**POLARITY SCORE-BASED MACHINE LEARNING CLASSIFIER FOR DISTINGUISHING FAKE AND AUTHENTIC TWEETS USING NATURAL LANGUAGE PROCESSING**

**ABSTRACT**

In every city, harassment and violence becomes one of the major problems for women. Further, women’s personal life is suffered by the bullying and abusive content presented in online social networking (OSN). Therefore, it is necessary to identify the women safety in OSN environment. However, the conventional methods failed to predict the maximum safety analysis. So, this work is focused on women safety prediction using decision tree (WSP-DT) classifier. Initially, twitter dataset is considered to implement the entire system, which is then pre-processed to remove the missing and unknown symbols. Then, natural language toolbox kit (NLTK) applied to perform the tokenization, conversion to lowercase, stop words identification, stemming and lemmatization of tweets. Then, text blob protocol is developed to identify sentiments of pre-processed tweets, which identifies the positive, negative and neutral polarities of tweets. Further, term frequency-inverse document frequency (TF-IDF) is applied to extract the data features based on word and character frequency. Finally, decision tree classifier applied to identify the fake or genuine tweet based on multi-level training. The simulations conducted on twitter dataset show that the proposed WSP-DT classifier resulted in superior performance than the other methods.

**CHAPTER 1**

**INTRODUCTION**

Communication is the process of exchanging information. As time goes by, many ways and platforms of communication are being developed. Since the industrial revolution, the original way of communicating; face-to-face communication has been used as a model to develop the various ways of communicating known to date. Transposing the principles and codes of natural face-to-face communication to today’s online communication is a major challenge for developers. Sarcasm is the communication practice that consists of meaning the opposite of what is said in order to mock or insult someone [1]. Sarcasm makes use of positive lingual contents in order to convey a negative message. Different types of approaches have been developed in order to implement sarcasm detection on online communication platforms. However, the levels of efficiency of these approaches have been the principal worries of developers. In this paper, propositions are made on how the lexicon algorithm can be extended in order to come up with systems that would be proven more efficient for sarcasm detection on textual contents. Sarcasm is a form of communication where the intended meaning of a statement is the opposite of its literal interpretation. Detecting sarcasm is a challenging task in natural language processing (NLP) due to the inherent complexity of sarcasm and the subtleties involved in its expression. With the widespread use of social media platforms like Twitter, there is a growing interest in developing methods to automatically detect sarcasm from tweets. Sarcasm detection has numerous applications, including sentiment analysis, opinion mining, and social media analytics. Sarcasm detection has gained significant attention in recent years due to its potential applications in various domains. Twitter, with its limited character count and fast-paced nature, has become a popular platform for users to express their opinions, emotions, and sarcasm. However, sarcasm in tweets often relies heavily on contextual cues, such as irony, ambiguity, and exaggerated statements, making it challenging to detect using traditional NLP techniques.

**CHAPTER 2**

**LITERATURE SURVEY**

Disseminated information and its dissemination process build a major problem in detecting these contents immediately, thus highlighting the importance of automatically identifying false news. To overcome this, Sahoo et al [[23](https://link.springer.com/article/10.1007/s11042-023-14883-3#ref-CR23)] proposed an automatic fake news identification technique for the environment chrome using this the detection of fake news on Facebook is possible. Specifically, this uses multiple features associated with a Facebook account in addition to some news content features to analyse the characteristics of the account across deep learning. Shu et al [[24](https://link.springer.com/article/10.1007/s11042-023-14883-3#ref-CR24)] present FakeNewsNet, a repository of fake news data, the news content includes two complete datasets with different features, spatiotemporal information, and social context to make facilitate fake news-related research. This comprehensive description of FakeNewsNet displays an analytical analysis of two datasets from various viewpoints and discussed the advantages of FakeNewsNet for potential applications in social media fake news research. SAF/S performs better in terms of accuracy and F1 score. SAF/A provides a similar result with 66.7% accuracy as SAF/S. This indicates that user engagements can help fake news detection in addition to news articles on the PolitiFact data set. Meanwhile, the selection strategy can be used for web search results to reduce noise in the data collection process.

The studies on user credibility in this context focus more on the frequency and timing of engaging in fake news propagation, rather than specification according to the content of users’ tweets. Duan et al [[8](https://link.springer.com/article/10.1007/s11042-023-14883-3#ref-CR8)] approach this challenge by elaborating two features one is linguistic and another one is a sentiment feature from operators’ tweet feed as well as retrieving the presence of hashtags, emojis, and political bias in their tweets. These features were later used to categorize operators as those who broadcast or did not broadcast fake news. 72% accuracy was obtained by this proposed approach, among the results in the first 4 positions acquired by systems for the task in the English language. Yet, in applications with diverse classification algorithms and the union of the different representations, not all combinations of representations increased the accuracy. NER in combination with other representations is not suitable for the use of SVMs or ANNs. Moreover, this limit had to be raised multiple times. The need for this is probably due to a large number of features (416,834).

A domain reputation analysis was proposed by Xu et al [[26](https://link.springer.com/article/10.1007/s11042-023-14883-3#ref-CR26)] that reveals the internet pages of real and fake news publishers revealing different registration behaviours, registration time, domain rankings and domain popularity. In addition, fake messages will disappear from the Internet after a certain time. This content on the false and original news corpus is unskilful in detecting false news, using time frequency-inverse document frequency (tf-IDF) and Latent Dichotomy Allocation (LDA) header modelling, while exploring document compatibility with word and word. Vectors are the most promising direction to predict original and false news. This shows the promising aspect of leveraging document similarity to distinguish fake and real news by measuring the document similarity of the news under tests with the known fake and real news corpus. On the other hand, the difference in the topics and word embeddings shows little or subtle difference between fake and real news.

Kumar et al [[18](https://link.springer.com/article/10.1007/s11042-023-14883-3#ref-CR18)] proposed a CNN + bidirectional LSTM ensembled network to gather fresh instances such as PolitiFact and build multiple information for the identification of original and false news and match multiple state-of-the-art approaches. Long Short-Term Memories (LSTMs), Convolutional neural networks (CNNs), attention mechanisms and ensemble methods are examples of multiple state-of-the-art approaches. This research collects 1356 news instances from various users via Twitter and media sources such as PolitiFact and creates several datasets for the real and the fake news stories. The study conclude that CNN + bidirectional LSTM ensembled network with attention mechanism achieved the highest accuracy of 88.78%, whereas Ko et al. tackled the fake news identification problem and achieved a detection rate of 85%. As the result, CNN + bidirectional LSTM ensembled network with focus mechanism obtained 88.78% of maximum accuracy. The results were satisfactory but not promising. The CNN architecture gave the lowest accuracy in comparison to the others that we studied. The LSTM architecture and bidirectional LSTM architecture performed significantly better in comparison to simple CNN architecture. We further increased our appetite for improved accuracy and incorporated more complex models as part of our methodology.

The hybrid deep learning design that merges recurrent neural and convolutional networks for false news identification was proposed by Nasir et al [[21](https://link.springer.com/article/10.1007/s11042-023-14883-3#ref-CR21)]. On two fake news datasets (ISO and FA-KES) this model was certified successfully, achieving the results of detection that are substantially better than other non-hybrid foundation techniques. A paired t-test was used to validate the statistical significance of the results; the experiments were repeated five times (using 5-fold cross-validation, i.e. 80%–20% split); and accuracy was reported at 95% confidence intervals. ISOT is chosen for training because it is much larger and has minimum space for improvement since many models perform above the 0.9 classification accuracy threshold. Moreover, complex neural network architectures not be considered as part of the study.

A deep convolutional neural network (FNDNet) for false news detection was proposed by Kaliyar et al [[16](https://link.springer.com/article/10.1007/s11042-023-14883-3#ref-CR16)]. This prototype (FNDNet) is outlined instead of relying on hand-crafted features to learn automatically, about the one-sided features for false news identification build in the deep neural network across many hidden layers. As the result, each layer contains many features that will be extracted by a deep Convolutional Neural Network (CNN). Benchmarked datasets were used to train and test the model, and the proposed model achieved state-of-the-art results with an accuracy of 98.36% on the test data. Various performance evaluation parameters such as Wilcoxon, false positive, true negative, precision, recall, F1, accuracy, etc. were used to validate the results. Despite the high performance of our classifier, there is a scope for improvement. A multi-model approach (a combination of different learning techniques) is the main necessity for fake news detection for solving the multi-class fake news detection problem.

Choudhari et al [[6](https://link.springer.com/article/10.1007/s11042-023-14883-3#ref-CR6)] proposed a linguistic model to identify the properties of the content and language-driven features will also generate with the help of this. This linguistic prototype extracts particular news features such as syntactic, sentimental, grammatical, and readability. The language-driven model demands an approach to managing handcrafted feature problems and is time-consuming to maintain the trouble of dimensionality problems. Therefore, a continuous learning model based on neutrality is utilized to achieve the best results for detecting fake news. The results are drawn up to verify the importance of the extracted features of the linguistic model and finally, the integrated linguistic feature-driven model that can achieve an average of 86% accuracy in detecting and categorizing fake messages. However, extensive features/parameters for model performance are lacking. Examine the latent semantic feature-driven fake news detection model, and explore various variants of convolution neural networks for image-driven fake news detection.

The Structure-aware Multi-Head Attention Network (SMAN) was proposed by Yuan et al [[27](https://link.springer.com/article/10.1007/s11042-023-14883-3#ref-CR27)], which merges the content of news, issuing, and reposting connections of users and publishers, to collectively optimize the credibility prediction tasks and fake news detection. As a result, we can explicitly make use of publishers’ and users’ credibility to detect early fake news. The research conducted experiments on three real-world datasets, and the results show that SMAN can detect fake news in 4 hours with an accuracy of over 91%, which is much faster than the state-of-the-art models.

Dang et al [[7](https://link.springer.com/article/10.1007/s11042-023-14883-3#ref-CR7)] utilize Term Frequency-Inverse Document Frequency (TF-IDF) and word embedding has been implemented to the whole range of datasets to solve the problems in sentiment analysis, for example, sentiment polarity. As a result, comparative knowledge was carried out on the experimental outputs obtained for various designs and input features. The experiments also revealed that CNN outperforms other models, presenting a good balance between accuracy and CPU runtime. RNN reliability is slightly higher than CNN reliability with most datasets but its computational time is much longer. However, exploring hybrid approaches, where multiple models and techniques are combined to enhance the sentiment classification accuracy achieved by the individual models or techniques, as well as to reduce the computational cost is the reliable cost.

Abdullah et al [[14](https://link.springer.com/article/10.1007/s11042-023-14883-3#ref-CR14)] used the multimodal approach with Convolutional Neural Network (CNN) and Long Short-Term memory (LSTM) to classify the fake news articles that achieved significant performance. We worked on a database with 12 different categories of news articles and used linguistic cue approaches with machine learning. We classified news based on its source and its previous history (such as domain name and/ or author name) with bimodal CNN and LSTM. Through reputable news sources, the model classifies reliable news articles with an accuracy of 99.7% on the training data and 97.5% on test data. However, as a piece of fake news can still be published on a reputable domain, we still had to consider other parameters such as news headlines.

Nida Aslam et al [[5](https://link.springer.com/article/10.1007/s11042-023-14883-3#ref-CR5)] proposed an ensemble-based deep learning model to classify news as fake or real using a LIAR dataset. Due to the nature of the dataset attributes, two deep-learning models were used. For the textual attribute “statement,” Bi-LSTM-GRU-dense deep learning model was used, while for the remaining attributes, the dense deep learning model was used. Experimental results showed that the proposed study achieved an accuracy of 0.898, a recall of 0.916, a precision of 0.913, and an F-score of 0.914, respectively, using only statement attributes. Moreover, the outcome of the proposed models is remarkable when compared with that of the previous studies for fake news detection using the LIAR dataset. Despite the significant results achieved by the proposed study, there is still room for improvement. The model needs to be investigated using other fake news datasets.

From the aforementioned studies, the research gap present in each manuscript motivates me to study the hybrid methods for fake news detection. Among the various hybrid methods that exist in the literature, those that model the social graph that spreads the news, or the user and news source features (profile), cannot be applied when only the text of the news is available. From the hybrid methods that examine only the textual content of news, the combination of LSTM and CNN has shown promising results. However, so far, LSTMs have been used for providing word embeddings and CNN for doing the final classification. Accordingly, the research proposed Naïve Bayes, a passive-aggressive classifier and Deep Neural Network (DNN) to be implemented to detect fake news with multivariate missing values to tackle the issue [[16](https://link.springer.com/article/10.1007/s11042-023-14883-3#ref-CR16)]. The passive-aggressive classifier is used for training the ISOT and LIAR dataset and the Naïve Bayes was used to test the model for detecting fake news. Finally, DNN is used for validation purposes which efficiently classify fake and real news. A conclusion to the analysis of the related literature is that fake news has played a significant role in many real-time disasters. In order, manual interventions are of no use due to the multiple datasets which contain information sharing on the internet. Machine learning techniques have experimented on a range of datasets and deep learning techniques are still to be fully evaluated on fake news detection and related tasks. Table [1](https://link.springer.com/article/10.1007/s11042-023-14883-3#Tab1) illustrates the comparison of the state-of-the-art techniques.

**CHAPTER 3**

**EXISTING SYSTEM**

A concept to analyse the safety of women by utilising the messages shared on social networking sites and then using machine learning algorithms to these messages. These days, nearly everyone uses social networking sites to share their thoughts and feelings. If a woman has a bad experience in a certain location, she will express her discomfort through negative language in her posts, tweets, and messages, and by analysing these communications, we can determine which locations are the most dangerous for women.

**Drawback:**

On the other hand, women have the impression that they are in danger when they are in public locations like malls or shopping malls on their route to their place of employment because of the many unseen eyes that shame their bodies and harass them.

**CHAPTER 4**

**PROPOSED SYSTEM**

**4.1 Overview**

In the work that was proposed, we downloaded tweets from Twitter using the TWEEPY package that is available for the Python programming language. However, every time we tried to download tweets online, the Internet was unavailable. As a result, we downloaded MEETOO tweets on women's safety and stored them in a dataset folder. This application will read these tweets in order to determine how ladies are feeling.

* Author is cleaning up tweets by using a programme called NLTK, which stands for natural language tool kit, in order to eliminate special symbols and stop words.
* The author uses the TEXTBLOB corpora package and dictionary to count positive, negative, and neutral polarity. Tweets with a polarity value of less than 0 are considered to have a negative polarity, while tweets with a polarity value of greater than 0 and less than 0.5 are considered to have a neutral polarity. Tweets with a polarity value of greater than 0.5 are considered to have a positive polarity.

every city is the prevalence of sexual harassment and assault. In addition, the harassing and violent content that can be found in online social networking sites can have a negative impact on the personal lives of women. As a result, it is essential to determine whether or not the OSN environment is safe for women. The conventional methods, on the other hand, were not successful in predicting the maximum safety analysis.

Figure 4.1 shows the proposed WPC-DT block diagram. Initially, dataset is collected using “TWEEPY” package, which download tweets from internet. The dataset mostly contains the “MEETOO” hashtag-based tweets. These tweets are specially focused on women safety issue. Then, the dataset is pre-processed using NLTK. Here, NLTK is used to identify stop words, and remove special symbols from tweet dataset. The NLTK also eliminates unknown characters, symbols, special letters from dataset. The empty samples are replaced by zeros, which resulted in pre-processed and normalized data.

Then, Textblob is used to count the positive, negative, and neutral polarity tweets. Tweets with polarity values less than 0 are considered negative, tweets with polarity values greater than 0 are considered neutral, and tweets with polarity greater than 0.5 are considered positive. Further, TF-IDF method is used to extract the data specific features. In addition, DT classifier trained with TF-IDF features. Finally, The DT classifier predicts the tweet status as “Genuine tweet” or “Fake tweet” by using sentiment analysis.

A diagram of a data processing process

Description automatically generated

Figure 4.1. Block diagram of proposed system.

**4.2 Data Preprocessing**

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task.A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

**Step 1:** Load the twitter Dataset: In this step, load the dataset into your data analysis environment. This dataset typically includes text reviews (the input) and corresponding labels (the output), which can be sentiments like "positive" or "negative." The goal is to train a model to predict these labels based on the text.

**Step 2:** Data Cleaning: Data cleaning involves several sub-steps: Removing Special Characters and Punctuation: Special characters (e.g., @, $, %) and punctuation (e.g., !, ?, .) are often irrelevant to sentiment analysis and can be removed to focus on the actual text content. Handling Irrelevant Information: Sometimes, there might be metadata or other information in the text data that's not relevant to the analysis. You should remove such information to concentrate on the review text itself.

**Step 3**: Tokenization:

* Tokenization is the process of splitting the text into smaller units, such as words or phrases (tokens). This step is crucial because it breaks down the text into manageable pieces for further analysis.
* For example, the sentence "I love this hotel" would be tokenized into ["I", "love", "this", "hotel"].

**Step 4:** Convert Text to Lowercase:

* Converting all text to lowercase ensures consistency in your text data. It prevents the model from treating "good" and "Good" as two different words, which could lead to incorrect feature extraction and modeling.
* For example, "Good" and "good" should be treated as the same word.

**Step 5:** Remove Stop Words:

* Stop words are common words in a language that often don't carry significant meaning and can be safely removed to reduce noise in the data. Examples include "the," "is," "and," "in," etc.
* Removing stop words can help improve the efficiency of the model and reduce the dimensionality of the data without losing much valuable information.

**Step 6:** Apply Stemming or Lemmatization:

* Stemming and lemmatization are techniques used to reduce words to their root form, which helps in standardizing words and improving feature extraction.
* Stemming: It involves removing suffixes from words to obtain the word stem. For example, "jumping" becomes "jump," "flies" becomes "fli," etc. Stemming is more aggressive but may result in non-words.
* Lemmatization: It is a more advanced technique that reduces words to their base or dictionary form (lemma). For example, "better" becomes "good," "running" becomes "run," etc. Lemmatization is more accurate but computationally expensive.
* The choice between stemming and lemmatization depends on your specific NLP task and dataset.

**4.3 Dataset Splitting**

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So, we always try to make a machine learning model which performs well with the training set and with the test dataset.

**Training** **Set**: A subset of dataset to train the machine learning model, and we already know the output.

**Test** **set**: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

**4.4 TF-IDF Feature Extraction**

TF-IDF which stands for Term Frequency – Inverse Document Frequency. It is one of the most important techniques used for information retrieval to represent how important a specific word or phrase is to a given document. Let’s take an example, we have a string or Bag of Words (BOW) and we have to extract information from it, then we can use this approach.

Figure 4.2 shows the TF-IDF feature extraction block diagram. The tf-idf value increases in proportion to the number of times a word appears in the document but is often offset by the frequency of the word in the corpus, which helps to adjust with respect to the fact that some words appear more frequently in general. TF-IDF use two statistical methods, first is Term Frequency and the other is Inverse Document Frequency. Term frequency refers to the total number of times a given term t appears in the document doc against (per) the total number of all words in the document and the inverse document frequency measure of how much information the word provides. It measures the weight of a given word in the entire document. IDF show how common or rare a given word is across all documents. TF-IDF can be computed as .

Diagram

Description automatically generated

Fig. 4.2: TF-IDF block diagram.

TF-IDF do not convert directly raw data into useful features. Firstly, it converts raw strings or dataset into vectors and each word has its own vector. Then we’ll use a particular technique for retrieving the feature like Cosine Similarity which works on vectors, etc.

Terminology

t — term (word)

d — document (set of words)

N — count of corpus

corpus — the total document set

**Step 1: Term Frequency (TF):** Suppose we have a set of English text documents and wish to rank which document is most relevant to the query, “Data Science is awesome!” A simple way to start out is by eliminating documents that do not contain all three words “Data” is”, “Science”, and “awesome”, but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document; the number of times a term occurs in a document is called its term frequency. The weight of a term that occurs in a document is simply proportional to the term frequency.

**Step 2: Document Frequency:** This measures the importance of document in whole set of corpora, this is very similar to TF. The only difference is that TF is frequency counter for a term t in document d, whereas DF is the count of occurrences of term t in the document set N. In other words, DF is the number of documents in which the word is present. We consider one occurrence if the term consists in the document at least once, we do not need to know the number of times the term is present.

**Step 3: Inverse Document Frequency (IDF):** While computing TF, all terms are considered equally important. However, it is known that certain terms, such as “is”, “of”, and “that”, may appear a lot of times but have little importance. Thus, we need to weigh down the frequent terms while scale up the rare ones, by computing IDF, an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. The IDF is the inverse of the document frequency which measures the informativeness of term t. When we calculate IDF, it will be very low for the most occurring words such as stop words (because stop words such as “is” is present in almost all of the documents, and N/df will give a very low value to that word). This finally gives what we want, a relative weightage.

Now there are few other problems with the IDF, in case of a large corpus, say 100,000,000, the IDF value explodes, to avoid the effect we take the log of idf . During the query time, when a word which is not in vocab occurs, the df will be 0. As we cannot divide by 0, we smoothen the value by adding 1 to the denominator.

The TF-IDF now is at the right measure to evaluate how important a word is to a document in a collection or corpus. Here are many different variations of TF-IDF but for now let us concentrate on this basic version.

**4.4 DTC Classifier**

DTC is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "DTC is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the DTC takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Diagram

Description automatically generated

Fig. 4.2: DTC algorithm.

**4.4.1 DTC algorithm**

Step 1: In DTC n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

**4.4.2 Important Features of DTC**

* **Diversity**- Not all attributes/variables/features are considered while making an individual tree, each tree is different.
* **Immune** **to** **the** **curse** **of** **dimensionality**- Since each tree does not consider all the features, the feature space is reduced.
* **Parallelization**-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build DTCs.
* **Train-Test** **split**- In a DTC we don’t have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
* **Stability**- Stability arises because the result is based on majority voting/ averaging.

**4.4.3 Assumptions for DTC**

Since the DTC combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better DTC classifier:

* There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
* The predictions from each tree must have very low correlations.

Below are some points that explain why we should use the DTC algorithm

* It takes less training time as compared to other algorithms.
* It predicts output with high accuracy, even for the large dataset it runs efficiently.
* It can also maintain accuracy when a large proportion of data is missing.

**4.4.4 Types of Ensembles**

Before understanding the working of the DTC, we must look into the ensemble technique. Ensemble simply means combining multiple models. Thus, a collection of models is used to make predictions rather than an individual model. Ensemble uses two types of methods:

**Bagging**– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, DTC. Bagging, also known as Bootstrap Aggregation is the ensemble technique used by DTC. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.

**Boosting**– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.

**4.5 Advantages of proposed system**

Decision trees are a popular machine learning algorithm with several advantages that make them useful for various tasks, including classification and regression. Here are some of the key advantages of decision trees:

* **Interpretability:** Decision trees are highly interpretable. The model's decision-making process can be easily visualized and understood, making it a valuable tool for explaining the logic behind predictions. This feature is particularly important in fields where model transparency is critical, such as healthcare and finance.
* **No Assumptions about Data:** Decision trees do not require any assumptions about the data distribution or relationships between variables. They can handle both numerical and categorical data, making them versatile for different types of datasets.
* **Handling Non-Linear Relationships:** Decision trees can model complex, non-linear relationships in the data effectively. They can capture interactions between features without the need for feature engineering or manual transformation.
* **Feature Importance:** Decision trees provide a built-in feature importance ranking. By examining the splits in the tree and how much they improve the model's performance, you can identify which features are most relevant for making predictions. This information can guide feature selection and data understanding.
* **Scalability:** Decision trees are relatively computationally efficient, especially when compared to more complex algorithms like deep neural networks. They can handle large datasets with many features without significant computational overhead.
* **Handling Missing Values:** Decision trees can naturally handle missing values by making decisions based on the available information. They do not require imputation or preprocessing steps to handle missing data.
* **Robustness to Outliers:** Decision trees are robust to outliers in the data. Outliers may affect individual splits in the tree, but they are less likely to have a significant impact on the overall model's performance.
* **Ensemble Learning:** Decision trees can be used as building blocks for ensemble methods like Random Forests and Gradient Boosting, which often outperform single decision trees by reducing overfitting and improving generalization.
* **Non-Parametric:** Decision trees are non-parametric models, meaning they do not make assumptions about the underlying data distribution. This makes them flexible and adaptable to various types of data.
* **Easy to Implement:** Decision trees are relatively simple to implement from scratch, making them accessible to those with basic programming skills. Additionally, there are numerous libraries and software packages that provide decision tree implementations.
* The analysis of the collection of texts on Twitter contains the names of persons as well as the names of women who speak out against abuse, harassment, and unethical behaviour on the part of men in Indian cities, which makes it unpleasant for them to move freely. These women say that the behaviour of men in these cities makes it difficult for them to move freely.
* The collecting of data that was obtained on the precarious position of women in Indian society via the use of Twitter.

**CHAPTER 5**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

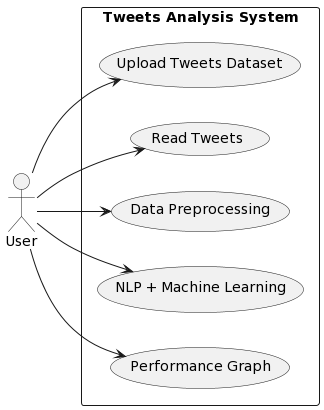
The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:** The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

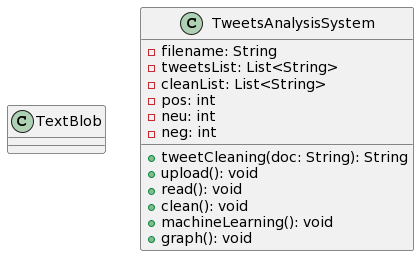
**USE CASE DIAGRAM**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



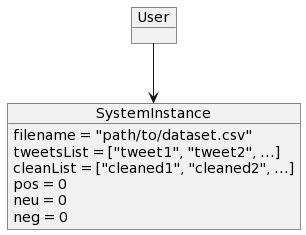
**Class diagram**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.



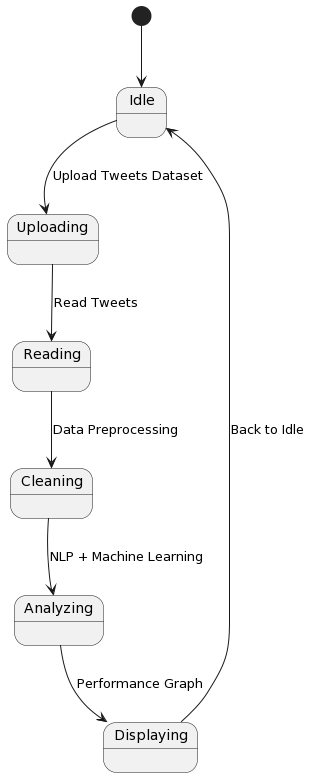
**Object diagram**

The object diagram is a special kind of class diagram. An object is an instance of a class. This essentially means that an object represents the state of a class at a given point of time while the system is running. The object diagram captures the state of different classes in the system and their relationships or associations at a given point of time.



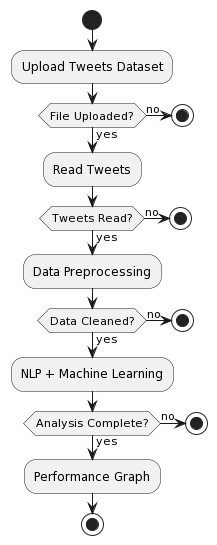
**State diagram**

A state diagram, as the name suggests, represents the different states that objects in the system undergo during their life cycle. Objects in the system change states in response to events. In addition to this, a state diagram also captures the transition of the object's state from an initial state to a final state in response to events affecting the system.



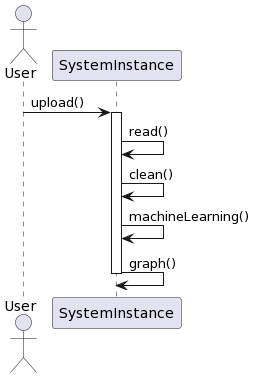
**Activity diagram**

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.



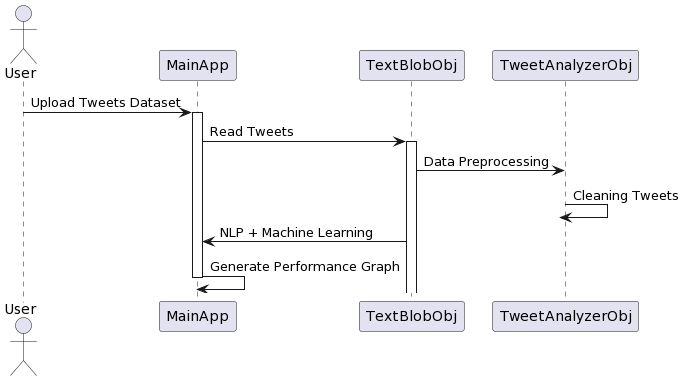
**Sequence diagram**

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".



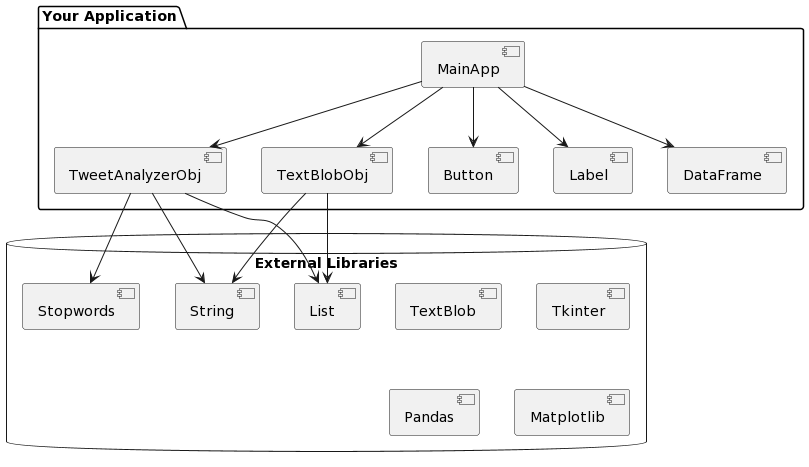
**Collaboration diagram**

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.



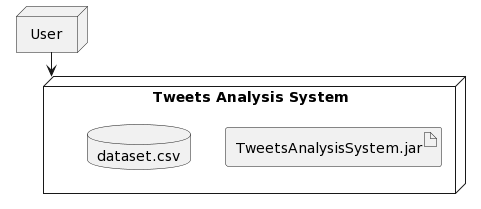
**Component diagram:**

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.



**Deployment diagram:**

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.



**CHAPTER 6**

**SOFTWARE ENVIRONMENT**

**What is Python?**

Below are some facts about Python.

* Python is currently the most widely used multi-purpose, high-level programming language.
* Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
* Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
* Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

* Machine Learning
* GUI Applications (like Kivy, Tkinter, PyQt etc. )
* Web frameworks like Django (used by YouTube, Instagram, Dropbox)
* Image processing (like Opencv, Pillow)
* Web scraping (like Scrapy, BeautifulSoup, Selenium)
* Test frameworks
* Multimedia

**Advantages of Python**

Let’s see how Python dominates over other languages.

**1. Extensive Libraries**

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

**2. Extensible**

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

**3. Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

**4. Improved Productivity**

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

**5. IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

**6. Simple and Easy**

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

**7. Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

**8. Object-Oriented**

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

**9. Free and Open-Source**

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

**10. Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

**11. Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

**Advantages of Python Over Other Languages**

**1. Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

**2. Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

**3. Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

**Disadvantages of Python**

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

**1. Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

**2. Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

**3. Design Restrictions**

As you know, Python is dynamically typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

**4. Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

**5. Simple**

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

**History of Python**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

**Python Development Steps**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.

Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

Print is now a function.

* Views and iterators instead of lists
* The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
* There is only one integer type left, i.e., int. long is int as well.
* The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
* Text Vs. Data Instead of Unicode Vs. 8-bit

**Purpose**

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

**Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Modules Used in Project**

**TensorFlow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

**NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Install Python Step-by-Step in Windows and Mac**

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

**How to Install Python on Windows and Mac**

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

**Download the Correct version into the system**

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

A screenshot of a computer

Description automatically generated with medium confidence

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Graphical user interface, application

Description automatically generated

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Graphical user interface, application

Description automatically generated

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Graphical user interface, text

Description automatically generated

* To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
* To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

**Installation of Python**

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.

Graphical user interface, text, application

Description automatically generated

Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

Graphical user interface, text, application, chat or text message

Description automatically generated

Step 3: Click on Install NOW After the installation is successful. Click on Close.

Graphical user interface, text, application, chat or text message

Description automatically generated

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

**Verify the Python Installation**

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.

Graphical user interface, application

Description automatically generated

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.

A screenshot of a computer

Description automatically generated with medium confidence

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

**Check how the Python IDLE works**

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.

Application

Description automatically generated with low confidence

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save

Graphical user interface, text, application, email

Description automatically generated

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print (“Hey World”) and Press Enter.

Graphical user interface, text, application, email

Description automatically generated

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won’t work.

**CHAPTER 7**

**SYSTEM REQUIREMENTS**

**Software Requirements**

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

* Python IDLE 3.7 version (or)
* Anaconda 3.7 (or)
* Jupiter (or)
* Google colab

**Hardware Requirements**

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

* Operating system : Windows, Linux
* Processor : minimum intel i3
* Ram : minimum 4 GB
* Hard disk : minimum 250GB

**CHAPTER 8**

**FUNCTIONAL REQUIREMENTS**

**Output Design**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

**Output Definition**

The outputs should be defined in terms of the following points:

* Type of the output
* Content of the output
* Format of the output
* Location of the output
* Frequency of the output
* Volume of the output
* Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

**Input Design**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**Input Stages**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**Input Types**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**Input Media**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**Error Avoidance**

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

**Error Detection**

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

**Data Validation**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

**User Interface Design**

It is essential to consult the system users and discuss their needs while designing the user interface:

**User Interface Systems Can Be Broadly Clasified As:**

* User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
* Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**User Initiated Interfaces**

User initiated interfaces fall into two approximate classes:

* Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
* Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

**Computer-Initiated Interfaces**

The following computer – initiated interfaces were used:

* The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
* Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**Error Message Design**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

**Performance Requirements**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system
* The existing system is completely dependent on the user to perform all the duties.

**CHAPTER 9**

**SOURCE CODE**

import tkinter

from textblob import TextBlob

from tkinter import \*

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from string import punctuation

from nltk.corpus import stopwords

main = tkinter.Tk()

main.title("Polarity score-based Supervised Learning for Classification of Fake and Authentic Tweets using NLP") #designing main screen

main.geometry("1300x1200")

global filename

tweets\_list = []

clean\_list = []

global pos, neu, neg

def tweetCleaning(doc):

    tokens = doc.split()

    table = str.maketrans('', '', punctuation)

    tokens = [w.translate(table) for w in tokens]

    tokens = [word for word in tokens if word.isalpha()]

    stop\_words = set(stopwords.words('english'))

    tokens = [w for w in tokens if not w in stop\_words]

    tokens = [word for word in tokens if len(word) > 1]

    tokens = ' '.join(tokens) #here upto for word based

    return tokens

def upload(): #function to upload tweeter profile

    global filename

    filename = filedialog.askopenfilename(initialdir="dataset")

    pathlabel.config(text=filename)

    text.delete('1.0', END)

    text.insert(END,filename+" loaded\n");

def read():

    text.delete('1.0', END)

    tweets\_list.clear()

    train = pd.read\_csv(filename,encoding='iso-8859-1')

    for i in range(len(train)):

        tweet = train.\_get\_value(i, 'Text')

        tweets\_list.append(tweet)

        text.insert(END,tweet+"\n")

    text.insert(END,"\n\nTotal tweets found in dataset is : "+str(len(tweets\_list))+"\n\n\n")

def clean():

    text.delete('1.0', END)

    clean\_list.clear()

    for i in range(len(tweets\_list)):

        tweet = tweets\_list[i]

        tweet = tweet.strip("\n")

        tweet = tweet.strip()

        tweet = tweetCleaning(tweet.lower())

        clean\_list.append(tweet)

        text.insert(END,tweet+"\n")

    text.insert(END,"\n\nTotal tweets found in dataset is : "+str(len(clean\_list))+"\n\n\n")

def machineLearning():

    text.delete('1.0', END)

    global pos, neu, neg

    pos = 0

    neu = 0

    neg = 0

    for i in range(len(clean\_list)):

        tweet = clean\_list[i]

        blob = TextBlob(tweet)

        if blob.polarity <= 0.5:

            neg = neg + 1

            text.insert(END,tweet+"\n")

            text.insert(END,"Predicted Sentiment : FAKE\n")

            text.insert(END,"Polarity Score      : "+str(blob.polarity)+"\n")

            text.insert(END,'====================================================================================\n')

        if blob.polarity > 0.51:

            pos = pos + 1

            text.insert(END,tweet+"\n")

            text.insert(END,"Predicted Sentiment : AUTHENTICATED\n")

            text.insert(END,"Polarity Score      : "+str(blob.polarity)+"\n")

            text.insert(END,'====================================================================================\n')

def graph():

    label\_X = []

    category\_X = []

    text.delete('1.0', END)

    text.insert(END,"Saftey Factor\n\n")

    text.insert(END,'Authenticated : '+str(pos)+"\n")

    text.insert(END,'Fake : '+str(neg)+"\n")

    text.insert(END,'Length of tweets  : '+str(len(clean\_list))+"\n")

    text.insert(END,'Authenticated : '+str(pos)+' / '+ str(len(clean\_list))+' = '+str(pos/len(clean\_list))+'%\n')

    text.insert(END,'Fake : '+str(neg)+' / '+ str(len(clean\_list))+' = '+str(neg/len(clean\_list))+'%\n')

    label\_X.append('Authenticated')

    label\_X.append('Fake')

    category\_X.append(pos)

    category\_X.append(neg)

    plt.pie(category\_X,labels=label\_X,autopct='%1.1f%%')

    plt.title('Sentiment Graph')

    plt.axis('equal')

    plt.show()

font = ('times', 16, 'bold')

title = Label(main, text='Polarity score-based Supervised Learning for Classification of Fake and Authentic Tweets using NLP')

title.config(bg='LightSteelBlue', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 14, 'bold')

uploadButton = Button(main, text="Upload Tweets Dataset", command=upload)

uploadButton.place(x=50,y=100)

uploadButton.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='sky blue3', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=370,y=100)

readButton = Button(main, text="Read Tweets", command=read)

readButton.place(x=50,y=150)

readButton.config(font=font1)

cleanButton = Button(main, text="Data Preprocessing", command=clean)

cleanButton.place(x=210,y=150)

cleanButton.config(font=font1)

mlButton = Button(main, text="NLP + Machine Learning", command=machineLearning)

mlButton.place(x=470,y=150)

mlButton.config(font=font1)

graphButton = Button(main, text="Performance Graph", command=graph)

graphButton.place(x=730,y=150)

graphButton.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=25,width=150)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=200)

text.config(font=font1)

main.config(bg='LightSteelBlue')

main.mainloop()

**CHAPTER 10**

**RESULTS AND DISCUSSION**

**10.1 Implementation description**

This is a Python script that creates a graphical user interface (GUI) using the tkinter library for a machine learning application. The application's purpose is to analyze tweets related to fake and authentic. It performs the following tasks:

* GUI Initialization:

It starts by creating a tkinter window named main.

The title of the window is set to "'Polarity score-based Supervised Learning for Classification of Fake and Authentic Tweets using NLP'".

The window's dimensions are set to 1300x1200 pixels.

* Importing Libraries:

The script imports necessary libraries including tkinter for GUI, TextBlob for sentiment analysis, matplotlib for plotting graphs, numpy and pandas for data manipulation, and others.

* Function Definitions:

tweetCleaning: Cleans a given tweet by removing punctuation, non-alphabetic characters, stopwords, and short words.

* upload: Allows the user to upload a dataset containing tweets.

read: Reads and displays the tweets from the uploaded dataset.

clean: Cleans the tweets using the tweetCleaning function.

machineLearning: Applies sentiment analysis using TextBlob and categorizes tweets into positive, negative, or neutral sentiments.

graph: Generates a pie chart to visualize the distribution of sentiments.

* Global Variables:

filename: Holds the path of the uploaded file.

tweets\_list: Stores the original tweets.

clean\_list: Stores the cleaned tweets.

pos, neu, neg: Counters for positive, neutral, and negative sentiments.

* Button Definitions:

Buttons are created for actions like uploading a dataset, reading tweets, cleaning tweets, applying machine learning, and generating a graph. Each button is associated with a specific function.

* Label and Text Box:

Labels and a text box are added to the GUI for displaying information and results.

* Execution:

The GUI window is configured with background color and launched with main.mainloop().

* Event Handling:

When a button is clicked, it triggers the associated function, performing the desired action.

* Displaying Results:

The GUI displays various information like the uploaded file's path, total number of tweets, cleaned tweets, and sentiment analysis results.

* Graph Plotting:

The graph function displays a pie chart showing the distribution of positive, negative, and neutral sentiments.

This script utilizes various libraries and functions for specific tasks, such as TextBlob for sentiment analysis and tkinter for GUI. The application is designed to be user-friendly, allowing the user to interact with it through the GUI.

* 1. **Dataset description**

The given dataset appears to contain information related to tweets, likely discussing topics related to the #MeToo movement and other social issues. Here's a breakdown of the columns in the dataset:

Text: The content of the tweet.

Id: Unique identifier for each tweet.

Length: Length of the tweet.

Created\_at: Timestamp indicating when the tweet was created.

Source: The platform or application used to post the tweet (e.g., Twitter for Android, Twitter for iPhone).

Favorite\_count: The number of users who marked the tweet as a favorite.

Retweet\_count: The number of times the tweet has been retweeted.

Lang: The language of the tweet.

A few observations:

The dataset seems to include both text data (the content of the tweets) and numerical data (tweet ID, length, favorite count, retweet count).

The dataset captures information about the source platform, which may be useful for analyzing user behavior across different platforms.

The timestamp provides information about when the tweets were posted, allowing for temporal analysis.

* 1. **Results and Description**

Figure 10.1 portrays the graphical user interface (GUI) design of this work's proposed system. This interface serves as the visual gateway through which users engage with the system's functionalities. Within this graphical environment, various interactive elements are likely presented, such as buttons, input fields, and display areas. Users can utilize these elements to perform a range of actions, including uploading datasets, initiating nlp processing, and reviewing the results. The GUI is a critical aspect of this work as it acts as a bridge between users and the system's underlying processes. It provides an intuitive means for users to navigate, input data, trigger operations, and visualize the outcomes of fake and authentic tweets, enhancing the overall user experience and facilitating efficient interaction with the system.

Figure 10.2 likely illustrates the process of uploading and reading datasets within the system. This image could depict a screen or dialog where users can select and import datasets into the system. It signifies the initial step in data analysis, allowing users to provide the raw data that will be subsequently processed and analyzed by the system.

Figure 10.3 showcases a snapshot of the dataset post-preprocessing. The preprocessing phase involves a series of steps like data cleaning, tokenization, and removal of irrelevant information. This figure may present a view of the dataset in a more organized and structured format, prepared for analysis. It provides users with an insight into how the data has been refined for subsequent tasks.

Figure 10.4 appears to exhibit the outcomes of sentiment analysis conducted on a test tweet. It possibly displays information such as the system's prediction regarding the sentiment of the tweet (whether it's negative, neutral, or positive) and a polarity score, typically ranging from 0 to 1, indicating the strength or intensity of the sentiment expressed in the tweet. This visual aids users in understanding how the system categorizes and scores sentiment within individual tweets.

Figure 10.5 likely provides a visual representation, such as a bar or pie chart, demonstrating the distribution of sentiment within the analyzed dataset. The percentages mentioned in your description (e.g., 74.6% negative, 22.3% neutral, 3.1% positive) indicate the proportion of tweets falling into each sentiment category. This graphical representation offers an at-a-glance overview of the sentiment landscape within the dataset, helping users assess sentiment patterns and trends.

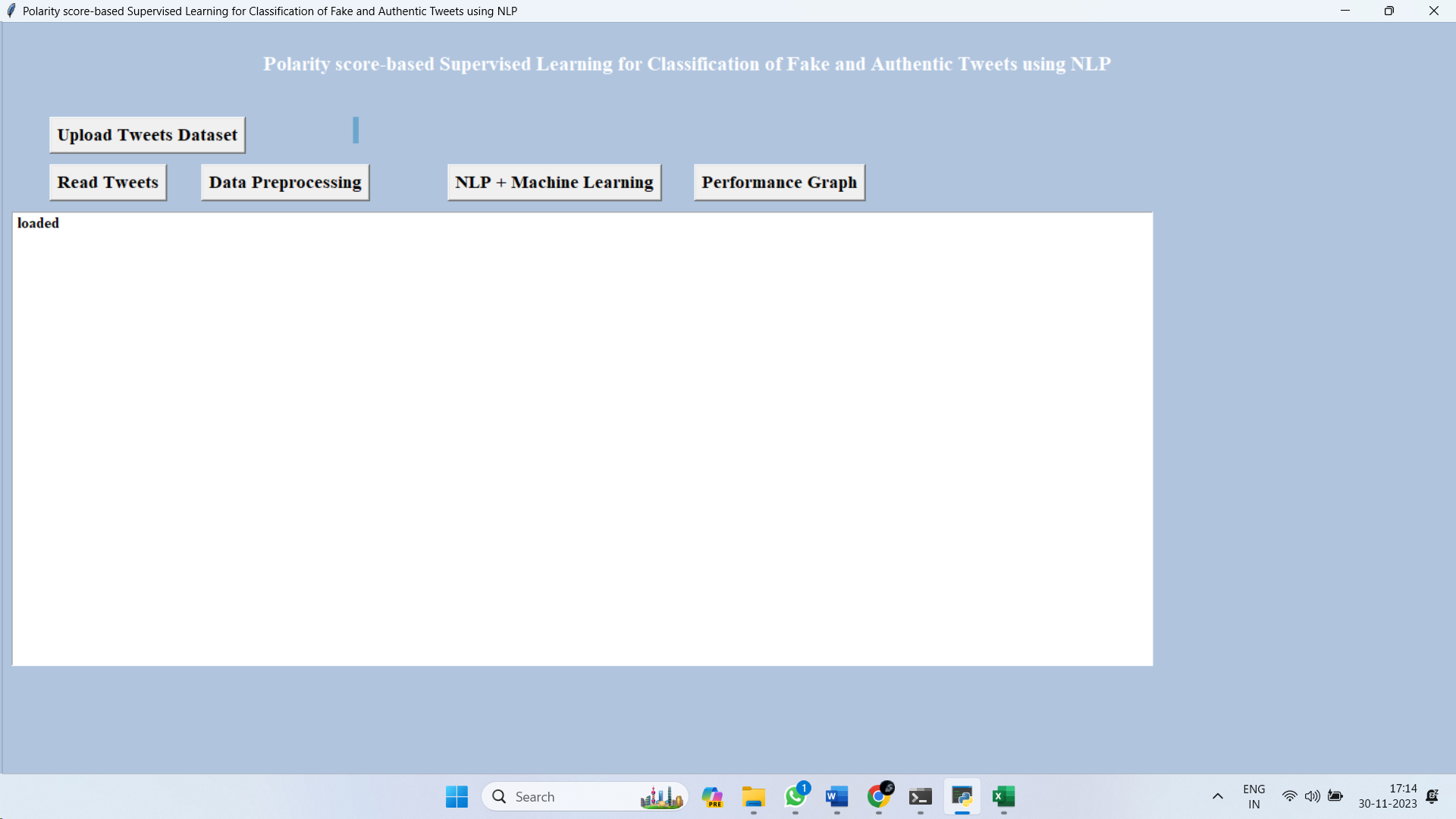


Figure 10.1: User interface of proposed system.

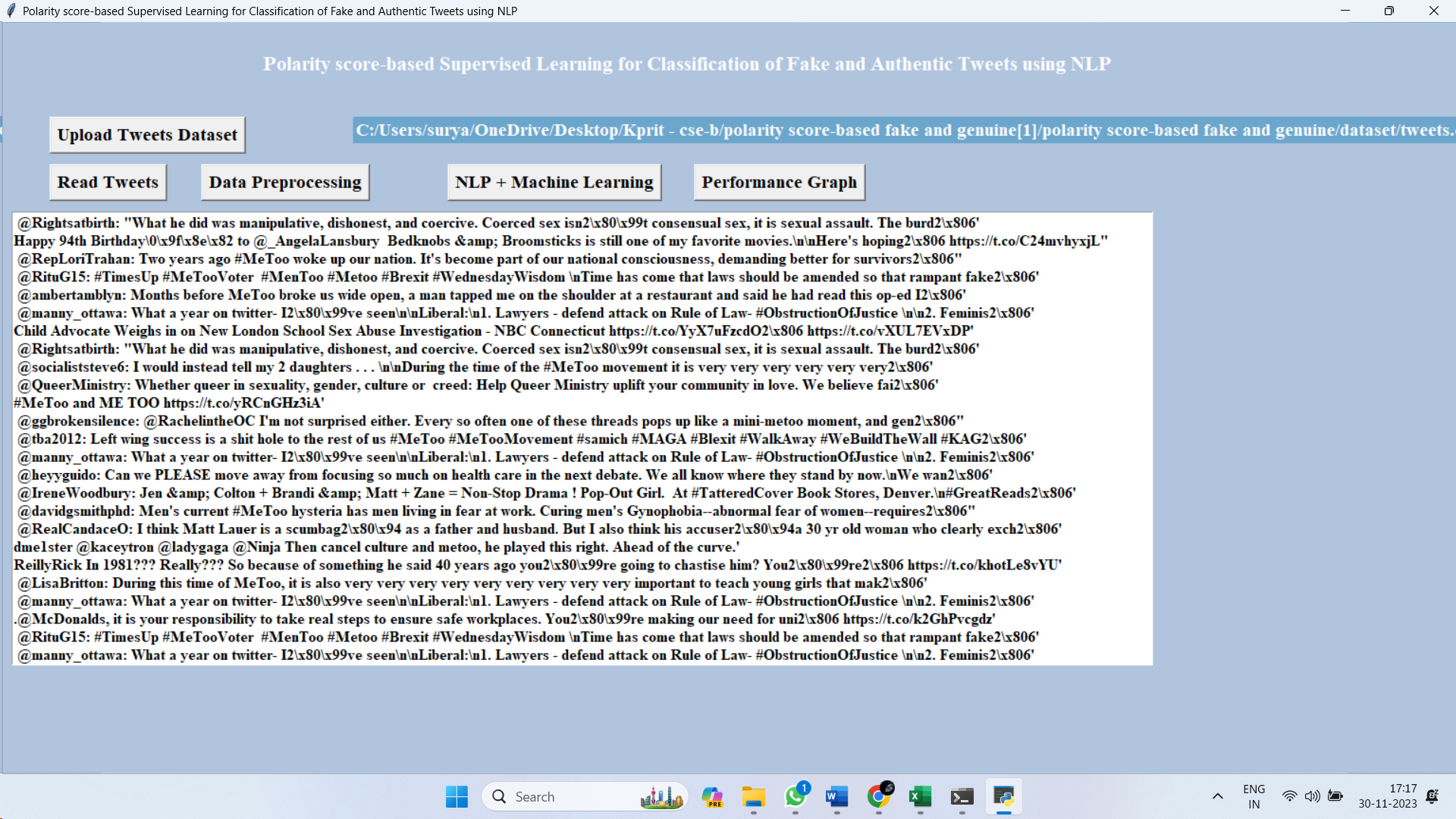


Figure 10.2: Dataset upload and reading.

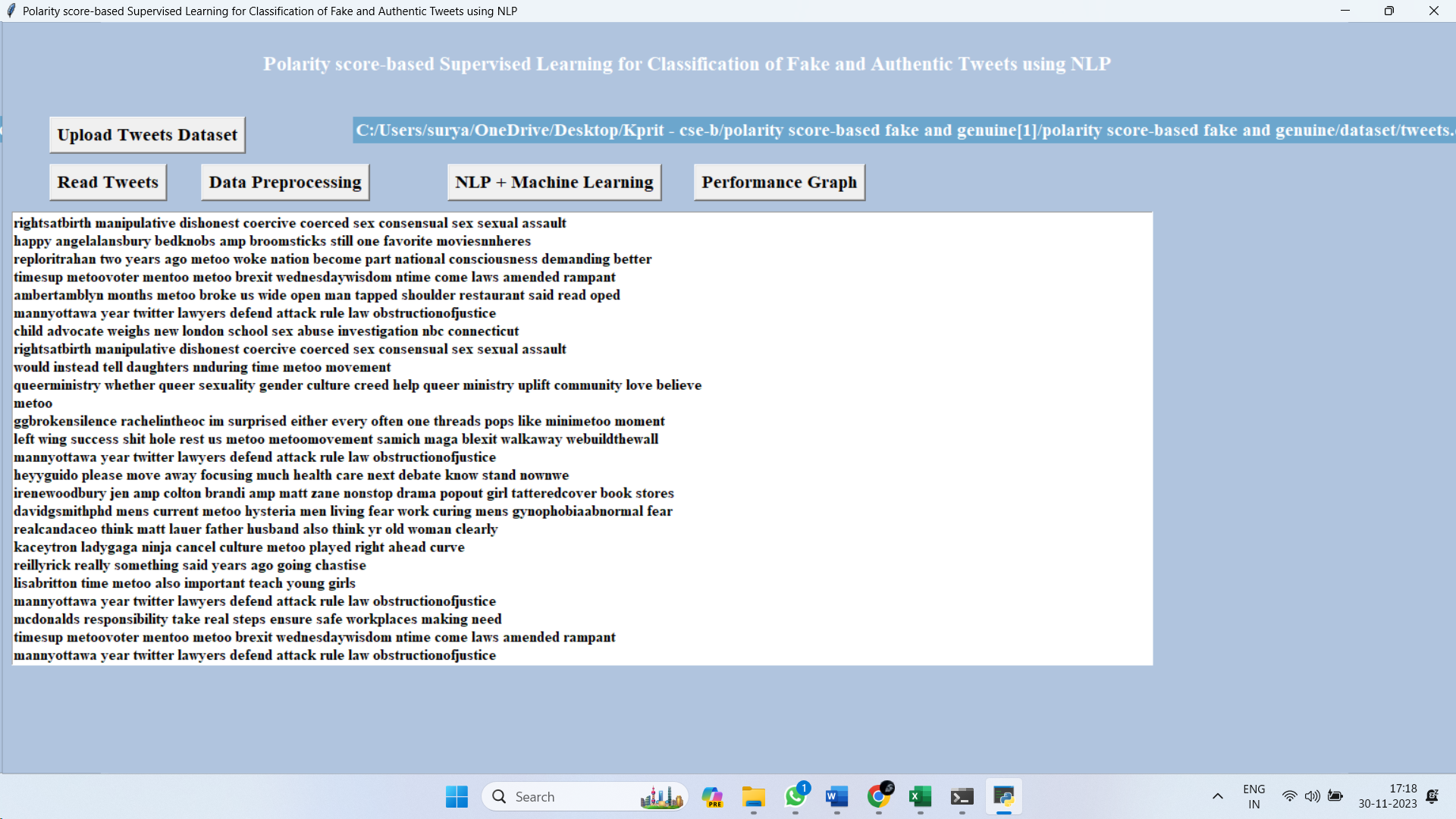


Figure 10.3: Dataset after preprocessing.

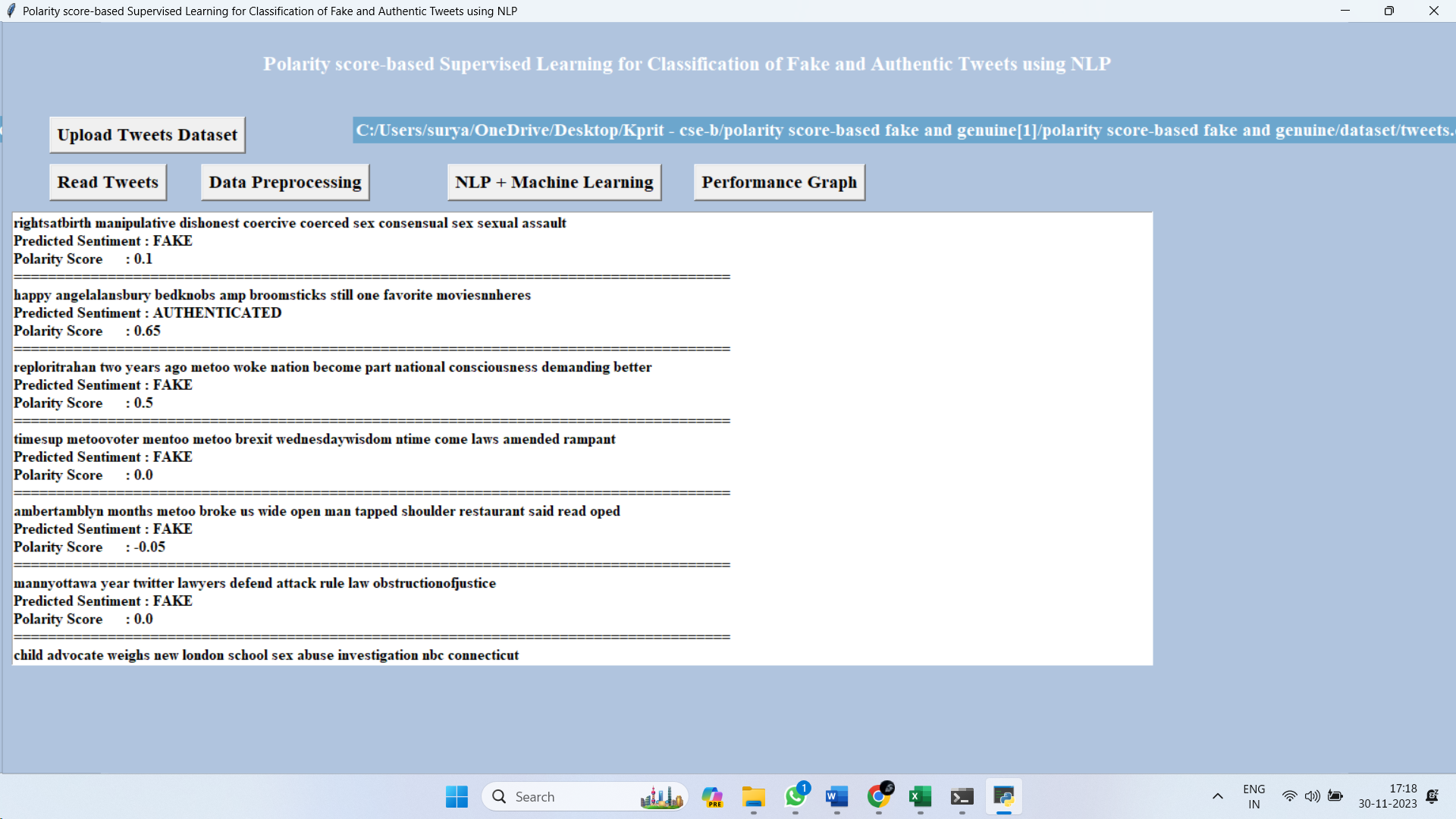


Figure 10.4: Prediction results from test tweet.

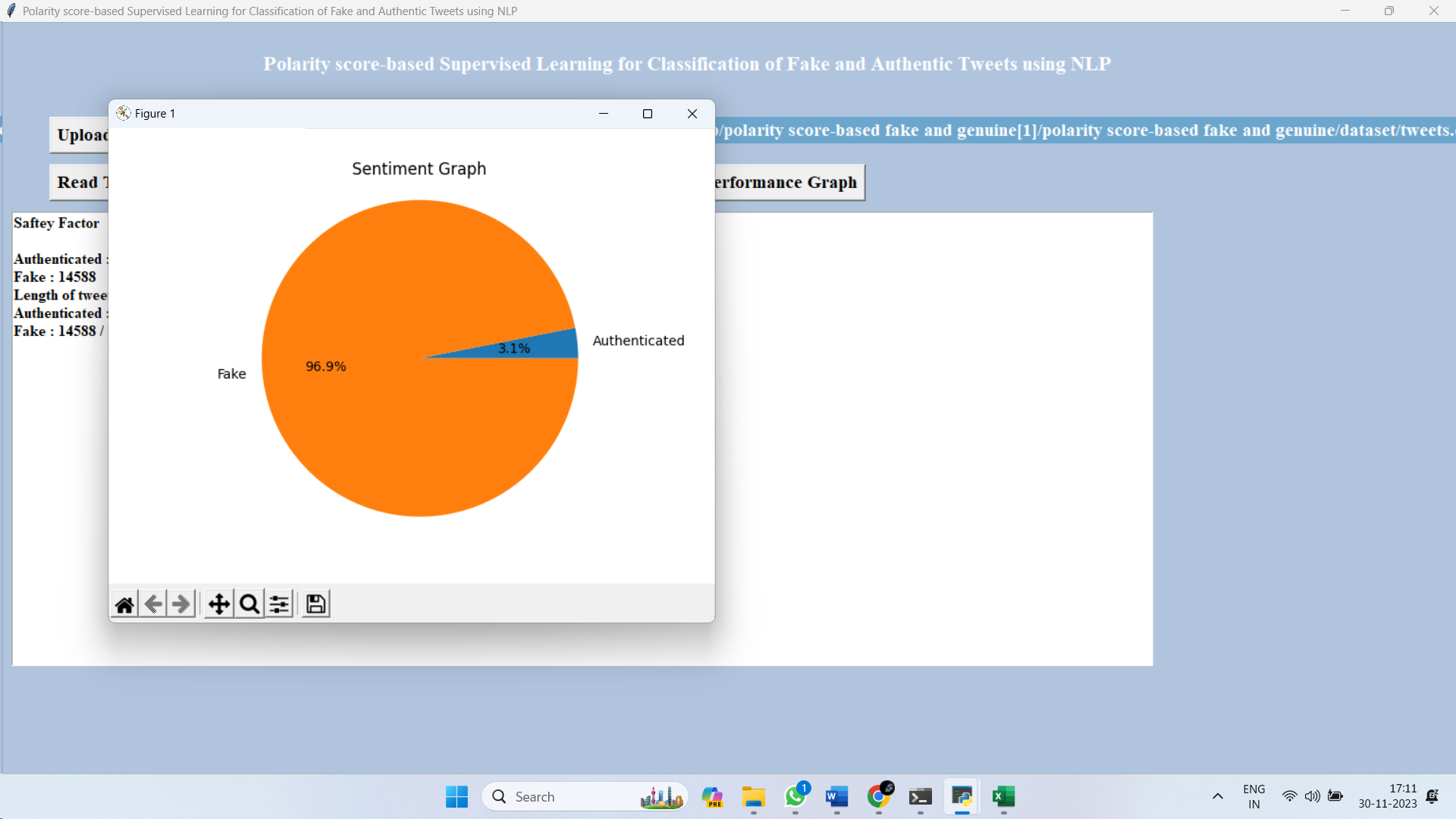


Figure 10.5: Sentiment graph performance measurement.

**CHAPTER 11**

**CONCLUSION AND FUTURE SCOPE**

**11.1 Conclusion**

One of the most significant challenges that women face in any community is the prevalence of sexual harassment and assault. In addition, the women's private lives are negatively impacted because of the bullying and abusive information that is displayed in OSN. As a result, it is essential to determine whether or not the OSN environment is safe for women. The standard approaches were not successful in predicting the maximum safety analysis. Therefore, the WSP-DT classifier is the primary focus of our study. At first, it is thought that the Twitter dataset would be used to build the whole system. After that, the dataset will be pre-processed to get rid of any unknown or missing symbols. Following that, NLTK was applied to tweets in order to accomplish the tasks of tokenization, conversion to lowercase, detection of stop words, stemming, and lemmatization. Then, a text blob protocol is built to detect the feelings of tweets that have already been pre-processed. This protocol determines the positive, negative, and neutral polarities of tweets. In addition, TF-IDF is used in order to extract the data characteristics based on the frequency of individual words and characters. In the end, a decision tree classifier was used to determine whether a tweet was phony or authentic based on the previous multi-level training. The simulations that were run on the Twitter dataset reveal that the suggested WSP-DT classifier produced better results than the other approaches when compared to those results.

Throughout the whole of this research work, we have spoken about a wide range of machine learning techniques. These algorithms can provide us with assistance in organising and analysing the vast quantity of data that we have gotten from Twitter. This data consists of the millions of tweets and text messages that are exchanged on a daily basis. These techniques of machine learning are highly effective and beneficial when it comes to the processing of enormous volumes of data. This comprises the SPC method and strategies based on the linear algebraic Factor Model. Both of these assist in further classifying the data into appropriate categories and are included below. In the process of collecting meaningful information from Twitter and acquiring an idea about the present scenario surrounding the safety of women in metropolitan areas in India, support vector machines are yet another sort of machine learning technique that is very popular.

**11.2 Future Scope**

Because this study is just focusing on Twitter, there is a possibility that in the not-too-distant future we may be able to extend the applicability of these machine learning algorithms to other social media platforms. Twitter is the only platform currently being considered for this study. Facebook and Instagram are two examples of these websites. It is possible that the current concept, which is being supplied, will be included into the user experience of the Twitter programme. This would allow Twitter to reach a wider audience and do sentiment analysis on millions of tweets in order to provide further safety.

**REFERENCES**

1. Gamon and Michael. “Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis”, Proceedings of the 20th international conference on Computational Linguistics. Association for Computational Linguistics, 2004.
2. Agarwal, Apoorv, Fadi Biadsy, and Kathleen R. Mckeown. “Contextual phrase-level polarity analysis using lexical affect scoring and syntactic n-grams”, Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2009.
3. Barbosa, Luciano, and Junlan Feng. “Robust sentiment detection on twitter from biased and noisy data”, Proceedings of the 23rd international conference on computational linguistics: posters. Association for Computational Linguistics, 2010.
4. Bermingham, Adam, and A. F. Smeaton. “Classifying sentiment in microblogs: is brevity an advantage?”, Proceedings of the 19th ACM international conference on Information and knowledge management. ACM, 2010.
5. V. Sahayak, V. Shete, and A. Pathan (2015). “Sentiment analysis on twitter data. International Journal of Innovative Research in Advanced Engineering (IJIRAE)”, 2(1), 178-183.
6. N. Mamgain, E. Mehta, A. Mittal and G. Bhatt, “Sentiment analysis of top colleges in India using Twitter data”, 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), 2016, pp. 525-530, doi: 10.1109/ICCTICT.2016.7514636.
7. B. Gupta, M. Negi, K. Vishwakarma, G. Rawat, and P. Badhani (2017). “Study of Twitter sentiment analysis using machine learning algorithms on Python”. International Journal of Computer Applications, 165(9), 0975-8887.
8. Reyes-Menendez, J. R. Saura, and C. Alvarez Alons. “Understanding# World Environment Day user opinions in Twitter: A topic-based sentiment analysis approach”. International journal of environmental research and public health. 2018 Nov;15(11):2537.
9. D. Kumar and S. Aggarwal. “Analysis of Women Safety in Indian Cities Using Machine Learning on Tweets”, 2019 Amity International Conference on Artificial Intelligence (AICAI), 2019, pp. 159-162, doi: 10.1109/AICAI.2019.8701247.
10. Vikram Chandra and Rampur Srinath. “Analysis of Women Safety using Machine Learning on Tweets”, (IRJET) 2020.
11. F. Bravo-Marquez, B. Pfahringer, S. Mohammad and E. Frank, “Affective Tweets: a Weka Package for Analysing effect in Tweets”, Journal of Machine Learning Research, vol. 20, no. 92, pp. 1-6, 2020.
12. K. Abdul Sattar, Q. Obeidat and M. Akure. “Towards harnessing based learning algorithms for tweets sentiment analysis international conference of innovation and intelligence for informatics Computing and technology 2020”.
13. K. R. Teja, K. A. Kumar, G. S. Praveen and D. N. Harini. “Analysis of Crimes Against Women in India Using Machine Learning Techniques”, In Communication Software and Networks 2021 (pp. 499-510). Springer, Singapore.
14. Srinivasan, S., P. Muthu Kannan, and R. Kumar. "A Machine Learning Approach to Design and Develop a BEACON Device for Women’s Safety." Recent Advances in Internet of Things and Machine Learning. Springer, Cham, 2022. 111-115.
15. Bonny, Afrin Jaman, et al. "Sentiment Analysis of User-Generated Reviews of Women Safety Mobile Applications." 2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT). IEEE, 2022.
16. Ashok, K., et al. "A Survey on Design and Application Approaches in Women-Safety Systems." 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS). Vol. 1. IEEE, 2022.
17. Tran, Martino, et al. "Monitoring the well-being of vulnerable transit riders using machine learning based sentiment analysis and social media: Lessons from COVID-19." Environment and Planning B: Urban Analytics and City Science (2022): 23998083221104489.
18. Zhong, Yongqi, et al. "Use of machine learning to estimate the per-protocol effect of low-dose aspirin on pregnancy outcomes: a secondary analysis of a randomized clinical trial." JAMA network open 5.3 (2022): e2143414-e2143414.
19. Patel, Bansi, and Manmitsinh C. Zala. "Crime Against Women Analysis & Prediction in India Using Supervised Regression." 2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT). IEEE, 2022.
20. Islam, Md M., et al. "Risk factors identification and prediction of anemia among women in Bangladesh using machine learning techniques." Current Women's Health Reviews 18.1 (2022): 118-133
21. [1] Agarwal, Apoorv, Fadi Biadsy, and Kathleen R. Mckeown. "Contextual phrase-level polarity analysis using lexical affect scoring and syntactic n-grams." Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2009.
22. [2] Barbosa, Luciano, and Junlan Feng. "Robust sentiment detection on twitter from biased and noisy data." Proceedings of the 23rd international conference on computational linguistics: posters. Association for Computational Linguistics, 2010.
23. [3] Bermingham, Adam, and Alan F. Smeaton. "Classifying sentiment in microblogs: is brevity an advantage?." Proceedings of the 19th ACM international conference on Information and knowledge management. ACM, 2010.
24. [4] Gamon, Michael. "Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis." Proceedings of the 20th international conference on Computational Linguistics. Association for Computational Linguistics, 2004.
25. [5] Kim, Soo-Min, and Eduard Hovy. "Determining the sentiment of opinions." Proceedings of the 20th international conference on Computational Linguistics. Association for Computational Linguistics, 2004.
26. [6] Klein, Dan, and Christopher D. Manning. "Accurate unlexicalized parsing." Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1. Association for Computational Linguistics, 2003.
27. [7] Charniak, Eugene, and Mark Johnson. "Coarse-to-fine n-best parsing and MaxEnt discriminative reranking." Proceedings of the 43rd annual meeting on association for computational linguistics. Association for Computational Linguistics, 2005.
28. [8] Gupta, B., Negi, M., Vishwakarma, K., Rawat, G., & Badhani, P. (2017). Study of Twitter sentiment analysis using machine learning algorithms on Python. International Journal of Computer Applications, 165(9), 0975-8887.
29. [9] Sahayak, V., Shete, V., & Pathan, A. (2015). Sentiment analysis on twitter data. International Journal of Innovative Research in Advanced Engineering (IJIRAE), 2(1), 178-183.
30. [10] Mamgain, N., Mehta, E., Mittal, A., & Bhatt, G. (2016, March). Sentiment analysis of top colleges in India using Twitter data. In Computational Techniques in Information and Communication Technologies (ICCTICT), 2016 International Conference on (pp. 525-530). IEEE.