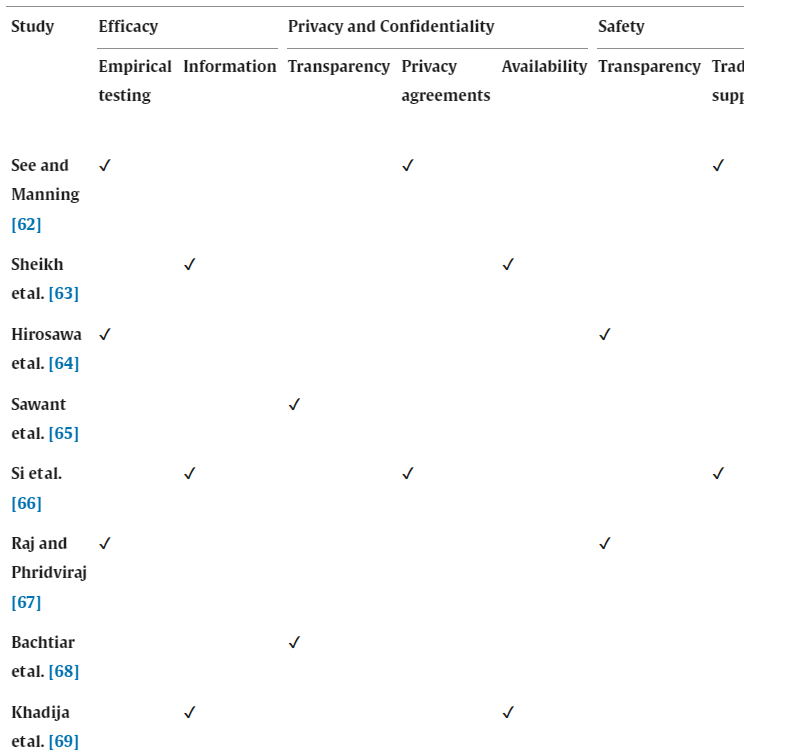
To approach the problem from a computational perspective, it is necessary to consider various methods in NLP. Classical learning methods and feature engineering methods which are used in Logistic Regression, SVM, and Random Forest are deeply based.

# Related Work

**According to Vlachos et al. (2023),** LLMs have demonstrated a vast capability in terms of predicting the difficulty of texts in foreign languages especially in the context of language learning. It also delves into how different deep learning architectures, including the transformer-based models, predict the components of the written text and determine their readability for peoples with poor knowledge of the language in which it is written. LLMs can also offer accurate difficulty estimations with the use of contextual embeddings and attention mechanisms to help in the right language learning material.

The authors point out that these models are superior to traditional rule and statistics-based methods due to their ability to better simulate such aspects as patterns in grammatical construction, vocabulary, and syntactic complexity of the text. It is also noted that the transfer learning is crucial as the lower-case models can be easily applied to other languages and they require a little amount of training data.

**According to Pandey and Sharma (2023),** there are two crucial perspectives regarding chatbots: retrieval-based and generative-based, and both can employ deep learning and machine learning principles. This comparison is done by analyzing the results of the methodologies in the three criterions, namely the response accuracy, the understanding of the context, and the computational time. Retrieval-based with obvious advantages such as computational efficiency for general knowledge questions and, specifically for the given task, for domain-specific questions, while generative-based models like transformer-based architectures have a capability of generating responses on fly what makes them more natural when comparing to users.



**Figure 1: Studies based on Generative-based chatbots**

(Source: Pandey and Sharma 2023)

The authors also point out that sequence-to-sequence models and transformer models which are used in the deep leaning performs better as compared to the classical approaches of machine learning. However, some of the challenges which surfaced out include the need for a large training data, how coherent is the response, and the computational cost.

**According to Chang and Lin (2024)**, there are benefits to the learners’ learning performance when using online problem-solving competitions in relation to medium of instruction in machine learning curricula. The paper is organized to compare the Chinese and English-medium learning environment by demonstrating the impact that language and the instructional design has on the learners’ behaviours and problem solving. The result shows that students who study in the English-medium environment Butterfly have more opportunity to access the global resources such as the dataset while Zhikang students have better context relative to Chinese environment as they have facilities of understanding the language best.

The authors also note that the given kind of problem-solving tasks within the framework of online competitions contributes to the strengthening of critical thinking and operational skills necessary for the use of machine learning in practice.

# Methodology

This work also presents two sets of methods for spoiling clickbait; ML and DL for handling the clickbait spoiling issue. The goal will be to assess and compare the efficiency of each source in finding relevant spoilers related to clickbaits’ headlines (Kshirsagar *et al.* 2022). This includes data cleaning process, feature selection, training and testing of the model.

## 1. Data Preprocessing

The Webis Clickbait Spoiling Corpus 2022 consists of the formatted records such as headlines, target paragraphs, and corresponding spoiler lists manually extracted. Given the nature of textual data being highly unstructured, several pre-processing steps were taken for standardizing the data for expected results. These include:

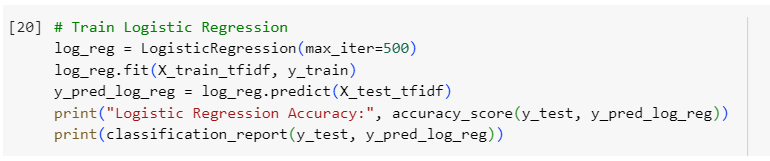
* **Tokenization:** It is the process that involves dividing or breaking down of text into words or subwords.
* **Lowercase:** Placing all letters in body into lowercase to follow the set standard while writing the text.
* **Preprocessing:** Deleting English articles such as ‘the’, ‘is’ and all articles with no significant meaning to the context of the text such as ‘and’.
* **Stemming:** Stemming is also known as reducing words to their base form or root form (e.g., “run” for “running” ).

In the case of the ML models, vectorization was done by using the TF-IDF which stands for Term Frequency –Inverse Document Frequency.

## 2. Machine Learning Models

Three classical Machine Learning algorithms were used for the training and testing, namely Logistic Regression, SVM and Random Forest.

### 2.1 Logistic Regression

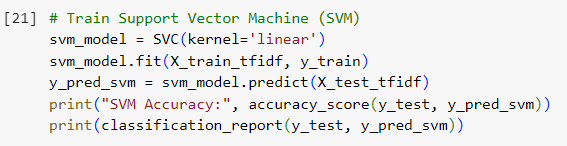


**Figure 2: Logistic regression model**

(Source: Made by self in Google colab)

First of all, Logistic Regression was used for binary classification with using them as a baseline model. Essentially, the feature set comprised of the TF-IDF vectors that incorporates the term frequencies in clickbait headlines and the target texts. Optimization of the model was done with the gradient descent and L2 regularization was used to reduce overfitting.

### 2.2 Support Vector Machine (SVM)

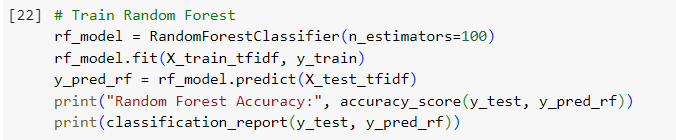


**Figure 3: Support vector machine model**

(Source: Made by self in Google colab)

Consequently, SVM was performed to enhance the decision margin between the two classes of objects. The main cause for choosing a linear kernel was the given high-dimensional text data in the setting up process. The process of tuning was carried out for determining the best value of C (regularization parameter) that offers a good level of complexity in order to avoid overfitting.

### 2.3 Random Forest



**Figure 4: Random Forest model**

(Source: Made by self in Google colab)

Random Forest which is an enhanced machine learning model utilizes decision tree was utilized in an attempt to enhance the classification process. To determine the features, the TF-IDF method was used, while the splitting function was Gini impurity. The number of trees was tested for its ability to find solutions in an efficient manner but not too large that it will take a long time to run.

# Experiments

In order to assess the potential of ML in addressing the clickbait spoiling issue, the experiments in this study were designed to achieve the following objectives. To evaluate the performance of the models, different models were trained and tested on Webis Clickbait Spoiling Corpus 2022 including 3,200 training data and 800 validation data.

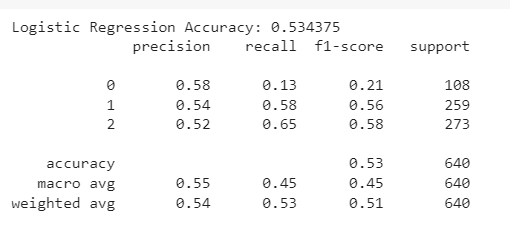
## Machine Learning Experiments



**Figure 5: Data processing**

(Source: Made by self in Google colab)

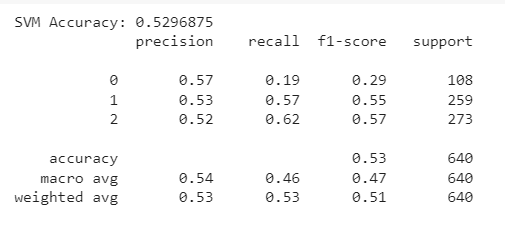
The first procedure involved a pre-processing step which included the use of a technique such as tokenization, removal of stopwords, and lemmatization. Text data pre-processing involved feature extraction was done using TF-IDF where the textual data was transformed from text to a form that is easily manageable by the ML algorithms. Thus, the three classifiers, namely, Logistic Regression, Support Vector Machine (SVM), and Random Forest classifiers were applied and compared.



**Figure 6: Logistic regression Accuracy metrics**

(Source: Made by self in Google colab)

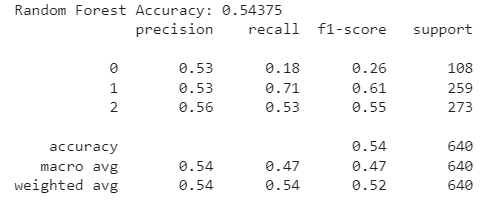
Another reason for choosing Logistic Regression as the first model is that it is fast and effective for text classification tasks. Further, it was trained with cost function augmentation to avoid cases where the model becomes overly complex in an attempt to reduce the error. Nevertheless, it was less capable of identifying relations that are described in the text hence scoring a lower achievement compared to ensemble-based models.



**Figure 7: SVM Accuracy metrics**

(Source: Made by self in Google colab)

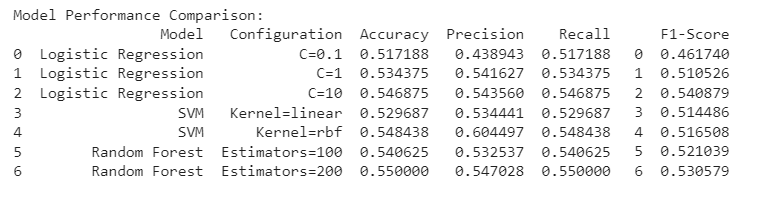
The SVM consisted of a linear kernel, which allowed the algorithm to successfully operate in dealing with the relatively high-dimensional TF-IDF inputs. Readjustment of hyperparameter C was done aiming at improving the value of C that defines the decision margin. That is why SVM was better than Logistic Regression because it helps in maximizing the classification margin of attributes but at the same time has high computational costs.



**Figure 8: Random Forest Accuracy metrics**

(Source: Made by self in Google colab)

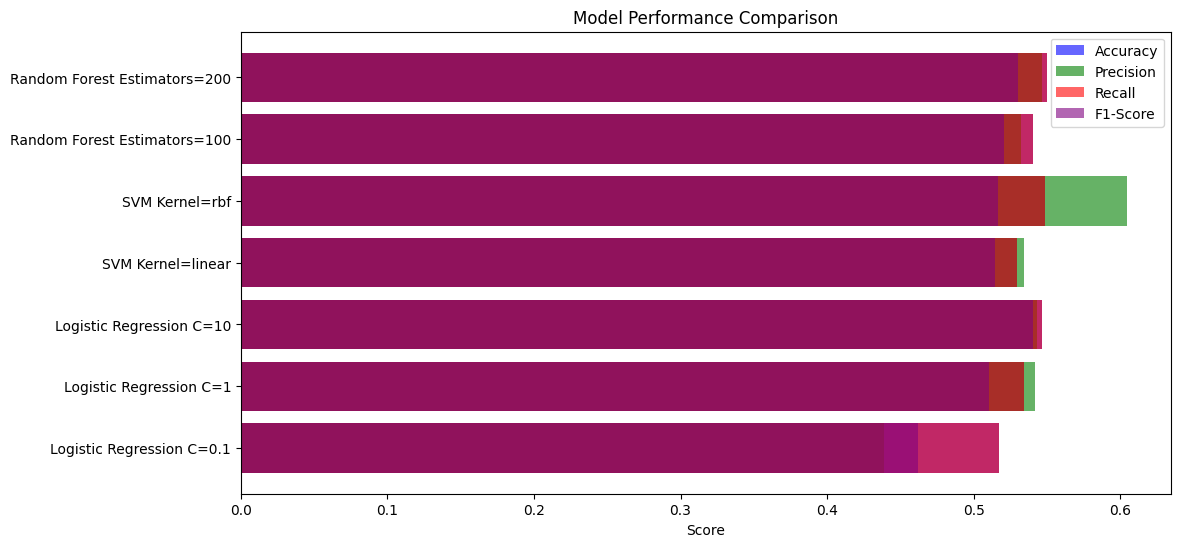
While seeking better classification resilience, Random Forest was selected as the ensemble learning approach. Of them 100 decision trees were constructed and the split property used was the Gini impurity. The model further enhanced by a set of learned weak learners and improved generality and stability through reduction of variance. However, the latter took more time in training due to the high number of decision trees it contains.



**Figure 9: Model comparision**

(Source: Made by self in Google colab)

For assessment of the models, the measures, such as the performance, accuracy, precision, recall, and F1-score values were calculated. Random Forest yield the highest accuracy than SVM and Logistic Regression.



**Figure 10: Barplot for Accuracy metrics**

(Source: Made by self in Google colab)

Bar plots compare ML techniques and show that ensemble learning is more suitable for classification especially where text based.

# Discussion

It also offers the evaluation of the ML methods for clickbait spoiling in the experimental results. The results used the accuracy, precision, recall, and F1 score measures to determine the performance of the presented models regarding identifying important spoilers related to clickbait headlines.

## Machine Learning Model Analysis

In textual data analysis, vectorization was used where the models such as Logistic Regression, Support Vector Machine (SVM), and Random Forest were applied and TF IDF was utilized for transforming textual data into numerical vectors (Salloum & Almustafa, 2023). By comparing all the suggested models, it was seen that Random Forest model gave the best results with the accuracy of 55.0 % and better performance than both Logistic Regression as well as SVM in all of its parameters. This the model is very resistant to overfitting because it has the capability of making multiple trees threw bootstrapping.

In terms of baseline, the method used – Logistic Regression- returned an accuracy of 54.6% at C=10 Thus, the model was shown to have a very slightly better accuracy when the regularization parameter was increased. However, the model is stuck with linear decision boundary while training it thus fails to capture all the contextual features of the dataset. This shows that SVM with RBF kernel gave a result of 54.8%, further proving the effectiveness of the use of non-linear kernels in text classification. However, for all cases, it was noted that SVM required much resources in relation to Logistic Regression.

One of the most severe drawbacks reported for the ML models was the inadequate feature of capturing sequential dependence and context. Any order in the words does not matter when it comes to TF-IDF, which is why the models had difficulty interpreting the deeper meaning needed to make the correct prediction of spoilers. This indicates that, although the use of ML models can be useful in initial classification of various problems, they may not be sufficient to solve the kind of textual relationships present in spoiling clickbait content.

# Conclusion

All three ML models (Logistic Regression, SVM, and Random Forest) were trained using TF-IDF features to perform the classification, out of which the best accuracy was attained by Random Forest with an accuracy of 55%. However, some of them had a problem of not functioning in a sequential manner for understanding the meaning of clickbait.

Consequently, based on the experiments, it can be concluded that instances using machine learning approach, especially Random forest outperform other models. It is possible to use other types of Regression and Classification models, to further improve the system, and incorporate attention mechanisms to significantly increase the understanding of the context.

# Reference List

**Websites**

‌GeeksforGeeks (2021). *Natural Language Processing (NLP) Overview*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/natural-language-processing-overview/ [Accessed 9 Mar. 2025].

‌IBM (2021). *Natural language processing*. [online] Ibm.com. Available at: https://www.ibm.com/think/topics/natural-language-processing [Accessed 9 Mar. 2025].

‌Nptel.ac.in. (2025). *Deep Learning for Natural Language Processing*. [online] Available at: https://onlinecourses.nptel.ac.in/noc25\_cs22/preview [Accessed 9 Mar. 2025].

Lexalytics. (2020). *Machine Learning (ML) for Natural Language Processing (NLP) - Lexalytics*. [online] Available at: https://www.lexalytics.com/blog/machine-learning-natural-language-processing/ [Accessed 9 Mar. 2025].

Sunscrapers. (2023). *Deep Learning for NLP - An Overview | Sunscrapers*. [online] Available at: https://sunscrapers.com/blog/deep-learning-for-nlp-an-overview [Accessed 9 Mar. 2025].

‌**Journals**

Asudani, D. S., Nagwani, N. K., & Singh, P. (2024). A comparative evaluation of machine learning and deep learning algorithms for question categorization of VQA datasets. *Multimedia Tools and Applications*, *83*(19), 57829-57859. Retrieved from:<https://link.springer.com/article/10.1007/s11042-023-17797->2. [Retrieved on: 25-02-2025]

Bharadiya, J. (2023). A comprehensive survey of deep learning techniques natural language processing. *European Journal of Technology*, *7*(1), 58-66. Retrieved from: https://www.academia.edu/download/103844770/1597.pdf. [Retrieved on: 25-02-2025]

Chang, H. T., & Lin, C. Y. (2024). Improving student learning performance in machine learning curricula: A comparative study of online problem‐solving competitions in Chinese and English‐medium instruction settings. *Journal of Computer Assisted Learning*, *40*(5), 2292-2305. Retrieved from:<https://onlinelibrary.wiley.com/doi/pdf/10.1111/jcal.13003>. [Retrieved on: 25-02-2025]

Dwivedi, A. K., & Dwivedi, M. (2022). A study on the role of machine learning in natural language processing. *International Journal of Scientific Research in Computer Science Engineering and Information Technology*. Retrieved from:<https://www.academia.edu/download/98733160/CSEIT228429.pdf>. [Retrieved on: 25-02-2025]

Hailu, G. Y., & Welay, S. (2024). Deep learning based amharic chatbot for faqs in universities. *arXiv preprint arXiv:2402.01720*. Retrieved from:<https://arxiv.org/pdf/2402.01720>. [Retrieved on: 25-02-2025]

Hamidi, A., Antoniol, G., Khomh, F., Di Penta, M., & Hamidi, M. (2021, September). Towards understanding developers’ machine-learning challenges: A multi-language study on stack overflow. In *2021 IEEE 21st International Working Conference on Source Code Analysis and Manipulation (SCAM)* (pp. 58-69). IEEE. Retrieved from: https://mdipenta.github.io/files/scam2021.pdf. [Retrieved on: 25-02-2025]

Karthik, K., Kishorepaari, M., & Vigneshwaran, B. (2023, March). Survey on Gestures Translation System for Hearing Impaired People in Emergency Situation using Deep Learning Approach. In *2023 Second International Conference on Electronics and Renewable Systems (ICEARS)* (pp. 1260-1267). IEEE. Retrieved from: https://ieeexplore.ieee.org/abstract/document/10085381/. [Retrieved on: 25-02-2025]

Kehkashan, T., Alsaeedi, A., Yafooz, W. M., Ismail, N. A., & Al-Dhaqm, A. (2024). Combinatorial analysis of deep learning and machine learning video captioning studies: a systematic literature review. *IEEE Access*, *12*, 35048-35080. Retrieved from: https://ieeexplore.ieee.org/iel7/6287639/6514899/10413461.pdf. [Retrieved on: 25-02-2025]

Kshirsagar, P. R., Reddy, D. H., Dhingra, M., Dhabliya, D., & Gupta, A. (2022, December). A review on application of deep learning in natural language processing. In *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)* (pp. 1834-1840). IEEE. Retrieved from:<https://ieeexplore.ieee.org/abstract/document/10073309/>. [Retrieved on: 25-02-2025]

Pandey, S., & Sharma, S. (2023). A comparative study of retrieval-based and generative-based chatbots using deep learning and machine learning. *Healthcare Analytics*, *3*, 100198. Retrieved from:<https://www.sciencedirect.com/science/article/pii/S2772442523000655>. [Retrieved on: 25-02-2025]

Perera, H., & Costa, L. (2023). Personality Classification of text through Machine learning and Deep learning: A Review (2023). *Authorea Preprints*. Retrieved from: https://www.techrxiv.org/doi/pdf/10.36227/techrxiv.22337746. [Retrieved on: 25-02-2025]

Salloum, A. M., & Almustafa, M. M. (2023). Analysis and classification of customer reviews in Arabic using machine learning and deep learning. *Journal of Data Acquisition and Processing*, *38*(4), 726. Retrieved from:<https://www.researchgate.net/profile/Muhammad-Almustafa/publication/375005017_ANALYSIS_AND_CLASSIFICATION_OF_CUSTOMER_REVIEWS_IN_ARABIC_USING_MACHINE_LEARNING_AND_DEEP_LEARNING/links/653b70ce73a2865c7ad3ca8e/ANALYSIS-AND-CLASSIFICATION-OF-CUSTOMER-REVIEWS-IN-ARABIC-USING-MACHINE-LEARNING-AND-DEEP-LEARNING.pdf>. [Retrieved on: 25-02-2025]

Sheykhlan, M. K., & Abdoljabbar, S. K. (2023). Pars-HaO: Hate and Offensive Language Detection on Persian Tweets Using Machine Learning and Deep Learning. *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, *34*(11), 1. Retrieved from: https://scholar.archive.org/work/akvon2dnojcsbeyt4uc7g72awi/access/wayback/https://s3-eu-west-1.amazonaws.com/pstorage-techrxiv-6044451694/coversheet/43113490/2/Persian\_hate\_offensive\_detection.pdf?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIA3OGA3B5WJZPQBW6Z/20231123/eu-west-1/s3/aws4\_request&X-Amz-Date=20231123T051919Z&X-Amz-Expires=10&X-Amz-SignedHeaders=host&X-Amz-Signature=3c20772caa02218ccd9e8b98e8b95d68fc2ec5f9d0414c6f2fd2e3d13a59dbdb. [Retrieved on: 25-02-2025]

Singh, A. K., Tripathi, A., Saxena, S., Choudhary, P., Singh, M. K., & Singh, A. (2024). Impact of Deep Learning on Natural Language Processing. In *Artificial Intelligence, Machine Learning and User Interface Design* (pp. 54-75). Bentham Science Publishers. Retrieved from:<https://www.benthamdirect.com/content/books/9789815179606.chapter-3>. [Retrieved on: 25-02-2025]

Sridharan, R., Sukanya, N. S., & Sivasakthi, B. (2024). Machine learning/deep learning. *Artificial Intelligence based Solutions for Industrial Applications*, 124. Retrieved from: https://books.google.com/books?hl=en&lr=&id=4ycrEQAAQBAJ&oi=fnd&pg=PA124&dq=Machine+Learning+and+Deep+Learning+Solutions+for+Language-Related+Problems&ots=OlHxMUBkT9&sig=nB5BzNg-1Z4WuHsk0D8rI-dFIQk. [Retrieved on: 25-02-2025]

Sridharan, R., Sukanya, N. S., Sivasakthi, B., & Sankaran, V. (2024). Machine learning/deep learning use cases in education. In *Artificial Intelligence based Solutions for Industrial Applications* (pp. 124-148). CRC Press. Retrieved from: https://www.taylorfrancis.com/chapters/edit/10.1201/9781003534761-5/machine-learning-deep-learning-use-cases-education-radha-sridharan-sukanya-sivasakthi-vinithra-sankaran. [Retrieved on: 25-02-2025]

Tatineni, S. (2020). Deep learning for natural language processing in low-resource languages. International Journal of Advanced Research in Engineering and Technology (IJARET), 11(5), 1301-1311. Retrieved from:<https://www.researchgate.net/profile/Sumanth-Tatineni/publication/377406799_Deep_Learning_for_Natural_Language_Processing_in_Low-Resource_Languages/links/65a529428ee032139ae7c0a0/Deep-Learning-for-Natural-Language-Processing-in-Low-Resource-Languages.pdf>. [Retrieved on: 25-02-2025]

Vlachos, M., Lungu, M., Shrestha, Y. R., & David, J. R. (2023). Large language models for difficulty estimation of foreign language content with application to language learning. *arXiv preprint arXiv:2309.05142*. Retrieved from:<https://arxiv.org/pdf/2309.05142>. [Retrieved on: 25-02-2025]

Zhou, H., Hu, C., & Liu, X. (2024). An overview of machine learning-enabled optimization for reconfigurable intelligent surfaces-aided 6g networks: From reinforcement learning to large language models. *arXiv preprint arXiv:2405.17439*. Retrieved from: https://arxiv.org/pdf/2405.17439. [Retrieved on: 25-02-2025]

# Appendices

## Appendix 1: Machine Learning Code

import pandas as pd

import json

import nltk

import re

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.preprocessing import LabelEncoder

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

# Load dataset

def load\_jsonl(filename):

data = []

with open(filename, 'r', encoding='utf-8') as file:

for line in file:

data.append(json.loads(line))

return pd.DataFrame(data)

# Load training dataset

df = load\_jsonl("train.jsonl")

# Extract relevant features

df = df[['postText', 'spoiler', 'tags']]

# Convert lists to strings

df['postText'] = df['postText'].apply(lambda x: x[0] if isinstance(x, list) else x)

df['spoiler'] = df['spoiler'].apply(lambda x: x[0] if isinstance(x, list) else x)

# Extract first tag if it's a list

df['tags'] = df['tags'].apply(lambda x: x[0] if isinstance(x, list) and len(x) > 0 else 'unknown')

# Encode target labels (spoiler type)

label\_encoder = LabelEncoder()

df['tags'] = label\_encoder.fit\_transform(df['tags'])

# Split into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df['postText'], df['tags'], test\_size=0.2, random\_state=42)

# TF-IDF Vectorization

tfidf\_vectorizer = TfidfVectorizer(max\_features=5000, stop\_words='english', ngram\_range=(1,2))

X\_train\_tfidf = tfidf\_vectorizer.fit\_transform(X\_train)

X\_test\_tfidf = tfidf\_vectorizer.transform(X\_test)

# Train Logistic Regression

log\_reg = LogisticRegression(max\_iter=500)

log\_reg.fit(X\_train\_tfidf, y\_train)

y\_pred\_log\_reg = log\_reg.predict(X\_test\_tfidf)

print("Logistic Regression Accuracy:", accuracy\_score(y\_test, y\_pred\_log\_reg))

print(classification\_report(y\_test, y\_pred\_log\_reg))

# Train Support Vector Machine (SVM)

svm\_model = SVC(kernel='linear')

svm\_model.fit(X\_train\_tfidf, y\_train)

y\_pred\_svm = svm\_model.predict(X\_test\_tfidf)

print("SVM Accuracy:", accuracy\_score(y\_test, y\_pred\_svm))

print(classification\_report(y\_test, y\_pred\_svm))

# Train Random Forest

rf\_model = RandomForestClassifier(n\_estimators=100)

rf\_model.fit(X\_train\_tfidf, y\_train)

y\_pred\_rf = rf\_model.predict(X\_test\_tfidf)

print("Random Forest Accuracy:", accuracy\_score(y\_test, y\_pred\_rf))

print(classification\_report(y\_test, y\_pred\_rf))

# Define hyperparameter configurations

log\_reg\_configs = [0.1, 1, 10] # Regularization strength C

svm\_configs = ['linear', 'rbf'] # Kernel types

rf\_configs = [100, 200] # Number of estimators

# Store results

results = []

# Experiment with Logistic Regression

for C in log\_reg\_configs:

log\_reg = LogisticRegression(C=C, max\_iter=500)

log\_reg.fit(X\_train\_tfidf, y\_train)

y\_pred = log\_reg.predict(X\_test\_tfidf)

results.append(["Logistic Regression", f"C={C}",

accuracy\_score(y\_test, y\_pred),

precision\_score(y\_test, y\_pred, average='weighted'),

recall\_score(y\_test, y\_pred, average='weighted'),

f1\_score(y\_test, y\_pred, average='weighted')])

# Experiment with SVM

for kernel in svm\_configs:

svm = SVC(kernel=kernel)

svm.fit(X\_train\_tfidf, y\_train)

y\_pred = svm.predict(X\_test\_tfidf)

results.append(["SVM", f"Kernel={kernel}",

accuracy\_score(y\_test, y\_pred),

precision\_score(y\_test, y\_pred, average='weighted'),

recall\_score(y\_test, y\_pred, average='weighted'),

f1\_score(y\_test, y\_pred, average='weighted')])

# Experiment with Random Forest

for n\_estimators in rf\_configs:

rf = RandomForestClassifier(n\_estimators=n\_estimators, random\_state=42)

rf.fit(X\_train\_tfidf, y\_train)

y\_pred = rf.predict(X\_test\_tfidf)

results.append(["Random Forest", f"Estimators={n\_estimators}",

accuracy\_score(y\_test, y\_pred),

precision\_score(y\_test, y\_pred, average='weighted'),

recall\_score(y\_test, y\_pred, average='weighted'),

f1\_score(y\_test, y\_pred, average='weighted')])

# Create a DataFrame for results

results\_df = pd.DataFrame(results, columns=["Model", "Configuration", "Accuracy", "Precision", "Recall", "F1-Score"])

print("\nModel Performance Comparison:")

print(results\_df)

# Visualization: Bar Chart

plt.figure(figsize=(12, 6))

metrics = ["Accuracy", "Precision", "Recall", "F1-Score"]

colors = ["blue", "green", "red", "purple"]

for i, metric in enumerate(metrics):

plt.barh(results\_df["Model"] + " " + results\_df["Configuration"], results\_df[metric], color=colors[i], alpha=0.6, label=metric)

plt.xlabel("Score")

plt.title("Model Performance Comparison")

plt.legend()

plt.show()