algorithms

MDPI

Article

Social Media Hate Speech Detection Using Explainable Artificial Intelligence (XAI)

Harshkumar Mehta and Kalpdrum Passi *

School of Engineering and Computer Science, Laurentian University, Sudbury, ON P3E 2C6, Canada
* Correspondence: kpassi@laurentian.ca

Abstract: Explainable artificial intelligence (XAI) characteristics have flexible and multifaceted potential in hate speech detection by deep learning models. Interpreting and explaining decisions made by complex artificial intelligence (AI) models to understand the decision-making process of these model were the aims of this research. As a part of this research study, two datasets were taken to demonstrate hate speech detection using XAI. Data preprocessing was performed to clean data of any inconsistencies, clean the text of the tweets, tokenize and lemmatize the text, etc. Categorical variables were also simplified in order to generate a clean dataset for training purposes. Exploratory data analysis was performed on the datasets to uncover various patterns and insights. Various pre-existing models were applied to the Google Jigsaw dataset such as decision trees, k-nearest neighbors, multinomial naïve Bayes, random forest, logistic regression, and long short-term memory (LSTM), among which LSTM achieved an accuracy of 97.6%. Explainable methods such as LIME (local interpretable model—agnostic explanations) were applied to the HateXplain dataset. Variants of BERT (bidirectional encoder representations from transformers) model such as BERT + ANN (artificial neural network) with an accuracy of 93.55% and BERT + MLP (multilayer perceptron) with an accuracy of 93.67% were created to achieve a good performance in terms of explainability using the ERASER (evaluating rationales and simple English reasoning) benchmark.

Keywords: explainable artificial intelligence; hate speech detection; offensive languages; LIME; BERT; neural networks

Citation: Mehta, H.; Passi, K. Social Media Hate Speech Detection Using Explainable Artificial Intelligence (XAI). *Algorithms* **2022**, *15*, 291. https://doi.org/10.3390/a15080291

Academic Editor: Ulrich Kerzel

Received: 29 June 2022
Accepted: 11 August 2022
Published: 17 August 2022

1. Introduction

Artificial intelligence has invaded various fields in the present times. Be it science, education, finance, or business, artificial intelligence has found its applications everywhere. However, currently, AI is limited to only its subset "machine learning" and has not even realized its full potential. Machine learning is the ability of computers to learn the relationship between input and output without being explicitly programmed. Thus, in machine learning, in contrast to traditional programming which requires writing algorithms, it is required to find the algorithm that learns patterns from a given dataset and builds a predictive model, on the basis of which the computer learns the patterns between input and output. The machine learning model is now able to give predictions on new and unseen data. However, these models do not provide an explanation as to how different features contribute to the output. Thus, the functioning of artificial intelligence is traditionally like a black box. This characteristic may not provide justifications in critical scenarios such as diagnosis of life-threatening diseases and defense. If there is an explanation given along with the output, combined with human reasoning, it may prove significantly useful. This forms the basis of explainable artificial intelligence (XAI). XAI gives answers to many questions along with the output. It is an emerging area of research and has found applications in varied fields.

Artificial intelligence is implemented as a "black box" that just gives the output after a certain input but how it is achieved is not revealed. While it may not be necessary to get the

reason behind the output in several cases, in fields such as medical research, the knowledge of answers to questions such as "how" and "why" becomes essential. If we do not know answers to when the model fails or succeeds, how to detect errors and correct them, etc., it may have serious implications. It may even raise questions on the efficacy of the model.

1.1. Need for Explainability

XAI is necessary if the users are to understand the AI results, trust the decisions of the algorithms, and manage the results in an organized manner. Regulatory considerations and ethical concerns are important to incorporate AI in day-to-day human transactions. Explainable AI plays an important part in instilling trust in the AI regulators and business partners for commercially beneficial and ethically viable decision making. Primarily, in critical scenarios such as medical research, explainability may lead to better trust in the model. If there is an explanation of the result and how the model generates insights, this reasoning coupled with human knowledge and reasoning may significantly improve results and provide effective applications. This can also help prevent errors in situations in which there is no scope for them. Thus, explainable AI (or XAI) is a new dimension of artificial intelligence where we can seek answers to "why" questions which is not possible traditionally. XAI has varied applications in healthcare, law and order, defense, etc. As mentioned, XAI is an emerging field of research.

1.2. Motivation

Artificial intelligence is implemented as a "black box" that just gives the output after a certain input but how it is achieved is not revealed. Machine learning has seen applications in various fields such as medical, research, business, education, industry, chatbots, recommendation systems, and even self-driving cars. However, some machine learning models may not be intuitive or transparent, which may be complex for people to understand. In such cases, these models may lose their effectiveness. In the past few years, deep learning models have also presented state-of-the-art results in many situations. However, deep learning models are still not able to justify whether they are making the right decision or not. XAI methods provide explanations that can be translated by humans without having a depth of knowledge in deep learning models. XAI characteristics have flexible and multifaceted potential in hate speech detection by the deep learning models. XAI, thus, provides a strong interconnection between an individual moderator and hate speech detection framework, which is a pivot for the research study in interactive machine learning. As the model becomes complex with an increased number of parameters, iterations, and optimization, it becomes even more difficult to validate the results by the model. The main goal and the intended contribution in this paper are interpreting and explaining decisions made by complex artificial intelligence (AI) models to understand their decision-making process in hate speech detection. For this purpose, pre-existing models were applied on the Google Jigsaw dataset to get the best prediction accuracy, and explainable methods such as LIME (local interpretable model—agnostic explanations) were applied to the HateXplain dataset. Variants of BERT (bidirectional encoder representations from transformers) model such as BERT + ANN (artificial neural network) and BERT + MLP (multilayer perceptron) were created to achieve a good performance in terms of explainability using the ERASER (evaluating rationales and simple English reasoning) benchmark (DeYoung et al. (2019)).

1.3. Literature Review

There has been recent research on hate speech detection using traditional natural language processing (NLP) techniques and using machine learning methods [1–3]. The extraction of text-, user-, and network-based features and characteristics identifying bullies has been successful [4]. Furthermore, abusive language detection including hate speech keyword identification, sexism, bullying, trolling, and racism were studied in [1,5–7] using deep learning techniques.

In recent times, there has been an increased interest in the explanability of artificial intelligence techniques including machine learning and deep learning methods to understand the reasons for labeling text with hate speech or other social media and medical applications. A novel explanation method based on LIME [3,8] for the explanation of predictions made by a classifier was proposed [9], and the best practices for the usage of these interpretable machine learning models and their applications were also discussed [10–14]. Deep learning and active learning approaches were used for explanability in [8,15–17].

Explainable AI (XAI) has become very popular in recent times to unravel the secrets of decision making by AI techniques. There have been novel definitions of explainable machine learning and deep learning [18], with a categorization of XAI techniques and methods based on factors such as their scope, methodology, algorithmic intuition, and explanation capability [19]. XAI models available out there, such as LIME, layer-wise relevance propagation, and DeepLIFT and how they are deployed were discussed in [20–23]. XAI has been applied in various applications such as the predictive maintenance (PdM) scenario in manufacturing [24] and social science research [25].

Table 1 gives a comprehensive explanation with contributions, findings, and limitations of these works.

Table 1. Summary of literature.

Ref.	Contribution	Key Findings	Limitation(s)
[1]	Automated hate speech detection and the problem of offensive language	Logistic regression, naïve bayes, decision trees, random forests, and SVM are tested using 5-fold cross-validation	The definition of hate speech in this research is limited to language that threatens or incites violence which excludes a large proportion of hate speech. Lexical methods used are inaccurate at identifying hate speech, and only a small percentage of tweets flagged by hate base lexicon are considered hate speech.
[2]	Detection of offensive content and identification of potential offensive users	Lexical syntactical feature (LSF) framework	Comparison of existing text-mining methods in detecting offensive contents with LSF framework in not detailed and lacks scientific validation.
[3]	A feature attribution method for explainability	Necessity and sufficiency explained in detailed in the context of hate speech	The analysis is limited by limitation of the existing dataset used which lacks variety of demographic groups.
[4]	Detecting bullying and aggressive behavior on Twitter	Random forest classifier using WEKA tool, 10-fold cross-validation	Results obtained with random forest classifier are only presented with respect to training time and performance due to limited space.
[5]	A unified deep learning architecture for abuse detection	Deep learning architecture for detection of abuse online	Network-related metadata are not considered in the dataset due to time limitations as it takes a significant amount of computation to crawl Twitter data due to Twitter API rate limits.
[6]	A unified approach to explaining complex ML models	SHAP (Shapley additive explanations) framework for the explanation of complex, ensemble and deep learning models	SHAP model is not consistent with human intuition in some cases, which can lead to false positives or false negatives; a different approach is not considered in such cases.
[7]	Explanation of RNN predictions in sentiment analysis	Propagation rule for growing connections in recurrent neural networks (RNN) architectures	Gradient-based sensitivity analysis used with approach is not able to get accurate relevance score when a sentiment is decomposed into words.
[8]	Intuitive explainability along with using various deep learning techniques	LIME explanation with individual examples	Some misclassification is observed in the case of nontoxic comments.

Table 1. *Cont.*

Ref.	Contribution	Key Findings	Limitation(s)
[9]	Explaining the predictions of any classifier	LIME model to explain the predictions of any classifier, SP-LIME model for selecting representative and nonredundant explanations	The method to perform the pick-up step for images is not addressed in this research.
[10]	Interpretable machine learning models	Technical foundations of explainable artificial intelligence, presentation of practical XAI algorithms such as occlusion, integrated gradients, and LRP, importance, applications, challenges and directions for future work	The explanation revealed by model in this research are difficult to interpret by human observer due to limited accessibility of the data representation. Deeper understanding of relevance maps is not obtained by the model.
[11]	Evaluation of interpretability and explainability in machine learning	Application-grounded, human-grounded, and functionally grounded approaches for evaluation of interpretability, discussion of open questions related to these evaluation approaches	The research is focused only on the taxonomy to define and evaluate interpretability and not on methods to extract explanations.
[12]	Framework for the explanation of the results of an artificial intelligence system	Proposed framework named "teaching explanations for decisions (TED)" to provide explanations of an AI system	The proposed TED framework assumes a training dataset to be having explanation and applies cartesian product using any machine learning algorithm to train classifier instead of using multitask setting.
[13]	Explainability of deep neural network models	Key directions for moving towards transparency of machine learning models, novel technological development for explainability	This research does not focus on exact choice of deep neural network for any particular domain and instead is only focused on generalized conceptual developments.
[14]	Overview of interpretability of machine learning models	Need for diverse metrics for targeted explanations, suggestions for explainability of deep learning models	The study only focuses on abstract overview of explainability without diving deep into explanation metrics.
[15]	Enhancing interpretability of tree-based machine learning models	Method for computation of the game theoretic Shapley values, local explanation method, tools for explainability using a combination of local explanation methods	Only local explanations are presented that focuses on single samples without considering global explanations.
[16]	A unified framework for machine learning interpretability	An open-source package InterpretML for glass-box and blackbox explainability	Computational performance for models across datasets is not consistent for explainable boosting machine (EBM) model discussed in this research.
[17]	An active learning approach for labeling text	Attention network visualization for indirect and informal communication	The semantic embeddings and lexicon expansion techniques discussed in the paper lack detailed explanations.
[18]	Explainable artificial intelligence (XAI): categorization, contributions, suggestions, and issues in responsible AI	Overview of explainable artificial intelligence, literature review and taxonomy, implications, vision, and future of XAI	Some functions are proprietary and are not exposed to the public in this research. Explainable AI methods give explanations that are not aligned with what the original method calculates.
[19]	Opportunities and challenges in explainable artificial intelligence (XAI)	Survey on seminal algorithms for explainable deep neural network algorithms, evaluation of XAI methods and techniques	Human attention is not able to arrive at XAI explanation maps for decision making. Quantitative measures of completeness and correctness of the explanation map are not available

Table 1. *Cont.*

Ref.	Contribution	Key Findings	Limitation(s)
[20]	Evaluation of explainable artificial intelligence models for convolutional neural networks (CNN) with proxy tasks	Proposed two 2 proxy tasks, namely, pattern task and Gaussian blot task, which are then used to evaluate LIME, layer-wise relevance propagation, and Deep LIFT, and results are discussed	The evaluation scheme discussed in the research has issues with cross-model evaluation and is less comprehensive.
[21]	Discussion of various explainable AI techniques	Need for XAI, key issues in XAI, objectives and scope of XAI, survey on various XAI techniques and methodologies	The study focuses on XAI and its importance but fails to discuss the limitations of conventional AI and its combination with XAI.
[22]	Fuzzy systems for explainable artificial intelligence	Need, timeline, applications, and future work of fuzzy systems for XAI	The research fails to address how to arrive at a solution to the problems that are not measurable in the evolutionary fuzzy systems (EFS) patterns.
[23]	A literature survey on explainable artificial intelligence (XAI) terminology	Background, terminology, objectives of explainable artificial intelligence (XAI), natural language generation approach	The survey does not explain how to evaluate natural language generation (NLG).
[24]	Predictive maintenance case study based on explainable artificial intelligence (XAI)	A machine learning model based on a highly efficient gradient boosting decision tree is proposed for the prediction of machine errors or any tool failure.	Results of this research are presented using a generic dataset and not a real data; however, the presented concept shows high maturity with promising results.
[25]	Insights from social sciences related to explainable artificial intelligence (XAI)	Why questions are diversified in explainable AI, explanations are biased and social	Adopting the work of this research into explainable AI is not a straightforward step, and the models discussed need to be refined and extended to provide good exploratory agent.

2. Materials and Methods

We used two datasets for hate speech detection using explainable artificial intelligence, and these datasets are discussed in this section. Both datasets include text written in English language. The Jigsaw dataset is used with some linear (e.g., decision trees) and some complex models (e.g., LSTM) to visualize how they compare with each other on a hate speech dataset. The Google Jigsaw dataset comprises user discussions from talk pages of English Wikipedia, and various existing semi-interpretable linear models were trained on it. The Jigsaw dataset does not have human annotations unlike the HateXplain dataset; hence, it is not possible to evaluate the ERASER benchmark on it. The HateXplain dataset contains posts from Twitter and Gab and is annotated, which makes it suitable for evaluating the ERASER benchmark for explainability.

2.1. Google Jigsaw Dataset

The first dataset that we used is a dataset released by Google Jigsaw as part of a Kaggle challenge. The dataset contains the following columns: comment, toxic, severe_toxic, obscene, threat, insult, and identity_hate. The dataset comprises discussions from Wikipedia. The labels in the dataset can be multinomial, i.e., a particular text can belong to two or more classes. Table 2 shows the Google Jigsaw dataset details.

2.2. HateXplain Dataset

The second dataset used is the HateXplain dataset which contains posts from Twitter and Gab. Combining these two sources, we obtained a dataset that contains over 20,000 data containing hateful, offensive, and normal text as labels.

From Twitter, we randomly took 1% of tweets from the period between January 2019 to June 2020. From Gab, we took the dataset provided in [26]. Reposts of the tweets were not considered, and the duplicates were removed. This ensured that the tweets contained only textual data. However, emojis were kept as they contribute significantly to emotion detection. Moreover, all usernames were removed, and a token <user> was inserted in their place. Table 3 shows the HateXplain dataset details.

Table 2. Google Jigsaw dataset details.

Classification	Frequency
Clean	201,081
Toxic	21,384
Obscene	12,140
Insult	11,304
Identity hate	2117
Severe toxic	1962
Threat	689

Table 3. HateXplain dataset details.

	Twitter	Gab	Total
Hateful	708	5227	5935
Offensive	2328	3152	5480
Normal	5770	2044	7814
Undecided	249	670	919
Total	9055	11,093	20,148

2.3. Extracting the Dataset

The datasets taken were in the CSV (comma-separated values) format. A CSV file stores tabular data in plain text separated by commas. Each line of a CSV file corresponds to one row of the dataset, the first row of the file being the header row or the row that contains the column or attribute names. The CSV format files were loaded into a data frame using the Pandas library of Python. Pandas are used for data analysis and manipulation and are extensively used for data science and machine learning use cases.

2.4. Data Preprocessing and Cleaning

Preprocessing of data is a crucial step that impacts a model's performance. The data obtained from Twitter or online sources are noisy and can have null or missing values, images, audio, video, etc. Preprocessing ensures that the data are cleaned, free from noise, and meaningful. However, we did not perform preprocessing on BERT-based models as these are pretrained language representation models. Moreover, BERT uses every information in a sentence including punctuation and stop words. For models not based on BERT, we used Python's various libraries and functions for data preprocessing and cleaning for this research project.

A summary of the steps performed for preprocessing and cleaning of the dataset is given below.

1. Rows with missing labels were dropped as they do not contribute to the learning process.
2. Using the natural language toolkit (NLTK) library, tokenization was performed, i.e., tokens of the sentences were created.
3. Stop words (if, then, the, and, etc.) were removed to keep only the text that would contribute to the learning process.

Data cleaning is an essential step before training the model as it provides various benefits. Data cleaning removes any incorrect or inconsistent information that improves data quality. Figure 1 shows the common steps performed in data cleaning. It includes the removal of unwanted observations followed by correcting any structural errors that

the observations in the dataset might have. The notion of "structural error" indicates any irregularities with the structure of the sentence such as typos in the name of features, the same attribute with a different name, mislabeled classes, additional spaces, and newline characters. The next steps are performed with an aim to manage unwanted outliers such as additional spaces, which is followed by handling any missing data in the dataset. The detailed steps performed for data cleaning are mentioned below [27].

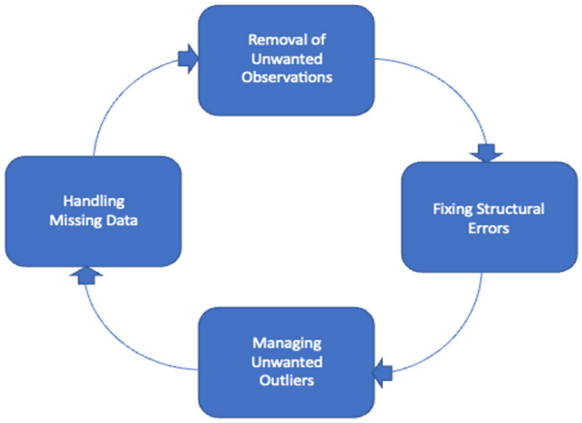

Figure 1. Data cleaning.

1. Firstly, a regular expressions module was imported to help with data cleaning tasks. Regular expressions are sequences of characters that are used for matching with other strings in search. Patterns and strings of characters can be searched using regular expressions. Python has a "re" module that can help to find patterns and strings using regular expressions. Regular expressions can be used to remove or replace certain characters as part of data cleaning and preprocessing.
2. Any newline characters or additional spaces were removed.
3. Any URLs were also removed as they do not contribute to the learning process.
4. Similarly, any other alphanumeric characters that included punctuation were removed for the same reason, including the following strings: !"#$%&\'()*+,-./:;<=>?@[\\]^_'{|}~. Only uppercase and lowercase letters along with digits 0–9 were kept.
5. Stopwords such as "the", "and", "then", and "if" were also removed as they are not a part of the learning process. Python's NLTK library has stopwords in about 16 different languages. We imported English stopwords to remove them from our dataset. These words were removed as they do not add any additional information to the learning process.
6. The outputs of these tasks were stored in a separate column, resulting in a column of tokenized words.

2.5. Tokenization, Sentence Padding, and Lemmatization

Tokenization is the process in which sentences are divided into smaller parts that are called tokens. These tokens serve as the basis for stemming and lemmatization and can aid in finding various patterns in the text. The natural language toolkit (NLTK) library of Python provides functions to perform word tokenization. Specifically, word tokenization can be conducted to yield either characters or subwords. For example, the word "clearer" can be either tokenized into "clear" and "er" or "c-l-e-a-r-e-r". In this study, we performed character tokenization that converts words into tokens as an array of integers, improving the efficiency of the learning process. We created a tokenizer object from a pretrained model that was imported and then fitted to the HateXplain dataset. This was achieved using the keras and TensorFlow libraries.

Padding was performed so that all the inputs were of equal length. Neural networks require all inputs to be of same length. Originally, the raw text had words and sentences of different lengths. In exploratory data analysis, we observed that the maximum sentence

length was mostly 200. Thus, we trimmed sentences with lengths greater than 200 and padded the remaining sentences.

Using natural language processing (NLP), word normalization was performed through lemmatization. In lemmatization, all words are reduced to their base/root forms. For instance, (1) go, going, gone, and goes are reduced to go, (2) read, reading, and reads are reduced to read, and (3) hated, hating, and hates are reduced to hate.

2.6. Simplification of Categorical Values

The original dataset had seven columns: "unnamed", "count", "hate_speech", "offensive language", "neither", and "tweet". To simplify the dataset for an efficient training and learning process, only three columns were kept: text, category, and label. The "tweet" column was converted to a "text" column. The label was derived from the class column in the original dataset, and the category label had values of 0, 1, and 2 encoded from the columns (hate_speech, offensive language, and neither) in the existing dataset. In this column, 0 represents hate_speech, 1 represents offensive_language, and 2 represents neither. Thus, the new and final dataset for the training and learning process had three columns: text, category, and label.

2.7. Exploratory Data Analysis (EDA)

EDA is the process of investigating data and drawing out patterns and insights [28]. EDA helps one understand the data better. It helps in understanding the various attributes in the dataset and how the various attributes contribute to the target variable, identifying anomalies. EDA also reveals any inconsistent or incomplete data. EDA serves as the basis of the data cleaning and preprocessing step. EDA helps with matching assumptions and intuitions with reality. Thus, EDA is a crucial step to intelligently proceed with the subsequent steps in the entire process of machine learning. Figure 2 captures the essence of exploratory data analysis.

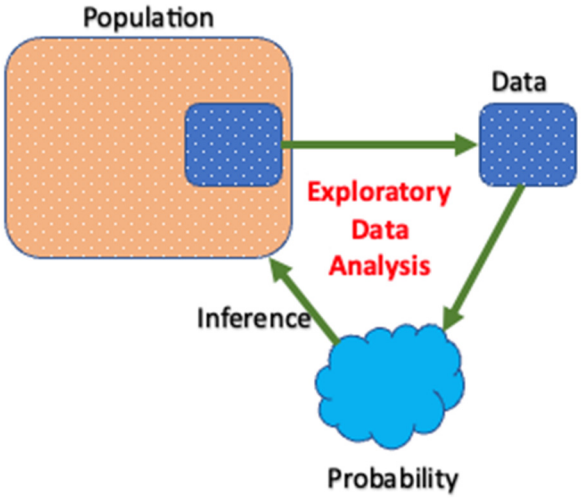

Figure 2. Exploratory data analysis.

2.8. Feature Extraction Methods

After the data are cleaned and preprocessed, they should be converted into a form that the model can understand. For this, all variables must be converted into numerical form. This process is called feature extraction or vectorization. This process also contributes to dimensionality reduction and, hence, helps with feature extraction, to keep only the features that improve the accuracy of the model. Feature extraction can be performed using methods. The importance of the words occurring in the dataset can be gauged, and redundant data can be removed. New features can also be formed from existing ones. Through such methods, features that matter and new features can be generated to form a

better version of the original dataset. We used Count Vectorizer in this research, which is used for converting text into a vector [29]. The TF-IDF (term frequency-inverse document frequency) statistic examines the relevance of a word to a document in a collection of documents. This is accomplished by multiplying two metrics: the number of times a word appears in a document and the word's inverse document frequency over a collection of documents. It has a variety of applications, including automatic text analysis and scoring words in machine learning techniques for natural language processing (NLP).

2.9. Classification Methods and Explainable Techniques

Different classifiers were used to predict hate speech on the Google Jigsaw data set, namely, artificial neural network (ANN) [29], multilayer perceptron (MLP) [30], decision trees, KNN, random forest, multinomial naïve Bayes, logistic regression, and long short-term memory (LSTM). Explainability was described on the HateXplain Dataset using BERT and LIME. We briefly discuss LSTM, BERT, and LIME in this section.

2.9.1. Deep Learning Model—Long Short-Term Memory (LSTM)

LSTM is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process not only single data points, but also entire sequences of data.

The input layer of LSTM was designed with 30,000 × 128 size (or 3,840,000 parameters) in order to incorporate the whole dataset comments as shown in Table 4. After lemmatizing, tokenizing, and removing stop words and punctuation marks, the top 30,000 words were taken for processing. Alphabets and numbers can be represented uniquely using 7 bit ASCII code, with 27 = 128. This layer inputs the tokenized words and fetches 3,840,000 entities from it.

Table 4. LSTM model on the Google Jigsaw dataset.

Layer Type	Output Shape	# of Parameters
Embedding	(None, none, 128)	3,840,000
LSTM 1	(None, none, 128)	131,584
LSTM 2	(None, 128)	131,584
Dense	(None, 6)	774

The function of dropout layers is to reduce the number of entities read, as well as to increase the number of features to be extracted from the input. The standard rate of dropout in LSTM is 0.2 (learning rate). The number of parameters (131,584) shows that, after the recurrence layers, the number of entities was reduced from 3,840,000 to 131,584. The dense layer outputs 774 units (roughly equal to 128 × 6) in order to tell which input word belongs to which class.

The data were divided into a 70–30 split. where 70% of the data were utilized for training and 30% were utilized for testing purposes. After that the model defined above was compiled with the loss function as binary cross-entropy and the Adam optimizer. Then, the model was fit on the training data with a batch size of 128.

The accuracy obtained by the LSTM model was 97.6%, the precision was 0.85, the recall was 0.83, the macro F1-score was 0.84, and the specificity was 0.82.

2.9.2. BERT (Bidirectional Encoder Representation from Transformers)

The BERT model is a relatively new language model that was presented in a paper by Google in 2018 [31]. This model has presented state-of-the-art results in natural language processing. The key feature of BERT is the bidirectionality of the model. The BERT model makes use of the encoder component of the transformer to furnish the representation of words. BERT is used for the creation of language representation models that can serve various purposes.

BERT has a base layer of "knowledge" that is derived from its pretraining. From this base layer of "knowledge", BERT can further be trained to adapt to specifications provided. BERT's transformer processes any given word with respect to the word's relation to all other words in that particular sentence. This enables BERT to understand the context of the word after looking at all surrounding words, unlike other models that understand the meaning of a word in one dimension only. There is another BERT variant that was trained on specifically hate speech detection task called AngryBERT [32]. AngryBERT jointly learns hate speech detection with emotion classification. It can outperform standard BERT in some hate speech tasks. However, the objective of this research was to detect hate speech along with explainable AI to evaluate how explainable the current high-performing black-box algorithms can be. Therefore, standard BERT was applied rather than AngryBERT so as to not only learn the hate speech pattern using the standard BERT variant but also consider the cases where correctly identifying hate speech is difficult for machines (e.g., sarcasm), enabling the recipient of the explanations to make better decisions.

BERT uses the following two semi-supervised models for pretraining [33]:

1. Masked language model (MLM): In this task, BERT learns a featured representation for each of the words present in the vocabulary. About 85% of the words are used for training, and the remainder are used for evaluation. The selection of the training and evaluation sets is random and in iterations. Through this process, the model learns featured representation in a bidirectional way i.e., learns both the left and the right contexts of the words. In this task, some of the tokens from each sequence are replaced with the token [Mask]. The model is trained to predict these tokens using other tokens from sequence.
2. Next sentence prediction (NSP): In this task, BERT learns the relationship between two different sentences. This task contributes to aspects such as question answering. The model is trained to predict the next sentence. It is similar to the textual entailment task where there are two sentences; it is a binary classification task to predict whether the second sentence succeeds the first sentence.

2.9.3. Local Interpretable Model—Agnostic Explanations (LIME)

LIME is an acronym for local interpretable model—agnostic explanations. Each portion of the name represents something we want to be able to explain. Local fidelity refers to the need for the explanation to accurately reflect the classifier's behavior "around" the instance being predicted. This explanation is pointless unless it is interpretable, i.e., if it can be understood by a person. LIME is an agnostic model as it is capable of giving explanations for the predictions of a supervised learning model. LIME can be used with all types of data, be it text, images, or videos. LIME provides local interpretable explanations by computing important features and attributes for a given data point. It works by providing weights to the data rows and, using feature selection techniques, it obtains the important features. LIME is especially successful in explainable artificial intelligence (XAI). It can also be applied to all types of data and in all domains. LIME is a concrete implementation of local surrogate models. Surrogate models are trained to approximate the prediction of underlying black box model. Methods such as SHAP (Shapley additive explanations), counterfactual explanations, and other language interpretability tools can be used to explain black-box models. However, the reason for using LIME is that it uses Lasso or short trees, which results in explanations being selective and concise, thus representing more human-friendly explanations. In social media arbitration, the recipient of explanations is often a layman or someone with very little time. Figure 3 shows an example of explanation by LIME [34].

First and foremost, we provide a discussion on interpretability. Some classifiers employ representations that are completely unfamiliar to consumers (e.g., word embeddings). LIME describes those classifiers in terms of interpretable representations (words), even if that is not the representation actually used by the classifier. Furthermore, LIME considers human constraints, such as the length of explanations.

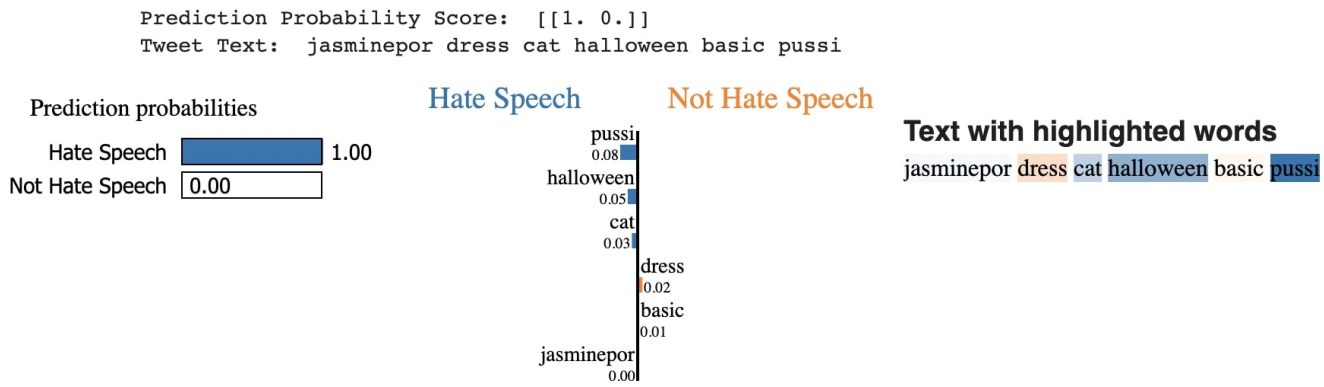

Figure 3. LIME.

Model agnosticism refers to LIME's ability to provide justification for any form of supervised learning model prediction. This method can be used with any type of data, including images, text, and video. LIME can handle any supervised learning model and provide reasoning in this way. LIME generates local optimal explanations by computing essential features in the immediate neighborhood of the instance to be explained. LIME cannot peek inside the model in order to be model agnostic. We disrupt the interpretable input around its neighborhood to check how the model's predictions respond in order to figure out the sections contributing to the prediction. The perturbed data points are then weighted according on their proximity to the original example, and an interpretable model is learned on the basis of those and the related predictions. It generates 5000 samples of the feature vector by default, all of which follow normal distributions. It discovers the target variables for samples whose decisions are explained by LIME after producing normally distributed samples. It allocates weights to each of the rows according to how close they are to the original samples after getting the locally created dataset and their predictions. Then, it extracts relevant features using a feature selection technique such as Lasso or PCA (principal component analysis). In the field of XAI, LIME has found great success and support, and it has been used for text, image, and tabular data. By tweaking the inputs, LIME observes the changes that happen in predictions. LIME generates a new dataset using inputs with variations and their corresponding predictions generated through a black-box model. On this dataset, LIME trains an explainable model with weights generated through the proximity of the instances generated. The model that is trained achieves a good local approximation, giving rise to the name local interpretable explanations. The explainable model trained for an instance minimizes loss and measures the proximity of the explanation to the prediction while keeping the model complexity low. LIME optimizes the loss part, and the user specifies the complexity of the model. LIME is applicable and expandable to all key machine learning fields, which is a noteworthy feature. Embeddings and vectorization of a given word or sentence can be considered a basic unit for sampling in the domain of text processing. In the case of an image, segmented chunks of the image are used as input samples.

3. Results

3.1. Model Training and Evaluation for Google Jigsaw Dataset

The results of all the models on the Google Jigsaw dataset, evaluated in terms of their accuracy, precision, and macro F1-score, are shown in Figure 4. Table 5 gives the scores of the evaluation metrics. It can be observed that LSTM was the best-performing model with an accuracy of 97.6%, closely followed by multinomial naïve Bayes with an accuracy of 96% and logistic regression with an accuracy of 97%. Random forest showed the highest precision of 90%, followed by the KNN classifier with a precision of 88%.

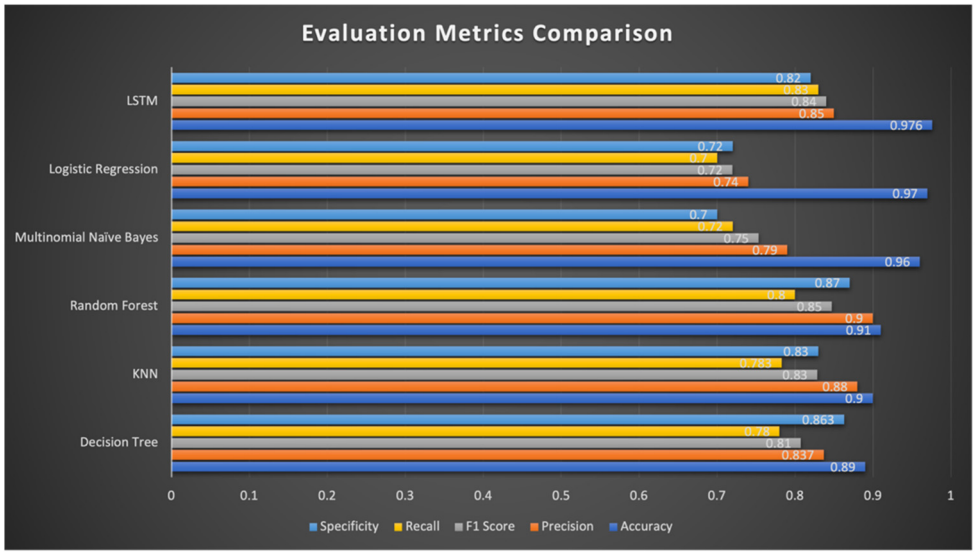

Figure 4. Result summary of all classification models on the Google Jigsaw dataset.

Table 5. LSTM model on the Google Jigsaw dataset.

Classifier Name	Accuracy	Precision	F1-Score	Sensitivity/Recall	Specificity
Decision Tree	0.89	0.83	0.81	0.78	0.86
K-nearest neighbors	0.90	0.88	0.83	0.78	0.83
Random forest	0.91	**0.90**	**0.85**	0.80	**0.87**
Multinomial naïve Bayes	0.96	0.79	0.75	0.72	0.70
Logistic regression	**0.97**	0.74	0.72	0.70	0.72
Long short-term memory (LSTM)	**0.97**	0.85	0.84	**0.83**	0.82

3.2. Model Training and Evaluation for HateXplain Dataset

3.2.1. BERT + MLP

This section provides a discussion on the training of the dataset using the BERT model along with other techniques to provide explainability. BERT is a machine learning framework for NLP tasks specially designed to help computational systems for understanding the complex structure of language in the given text by using the surrounding text to establish some meaning. From the TensorFlow hub, a BERT model (TensorFlow Hub, 2021) and a preprocessor model were selected. There are various methods to deal with unbalanced data such as sampling techniques (upsampling and downsampling) where data are resampled, weighted loss where losses are weighted differently for data having class imbalance, and data augmentation which is used to artificially create variations in existing dataset. In this research, unbalanced data were dealt with using weight optimization, and bias was set. For weight optimization, appropriate weights were calculated for each class, depending upon their proportion. These weight factors were then multiplied to individual class so that the bias between classes could be removed.

Next, BERT was trained with the MLP model. Table 6 depicts the model summary for the BERT + MLP model, where the first column indicates the type of the layer, the second and third columns indicate the output shape and number of parameters generated by processing of the layer, respectively, and the last column represents the previous layer it is connected to. There were a total 29,027,843 trainable parameters.

There were a total 29,027,844 parameters. Among them, 29,027,843 were trainable parameters and only one was a nontrainable parameter.

Table 6. BERT + MLP model summary.

Layer (Type)	Output Shape	# of Parameters	Connected to
text (InputLayer)	[(None,)]	0	[]
preprocessing (KerasLayer)	{'input_word_ids': (None,128), 'input_mask': (None, 128), 'input_type_ids': (None, 128)}	0	['text [0][0]']
BERT_encoder (KerasLayer)	{'pooled_output': (28,763,649, None, 512), 'encoder_outputs': [(None, 128, 512), (None, 128, 512), (None, 128, 512), (None, 128, 512)], 'default': (None, 512), 'sequence_output': (None, 128, 512)}	28,763,649	['preprocessing [0][0]', 'preprocessing [0][1]', 'preprocessing [0][2]',]
dense (Dense)	(None, 512)	262,656	['BERT_encoder [0][5]']
dropout (Dropout)	(None, 512)	0	['dense [0][0]']
classifier (Dense)	(None, 3)	1539	['dropout [0][0]']
Total params: 29,027,844			
Trainable params: 29,027,843			
Non-trainable params: 1			

As shown in Figure 5, the architecture of the BERT + MLP model was fine-tuned in order to achieve the most efficient performance. The model contained one input and preprocessing layer, along with the BERT encoder, which was a keras layer. A dense layer was used after the keras layer to reduce the parameters and increase the number of features being propagated to the next layer. A dropout later was added to avoid overfitting of the model, followed by one dense layer used to represent the results as a classification problem. The model defined above was then compiled with the loss function as sparse categorical cross-entropy and the Adam optimizer.

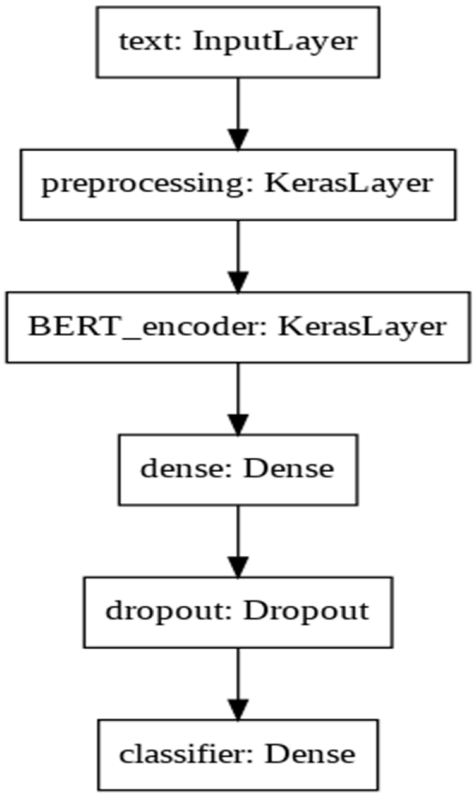

Figure 5. BERT + MLP model architecture.

3.2.2. BERT + ANN

Next, BERT was used with ANN to train the model and evaluate the performance. Table 7 depicts the model summary for the BERT + ANN model, where the first column indicates the type of the layer, the second and third columns indicate the output shape and the number of parameters generated by processing of the layer, respectively, and the last column represents the previous layer it is connected to.

Table 7. BERT + ANN model summary.

Layer (Type)	Output Shape	# of Parameters	Connected to
text (InputLayer)	[(None,)]	0	[]
preprocessing (KerasLayer)	{'input_word_ids':(None,128), 'input_mask': (None, 128), 'input_type_ids': (None, 128)}	0	['text [0][0]']
BERT_encoder (KerasLayer)	{'pooled_output': (28,763,649, None, 512), sequence_outputs': (None, 128, 512), 'encoder_outputs': [(None, 128, 512), (None, 128, 512), (None, 128, 512), (None, 128, 512)], 'default': (None, 512), 'sequence_output': (None, 128, 512)}	28,763,649	['preprocessing [0][0]', 'preprocessing [0][1]', 'preprocessing [0][2]',]
Conv1d (Conv1D)	(None, 127, 32)	32,800	['BERT_encoder [0][6]']
Conv1d_1 (Conv1D)	(None, 126, 64)	4160	['conv1d [0][0]']
Global_max_pooling1d (GlobalMaxpooling1D)	(None, 64)	0	['conv1d_1 [0][0]']
dense_1 (Dense)	(None, 512)	33,280	['BERT_encoder [0][5]']
dropout_1 (Dropout)	(None, 512)	0	['dense_1 [0][0]']
classifier (Dense)	(None, 3)	1539	['dropout_1 [0][0]']
Total params: 28,835,428			
Trainable params: 28,835,427			
Non-trainable params: 1			

As shown in Figure 6, the architecture of the BERT + ANN model was fine-tuned in order to achieve the most efficient performance. The model contained one input and preprocessing layer, along with the BERT encoder, which was a keras layer. BERT was combined with convolution layers, followed by a 1D global max-pooling layer, which computed the maximum of all the input sizes for each of the input channels. A dense layer was introduced after the 1D global max-pooling layer to reduce the parameters and increase the number of features being propagated to the next layer. In the end, a dropout later was added to avoid overfitting, followed by a dense layer. The model defined above was then compiled with the loss function as sparse categorical cross-entropy and the Adam optimizer.

The BERT + ANN and BERT + MLP models were trained for 50 epochs. As the number of epochs increased, the accuracy improved. The parameters used to find the number of training steps and number of warmup steps were as follows: number of epochs = 50, number of training steps = steps per epoch $\times$ number of epochs, and number of warmup steps = 0.1 $\times$ number of training steps.

The accuracy obtained by the BERT + MLP model and BERT + ANN model was 93.67% and 93.55%, respectively, indicating that the gap in conventional evaluation metrics was minimal; however, in terms of the explainability metrics, BERT + ANN performed slightly better than the BERT + MLP model, as discussed later in this section.

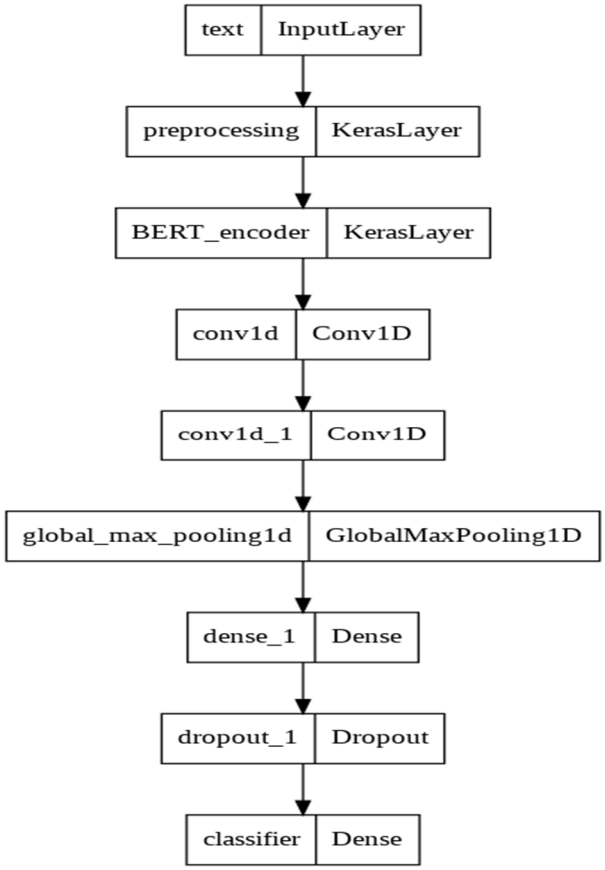

Figure 6. BERT + ANN model architecture.

3.2.3. LIME with Machine Learning Models

This section discusses the implementation of the LIME model with other linear machine learning models in order to provide explainability and interpretability.

The same labeled dataset used for BERT with ANN and MLP was used for training the LIME model. LIME was trained using linear noncomplex machine learning models such as random forest, naïve Bayes, decision tree, and logistic regression to extract the explanations. Table 5 summarizes the accuracy achieved by each of the models on the HateXplain dataset. It can be seen that logistic regression performed the best with an accuracy of 88.57%.

In this section, LIME classification is demonstrated using an example. The comment text was as follows: "@ComedyPosts: Harlem shake is just an excus to go full retard for 30 s". After the preprocessing was performed on the text, the comment text was reduced to "comedypost harlem shake excus go full retard second". This comment text was obtained from the corpus of the preprocessed pandas data frame and applied to the LIME text explainer for each of the machine learning models. The same comment text was used for all the models so as to compare each model.

Explainability with Random Forest

Figure 7 shows the explainability with LIME and random forest for a particular tweet. It can be observed that the LIME explainer gave weights to each useful word in the comment to indicate its importance in the overall decision making. From Figure 7, we can see that words such as "excus" and "retard" had the highest weights in contributing to the overall prediction probability at 0.10 and 0.07 respectively. The prediction probability of the tweet to be considered as hate speech was reduced by the word "full". Text that contributed in either direction is highlighted on the right side of the figure. The overall prediction probability for hate speech was 90% using the random forest classifier.

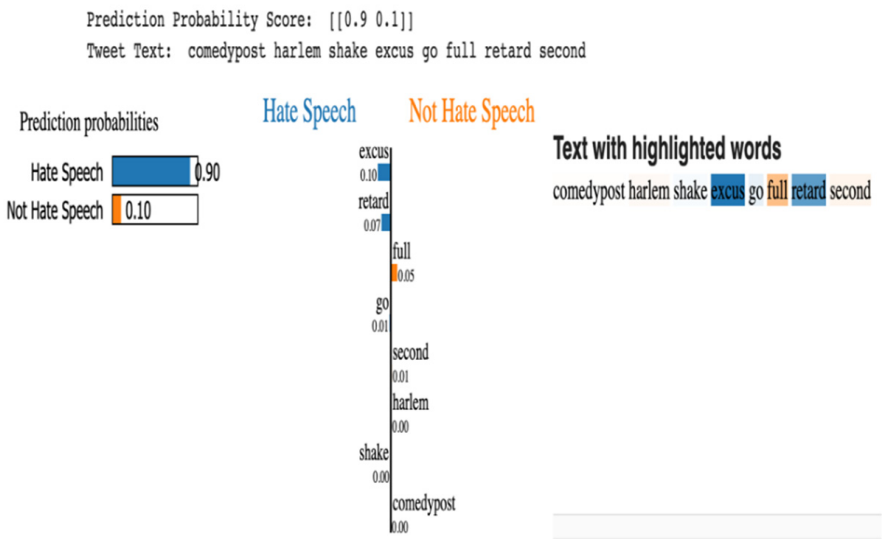

Figure 7. Explainability with random forest.

Explainability with Gaussian Naïve Bayes

Figure 8 shows the explainability with LIME and Gaussian naïve Bayes for the example tweet. It can be observed that the LIME explainer gave weights to each useful word in the comment to indicate its importance in the overall decision making. From Figure 8, we can see that words such as "full" and "excus" had the highest weights in contributing to the overall prediction probability at 0.08 and 0.07, respectively. Interestingly, the prediction probability of the tweet to be considered hate speech was reduced by the word "retard" in the case of the gaussian naïve Bayes classifier. The word retard had the prediction probability of 0.14, which eventually increased the overall prediction probability of the text not being hate speech by 20%. Text that contributed in either direction to the prediction is highlighted on the right side of the figure. The overall prediction probability for hate speech was 80% using the Gaussian naïve Bayes classifier.

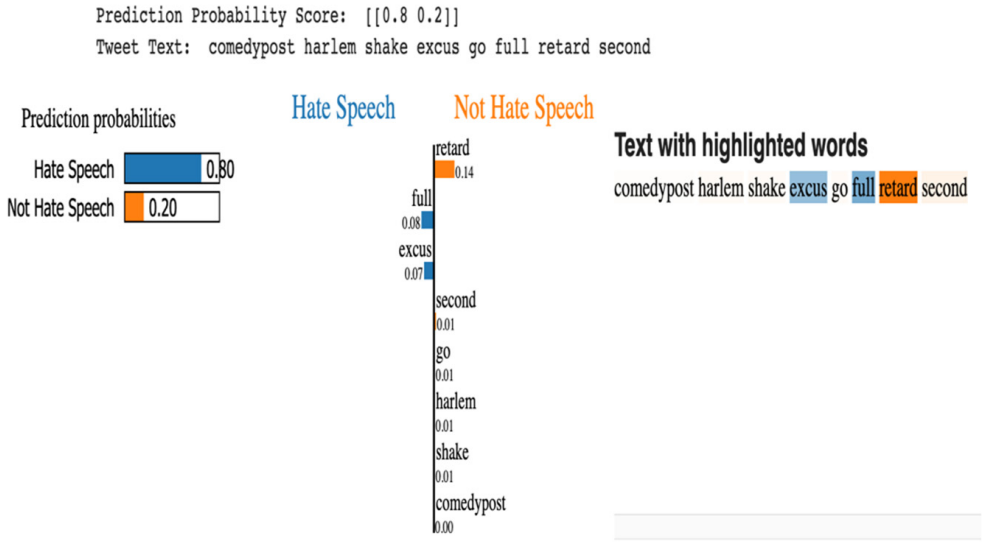

Figure 8. Explainability with Gaussian naïve Bayes.

Explainability with Decision Tree

Figure 9 shows the explainability with LIME and decision tree for the example tweet. It can be observed that the LIME explainer gave weights to each useful word in the comment to indicate its importance in the overall decision making. From Figure 9, we can see that

words such as "full", "excus", and "retard" had the highest weights in contributing to the overall prediction probability at 0.07, 0.06, and 0.06, respectively. The decision tree classifier did not give weight to predict the comment as non-hate speech for any of the words. We can see the trend in the text with highlighted words. The overall prediction probability for hate speech was 100% using the decision tree classifier.

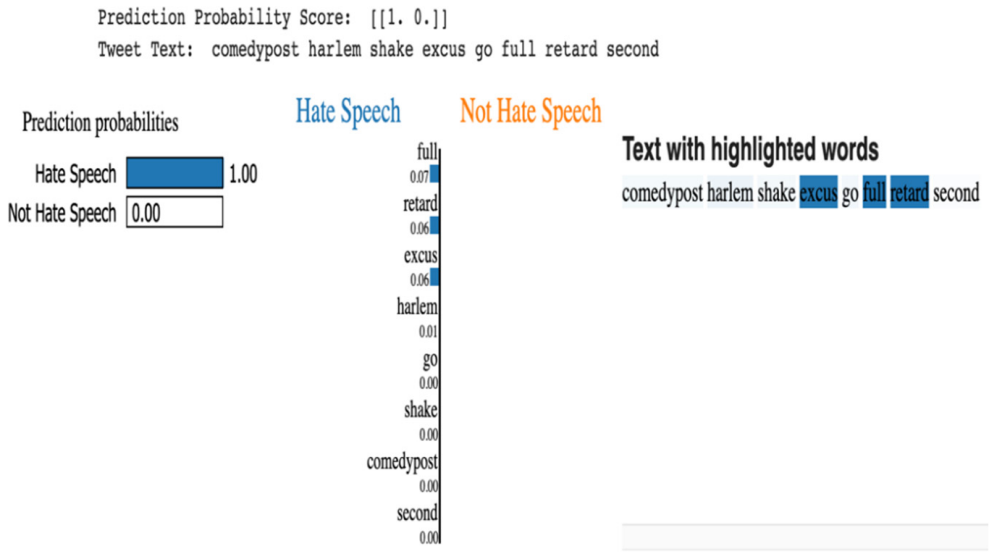

Figure 9. Explainability with decision tree.

Explainability with Logistic Regression

Figure 10 shows the explainability with LIME and logistic regression for the example tweet. It can be observed that the LIME explainer gave weights to each useful word in the comment to indicate its importance in the overall decision making. From Figure 10, we can see that words such as "excus" and "second" had the highest weights in contributing to the overall prediction probability at 0.03 and 0.04, respectively. On the other hand, words such as "retard" and "full" contributed to the text not being hate speech with the weights of 0.04 and 0.03, respectively. Text that contributed in either direction of prediction is highlighted on the right side of the figure. The overall prediction probability for hate speech was 95% using the logistic regression classifier.

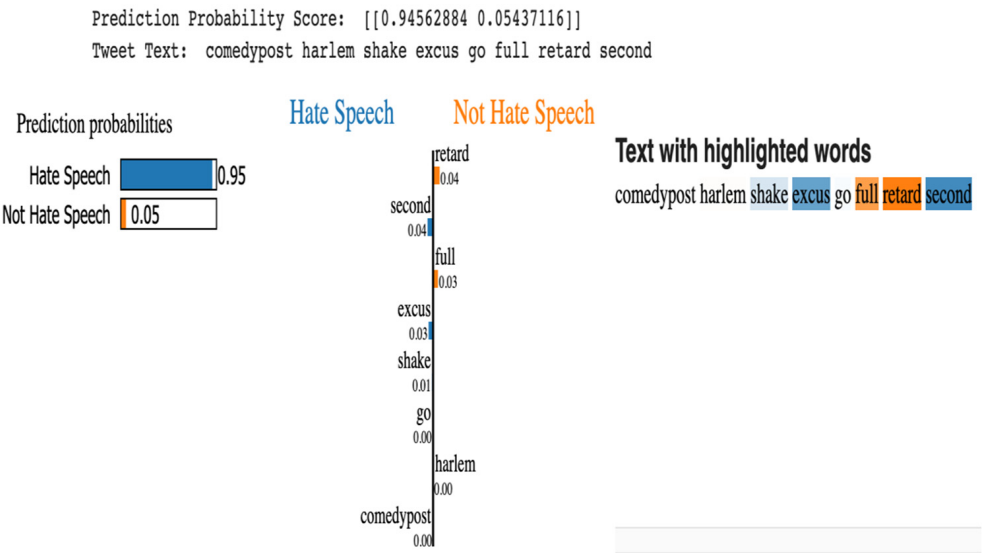

Figure 10. Explainability with logistic regression.

3.2.4. Summary of Results for the HateXplain Dataset

The results of all the models on the HateXplain dataset, evaluated in terms of their accuracy, precision, and macro F1-score are visualized in Figure 11. Table 8 gives the evaluation scores for all the models. It can be observed that BERT variants performed significantly better than the other linear explainable models, with BERT + MLP having the highest accuracy of 93.67%, closely followed by BERT + ANN with an accuracy of 93.55%. It can be observed that the measures such as precision, recall, and macro F1-score also indicated that the BERT variants outperformed the other linear models. Logistic regression with LIME performed best among the linear models with an accuracy of 88.57% and macro F1-score of 93.75%. The results are visualized in Figure 11 as a bar chart.

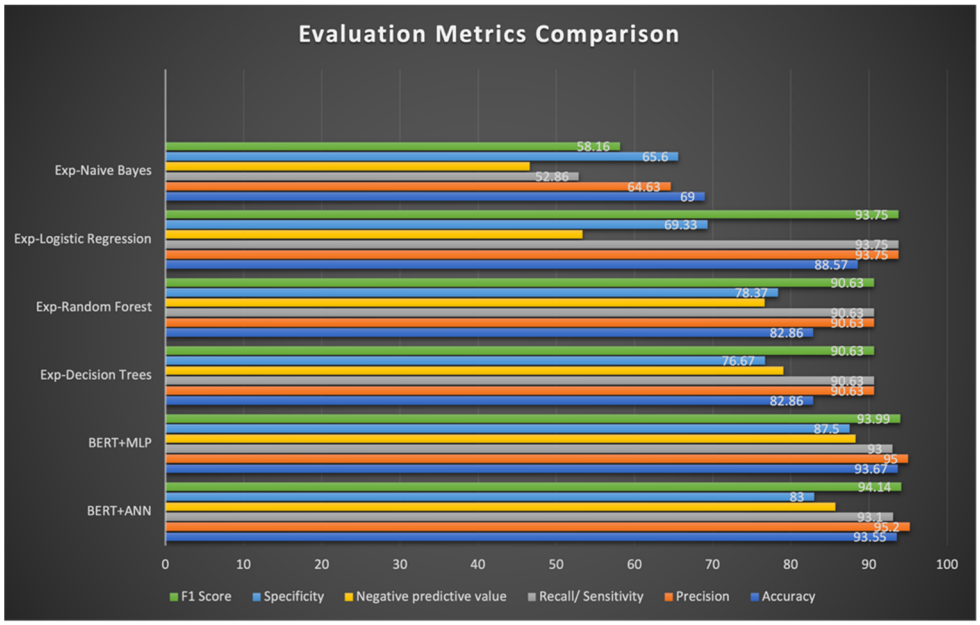

Figure 11. Result summary of all models on the HateXplain dataset.

Table 8. Results of models on the HateXplain dataset.

S. No	Full Form	Features	Accuracy	Precision	Recall/ Sensitivity	Negative Predictive Value	Specificity	F1-Score
1	Bidirectional encoder representations from transformers + artificial neural network layers (BERT + ANN)	Explainable, layer-wise propagation	**93.55**	**95.2**	**93.1**	**85.7**	**83**	**94.14**
2	Bidirectional encoder representations from transformers + multilayer perceptron (BERT + MLP)	Explainable, layer-wise propagation	**93.67**	**95**	**93**	**88.3**	**87.5**	**93.99**
3	Exp-decision trees	Explainable (LIME)	82.86	90.63	90.63	79.03	76.67	90.63
4	Exp-random forest	Explainable (LIME)	82.86	90.63	90.63	76.65	78.37	90.63
5	Exp-logistic regression	Explainable (LIME)	88.57	93.75	93.75	53.33	69.33	93.75
6	Exp-naïve Bayes	Explainable (LIME)	69	64.63	52.86	46.6	65.6	58.16

Explainability Metrics

We used the ERASER benchmark [35] in order to measure the explainability of the trained models. ERASER (evaluating rationales and simple English reasoning) is a benchmark to evaluate rationalized NLP models, which was proposed by DeYoung et al. (2020). This is achieved by measuring the agreement with human rationales. Measuring exact matches between predicted and reference rationales is likely too harsh; thus, explainability

is assessed by measuring plausibility and faithfulness. The prediction is counted as a match if any of the word predictions overlap with the rationales annotated by humans. Token level calculations are compared with human annotations to derive the explainability. Various measures were used from the ERASER benchmark to calculate these comparisons.

Plausibility is the measure of how cogent the interpretation is to a human. To measure plausibility, the metrics IOU (intersection over union) F1-score, token F1-score, and area under the precision–recall curve (AUPRC) score were calculated. The IOU (intersection over union) F1-score was calculated for token level. Partial matches were considered where prediction overlapped more than 0.5 with either of the ground truth rationales. Token-level F1-scores were measured from the token-level precision and recall. AUPRC was used to measure soft token scoring. Higher values of all these metrics indicated greater plausibility.

Faithfulness is the measure of the accuracy of the true reasoning process of the model. To measure the faithfulness of the models, comprehensiveness and sufficiency were calculated. The comprehensiveness score is a measure of change in the probability of the output of the originally predicted class after eliminating significant tokens. A higher comprehensiveness score indicates a more faithful interpretation. Sufficiency measures the sufficiency of the important tokens to sustain the predictions. It captures the degree to which the snippets within the exact rationales are adequate for a model to make a prediction. A lower sufficiency indicates a more faithful model.

Table 9 provides a summarized view of the explainability metrics calculated on all the models implemented. It can be observed that BERT + MLP was the best-performing model in terms of plausibility. The BERT + MLP model showed the best values of IOU F1, token F1, and AUPRC as compared to the other models. In terms of faithfulness, the BERT + ANN model showed the best results with the highest comprehensiveness score of 0.4199. The achieved results are an improvement compared to the base paper by Mathew et al. (2020). BERT variants had the most convincing interpretation to the humans. BERT + ANN achieved a slightly higher comprehensiveness than BERT + MLP, due to the simpler structure of ANN than MLP. The same trend in the parameters of sufficiency was observed in the base paper by Mathew et al. (2020).

Table 9. Explainability metrics.

Technique	Plausibility			Faithfulness	
	IOU F1	**Token F1**	**AUPRC**	**Comprehensiveness**	**Sufficiency**
BERT + ANN	**0.1888**	**0.5074**	**0.8384**	**0.4199**	**0.0055**
BERT + MLP	**0.298**	**0.5298**	**0.8589**	**0.3574**	**0.003**
DT-LIME	0.1676	0.3887	0.7487	0.2993	0.0442
RF-LIME	0.2387	0.5118	0.8469	0.4132	0.0014
LR-LIME	0.1008	0.2271	0.5284	0.2132	0.0482
NB-LIME	0.1287	0.1818	0.5938	0.1999	0.0514

Bias-Based Metrics

The hate speech detection models could make biased predictions for particular groups who are already the target of such abuse (Sap et al. 2019; Davidson, Bhattacharya, and Weber 2019). To measure these unintended model biases, the AUC-based metrics by Borkan et al. (2019) were used. We computed the subgroup AUC (area under the ROC curve), BPSN (background positive, subgroup negative) AUC, and BSNP (background negative, subgroup positive) AUC. Subgroup AUC metrics for this use case are a measure of the ability of the model to segregate the toxic and normal comments. A higher value of subgroup AUC suggests that the model is better at differentiating between toxic and normal posts. The BPSN (background positive, subgroup negative) AUC metric is a measure of false-positive rates of the model, while the BNSP (background negative, subgroup positive) AUC is a measure of false-negative rates of the model. A higher value of BPSN indicates a lower likelihood of the model giving false positives, while a higher value of BSNP indicates

a lower likelihood of the model giving false negatives. For this dataset, these metrics were calculated with respect to a community.

Table 10 provides a summarized view of the bias-based metrics calculated on all the models implemented. We can see that the bias-based metrics of BERT variants were significantly more accurate than the other linear models. BERT + MLP had the highest values of subgroup AUC, BPSN AUC, and BSNP AUC with 0.8229, 0.7752, and 0.8077, respectively, followed by BERT + ANN with values of 0.7977, 0.7188, and 0.7391, respectively.

Table 10. Explainability Metrics.

Technique	Subgroup AUC	BPSN AUC	BSNP AUC
BERT + ANN	**0.7977**	**0.7188**	**0.7391**
BERT + MLP	**0.8229**	**0.7752**	**0.8077**
DT-LIME	0.6926	0.6578	0.6617
RF-LIME	0.7627	0.6977	0.5978
LR-LIME	0.5266	0.4522	0.4991
NB-LIME	0.6136	0.4812	0.5049

4. Conclusions

In this research study, two datasets were taken to demonstrate hate speech detection using explainable artificial intelligence (XAI). Exploratory data analysis was performed on the datasets to uncover various patterns and insights, and various explainable models were trained on both datasets to extract useful interpretable results. The conclusions of the study are discussed in this section.

4.1. Conclusions of the Study on the Google Jigsaw Dataset

The Google Jigsaw dataset comprises user discussions from talk pages of English Wikipedia, and it was released by Google Jigsaw. We trained various existing interpretable models (decision tree, KNN, random forest, multinomial naïve Bayes, logistic regression, and LSTM) on this dataset. We found that LSTM outperformed the other models in terms of accuracy (97.6%) and recall (83%) scores. The random forest model had the best performance in terms of precision (90%) and specificity (87%). KNN, logistic regression, and multinomial naïve Bayes had low evaluation scores as compared to the other models, but they performed very well in terms of accuracy with 90%, 97%, and 96%, respectively. Decision trees and random forest also had significantly good performance with an accuracy of 89% and 91%, respectively. It was observed that the LSTM model gave better overall performance in terms of accuracy, precision, recall, and macro F1-score as compared to the study of Risch et al. (2020).

4.2. Conclusion of the Study on the HateXplain Dataset

The HateXplain dataset comprises posts from Twitter and Gab and is annotated by human annotators. Several state-of-the-art models were tested on this dataset to perform evaluation on several aspects of the hate speech detection. These models contained explainability imbibed in various ways. LIME was used with interpretable models such as decision trees, random forest, logistic regression, and naïve Bayes to extract weights of words that contributed significantly to the model's decision making. Furthermore, variants of BERT were created to achieve the best performance. The best performance was observed for the BERT variants, BERT + ANN and BERT + MLP, as compared to the other models. BERT + ANN had a slightly better overall performance than BERT + MLP. For appropriate comparisons, the evaluation metrics were divided into three subsets, namely, performance metrics (accuracy, precision, recall, negative predicted value, specificity, and macro F1-score), bias-based metrics, and explainability metrics (plausibility and faithfulness) as in mathew et al. (2020). LIME was used to demonstrate the textual explanations on some data of the black-box models. We used explanation metrics based on the ERASER benchmark by DeYoung et al. for the human-annotated dataset HateXplain. These metrics

suggested how faithful the results of these models were in identifying hateful comments as compared to other existing models. LIME is a surrogate model which is used to highlight contributing words or tokens that can play major part in a comment being hateful or not hateful. The accuracy scores of BERT + MLP and BERT + ANN were 93.67% and 93.55%, respectively, outperforming the simple BERT implementations by Mathew et al. (2020) with an accuracy score 69.8%. The prime reason behind this difference was the combination of BERT with neural network models such as MLP and ANN. Furthermore, our models are trained on 50 epochs, which took around 11.5 and 8.3 h, on Google Colab Pro. The precision scores of BERT + ANN and BERT + MLP were 95.2% and 95%, while recall scores were 93.1% and 93%, respectively. The results of the macro F1-score were calculated to be 94.14% and 93.99%, respectively.

In terms of bias-based metrics, the BERT variant models performed better in reducing the unintended model bias for all the bias metrics. We observed that the presence of community terms within the rationales was effective in reducing the unintended bias. The BERT + MLP model handled this bias much better than other models in terms of subgroup, BPSN (background positive, subgroup negative), and BNSP (background negative, subgroup positive) AUC with values of 0.8229, 0.7752, and 0.8077, respectively, representing an improvement over simple BERT implementation (0.807, 0.745, and 0.763, respectively) by Mathew et al. (2020). Future research on hate speech should consider the impact of the model performance on individual communities to have a clear understanding.

Considering the explainability metrics using the ERASER benchmark by DeYoung et al. (2019), two main factors were evaluated: plausibility (defined by IOU F1, token F1, and AUPRC) and faithfulness (defined by comprehensiveness and sufficiency). The best-performing models, BERT + ANN and BERT + MLP, had plausibility (IOU F1, token F1, and AUPRC) values of 0.188, 0.507, and 0.8384, and 0.29, 0.529, and 0.8589, respectively, compared to the base BERT model (0.222, 0.506, and 0.841, respectively) in the paper by Mathew et al. (2020). BERT + MLP performed better than the simple BERT implementation. Similarly, the faithfulness (comprehensiveness and sufficiency) values were found to be 0.419 and 0.0055 for BERT + ANN and 0.3574 and 0.003 for BERT+ MLP. BERT + ANN performed better compared to the BERT implementation in the paper by Mathew et al. (2020) (0.436 and 0.008, respectively).

Hence, it can be derived that the variants of BERT used in the research work had superior performance to the base model; BERT + ANN performed best in terms of explainability, and BERT + MLP performed best overall compared to traditional models such as logistic regression, KNN, naïve Bayes, decision trees, and random forests.

Author Contributions: Conceptualization, H.M. and K.P.; methodology, H.M. and K.P.; software, H.M.; validation, H.M. and K.P.; formal analysis, H.M.; investigation, H.M.; resources, H.M.; data curation, H.M.; writing—original draft preparation, H.M.; writing—review and editing, K.P.; visualization, H.M.; supervision, K.P.; project administration, K.P.; funding acquisition, K.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets are publicly available as follows: Google Jigsaw dataset: https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data (accessed 5 January 2022); HateXplain dataset: https://github.com/hate-alert/HateXplain/tree/master/Data (accessed 7 January 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Davidson, T.; Warmsley, D.; Macy, M.; Weber, I. Automated Hate Speech Detection and the Problem of Offensive Language. Available online: http://arxiv.org/abs/1703.04009 (accessed on 11 August 2022).
2. Chen, Y.; Zhou, Y.; Zhu, S.; Xu, H. Detecting offensive language in social media to protect adolescent online safety. In Proceedings of the 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust and 2012 ASE/IEEE International Conference on Social Computing, SocialCom/PASSAT, Amsterdam, The Netherlands, 3–5 September 2012; pp. 71–80. [CrossRef]

3. Balkir, E.; Nejadgholi, I.; Fraser, K.C.; Kiritchenko, S. Necessity and sufficiency for explaining text classifiers: A case study in hate speech detection. *arXiv* **2022**, arXiv:2205.03302.
4. Chatzakou, D.; Kourtellis, N.; Blackburn, J.; de Cristofaro, E.; Stringhini, G.; Vakali, A. Mean birds: Detecting aggression and bullying on Twitter. In *WebSci 2017—Proceedings of the 2017 ACM Web Science Conference*; Association for Computing Machinery: New York, NY, USA, 2017; pp. 13–22. [CrossRef]
5. Founta, A.M.; Chatzakou, D.; Kourtellis, N.; Blackburn, J.; Vakali, A.; Leontiadis, I. A Unified Deep Learning Architecture for Abuse Detection. In *WebSci 2019—Proceedings of the 11th ACM Conference on Web Science*; Association for Computing Machinery: New York, NY, USA, 2018; pp. 105–114. [CrossRef]
6. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
7. Arras, L.; Montavon, G.; Müller, K.R.; Samek, W. Explaining recurrent neural network predictions in sentiment analysis. In *EMNLP 2017—8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA 2017—Proceedings of the Workshop*; Association for Computational Linguistics: Copenhagen, Denmark, 2017. [CrossRef]
8. Mahajan, A.; Shah, D.; Jafar, G. Explainable AI approach towards toxic comment classification. In *Emerging Technologies in Data Mining and Information Security*; Springer: Singapore, 2021; pp. 849–858.
9. Ribeiro, M.T.; Singh, S.; Guestrin, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In Proceedings of the NAACL-HLT 2016—2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Demonstrations Session, San Francisco, CA, USA, 13–17 August 2016; pp. 1135–1144. [CrossRef]
10. Samek, W.; Montavon, G.; Lapuschkin, S.; Anders, C.J.; Müller, K.-R. Toward Interpretable Machine Learning: Transparent Deep Neural Networks and Beyond. Available online: https://doi.org/10.48550/arXiv.2003.07631 (accessed on 11 August 2022). [CrossRef]
11. Doshi-Velez, F.; Kim, B. Towards A Rigorous Science of Interpretable Machine Learning. 2017. Available online: https://doi.org/10.48550/arXiv.1702.08608 (accessed on 11 January 2022). [CrossRef]
12. Hind, M.; Wei, D.; Campbell, M.; Codella, N.C.F.; Dhurandhar, A.; Mojsilović, A.; Natesan Ramamurthy, K.; Varshney, K.R. TED: Teaching AI to explain its decisions. AIES 2019. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 123–129. [CrossRef]
13. Montavon, G.; Samek, W.; Müller, K.-R. Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.* **2018**, *73*, 1–15. [CrossRef]
14. Gilpin, L.H.; Bau, D.; Yuan, B.Z.; Bajwa, A.; Specter, M.; Kagal, L. Explaining explanations: An overview of interpretability of machine learning. In Proceedings of the 2018 IEEE 5th International Conference on Data Science and Advanced Analytics, DSAA, Turin, Italy, 1–3 October 2018. [CrossRef]
15. Lundberg, S.M.; Erion, G.; Chen, H.; DeGrave, A.; Prutkin, J.M.; Nair, B.; Katz, R.; Himmelfarb, J.; Bansal, N.; Lee, S. Explainable AI for Trees: From Local Explanations to Global Understanding. *arXiv* **2019**, arXiv:1905.04610. Available online: http://arxiv.org/abs/1905.04610 (accessed on 11 May 2022). [CrossRef] [PubMed]
16. Nori, H.; Jenkins, S.; Koch, P.; Caruana, R. InterpretML: A Unified Framework for Machine Learning Interpretability. *arXiv* **2019**, arXiv:1909.09223. [CrossRef]
17. Ahmed, U.; Lin, J.C.-W. Deep Explainable Hate Speech Active Learning on Social-Media Data. *IEEE Trans. Comput. Soc. Syst.* **2022**, 1–11. [CrossRef]
18. Barredo Arrieta, A.; Díaz-Rodríguez, N.; del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; Garcia, S.; Gil-Lopez, S.; Molina, D.; Benjamins, R.; et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion* **2020**, *58*, 82–115. [CrossRef]
19. Das, A.; Rad, P. Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey. *arXiv* **2020**, arXiv:2006.11371. [CrossRef]
20. Kanerva, O. Evaluating Explainable AI Models for Convolutional Neural Networks with Proxy Tasks. Available online: https://www.semanticscholar.org/paper/Evaluating-explainable-AI-models-for-convolutional_Kanerva/d91062a3e13ee034af6807e1819a9ca3051daf13 (accessed on 25 January 2022).
21. Gohel, P.; Singh, P.; Mohanty, M. Explainable AI: Current STATUs and Future Directions. Available online: https://doi.org/10.1109/ACCESS.2017 (accessed on 30 January 2022). [CrossRef]
22. Fernandez, A.; Herrera, F.; Cordon, O.; Jose Del Jesus, M.; Marcelloni, F. Evolutionary fuzzy systems for explainable artificial intelligence: Why, when, what for, and where to? *IEEE Comput. Intell. Mag.* **2019**, *14*, 69–81. [CrossRef]
23. Clinciu, M.-A.; Hastie, H. A Survey of Explainable AI Terminology. In *Proceedings of the 1st Workshop on Interactive Natural Language Technology for Explainable Artificial Intelligence (NL4XAI 2019)*; Association for Computational Linguistics: Copenhagen, Denmark, 2019; pp. 8–13. [CrossRef]
24. Hrnjica, B.; Softic, S. Explainable AI in Manufacturing: A Predictive Maintenance Case Study. In *IFIP Advances in Information and Communication Technology, 592 IFIP*; Springer: New York, NY, USA, 2020; pp. 66–73. [CrossRef]
25. Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.* **2019**, *267*, 1–38. [CrossRef]
26. Mathew, B.; Saha, P.; Yimam, S.M.; Biemann, C.; Goyal, P.; Mukherjee, A. HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection. *arXiv* **2020**, arXiv:2012.10289. Available online: http://arxiv.org/abs/2012.10289 (accessed on 14 June 2021).

27. ML | Overview of Data Cleaning. GeeksforGeeks. 15 May 2018. Available online: https://www.geeksforgeeks.org/data-cleansing-introduction/ (accessed on 3 April 2022).
28. Pearson, R.K. Exploratory Data Analysis: A First Look. In *Exploratory Data Analysis Using R*; Chapman and Hall/CRC: New York, NY, USA, 2018.
29. Using CountVectorizer to Extracting Features from Text. GeeksforGeeks. 15 July 2020. Available online: https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/ (accessed on 3 April 2022).
30. Bisong, E. The Multilayer Perceptron (MLP). In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*; Apress: Berkeley, CA, USA, 2019; pp. 401–405. [CrossRef]
31. Kamath, U.; Graham, K.L.; Emara, W. Bidirectional encoder representations from transformers (BERT). In *Transformers for Machine Learning*; Chapman and Hall/CRC: New York, NY, USA, 2022; pp. 43–70. [CrossRef]
32. Awal, M.R.; Cao, R.; Lee, R.K.-W.; Mitrovic, S. AngryBERT: Joint Learning Target and Emotion for Hate Speech Detection. *arXiv* **2021**, arXiv:2103.11800. Available online: http://arxiv.org/abs/2103.11800 (accessed on 16 July 2022).
33. Nair, R.; Prasad, V.N.V.; Sreenadh, A.; Nair, J.J. Coreference Resolution for Ambiguous Pronoun with BERT and MLP. In Proceedings of the 2021 International Conference on Advances in Computing and Communications (ICACC), Kochi, India, 21–23 October 2021; pp. 1–5. [CrossRef]
34. Biecek, P.; Burzykowski, T. Local interpretable model-agnostic explanations (LIME). In *Explanatory Model Analysis*; Chapman and Hall/CRC: New York, NY, USA, 2021; pp. 107–123. [CrossRef]
35. DeYoung, J.; Jain, S.; Rajani, N.F.; Lehman, E.; Xiong, C.; Socher, R.; Wallace, B.C. ERASER: A Benchmark to Evaluate Rationalized NLP Models. *arXiv* **2020**, arXiv:1911.03429. [CrossRef]