# **A screenshot of a questionnaire AI-generated content may be incorrect.**

A white paper with black text

AI-generated content may be incorrect.

Table of Contents

[List of Figures 4](#_Toc188904517)

[List of Tables 5](#_Toc188904518)

[Task 01: Theory Exercise 6](#_Toc188904519)

[Introduction to Machine Learning 6](#_Toc188904520)

[1. K-Means Clustering 7](#_Toc188904521)

[**a.** **Working Principles of K-Means:** 7](#_Toc188904522)

[**b.** **Advantages of K-Means Clustering** 9](#_Toc188904523)

[**c.** **Disadvantages of K-Means Clustering** 9](#_Toc188904524)

[**d.** **Real World Applications** 10](#_Toc188904525)

[2. Performance Evaluation Metrics of Classification Algorithms 11](#_Toc188904526)

[**A.** **Confusion Matrix** 11](#_Toc188904527)

[**B.** **Accuracy** 12](#_Toc188904528)

[**C.** **Precision** 12](#_Toc188904529)

[**D.** **Recall** 12](#_Toc188904530)

[**E.** **F1-Score** 13](#_Toc188904531)

[**F.** **AUC ROC Curve** 13](#_Toc188904532)

[**G.** **Logarithmic Loss (Log Loss)** 14](#_Toc188904533)

[Task 02: Programming Exercise 15](#_Toc188904534)

[Hate Speech and Offensive Language Classification in Twitter Dataset 15](#_Toc188904535)

[**1.** **Data Understanding and Preprocessing** 15](#_Toc188904536)

[**2.** **Feature Engineering Process** 24](#_Toc188904537)

[**3.** **Model Selection and Training** 26](#_Toc188904538)

[**4.** **Hyperparameter Tuning** 30](#_Toc188904539)

[**5.** **Model Evaluation** 31](#_Toc188904540)

[**6.** **Model Deployment** 37](#_Toc188904541)

[**7.** **Conclusion and Recommendations** 41](#_Toc188904542)

[Bibliography 43](#_Toc188904543)

# **List of Figures**

[Figure 1: Figure Showing K-Means Clustering (JavaPoint, 2024) 7](#_Toc188900578)

[Figure 2: Steps of Working Principles of K-Means Clustering (Madushan, 2017) 8](#_Toc188900579)

[Figure 3: Email Spam Detection (Binary Classification Problem) (Zheng, 2015) 11](#_Toc188900580)

[Figure 4: Confusion Matrix (Kumar, 2024) 11](#_Toc188900581)

[Figure 5: ROC AUC (GeeksForGeeks, 2024) 14](#_Toc188900582)

[Figure 6: Importing Required Libraries 16](#_Toc188900583)

[Figure 7: Dataset Import and First 5 Rows of Dataset 16](#_Toc188900584)

[Figure 8: Information about Dataset Shape and Columns 17](#_Toc188900585)

[Figure 9: Dropping Unwanted Columns 17](#_Toc188900586)

[Figure 10: Dataset Information 18](#_Toc188900587)

[Figure 11: Checking Null Values 18](#_Toc188900588)

[Figure 12: Sample Tweet Text 18](#_Toc188900589)

[Figure 13: Code to Remove Unwanted Text and Symbols 19](#_Toc188900590)

[Figure 14: Class Distribution in Dataset 20](#_Toc188900591)

[Figure 15: Word Cloud of Class 2 (No Hate or Offensive Language) 21](#_Toc188900592)

[Figure 16: Word Cloud of Offensive Language Class 21](#_Toc188900593)

[Figure 17: Word Cloud of Hate Speech Class 22](#_Toc188900594)

[Figure 18: Text Length Distribution 22](#_Toc188900595)

[Figure 19: Top 10 Repeated Words in Hate Speech Class 23](#_Toc188900596)

[Figure 20: Top 10 Most Repeated Words in Offensive Language Corpus 24](#_Toc188900597)

[Figure 21: Top 10 Most Repeated Words in No Hate or Offensive Class Corpus 24](#_Toc188900598)

[Figure 22: Stop Word Removal and Stemming 24](#_Toc188900599)

[Figure 23: Code to Balance the Dataset 25](#_Toc188900600)

[Figure 24: Code to Implement TF-IDF Vectorizer 26](#_Toc188900601)

[Figure 25: Feature Scaling Using Standard Scaler 26](#_Toc188900602)

[Figure 26: Dimensionality Reduction using PCA 26](#_Toc188900603)

[Figure 27: Dataset Splitting 27](#_Toc188900604)

[Figure 28: Graph of Logistic Regression (Tech-AI-Math, 2023) 27](#_Toc188900605)

[Figure 29: Implementing Logistic Regression 27](#_Toc188900606)

[Figure 30: Prediction and Accuracy of Logistic Regression 28](#_Toc188900607)

[Figure 31: Working of Random Forest Classifier (Khushaktov, 2023) 28](#_Toc188900608)

[Figure 32: Implementing Random Forest Classifier 28](#_Toc188900609)

[Figure 33: Prediction and Accuracy Score of Random Forest Classifier 29](#_Toc188900610)

[Figure 34: Graph Showing SVM (Java Tpoint, 2024) 29](#_Toc188900611)

[Figure 35: Implementing SVC 29](#_Toc188900612)

[Figure 36: Prediction and Accuracy of SVM 30](#_Toc188900613)

[Figure 37: Tuning Hyperparameter of Logistic Regression 30](#_Toc188900614)

[Figure 38: Tuning Hyperparameter of Random Forest 31](#_Toc188900615)

[Figure 39: Hyperparameter Tuning of SVC (SVM) 31](#_Toc188900616)

[Figure 40: Classification Report of Logistic Regression 32](#_Toc188900617)

[Figure 41: Confusion Matrix Heatmap of Logistic Regression 32](#_Toc188900618)

[Figure 42: AUC-ROC Curve for Logistic Regression 33](#_Toc188900619)

[Figure 43: Classification Report of Random Forest Classifier 34](#_Toc188900620)

[Figure 44: Confusion Matrix Heatmap of Random Forest Classifier 34](#_Toc188900621)

[Figure 45: AUC-ROC Curve for Random Forest Classifier 35](#_Toc188900622)

[Figure 46: Classification Report of SVM 35](#_Toc188900623)

[Figure 47: Confusion Matrix Heatmap of SVM 36](#_Toc188900624)

[Figure 48: AUC-ROC Curve for SVM 37](#_Toc188900625)

[Figure 49: Code Showing Preprocessing and Feature Engineering in Model Deployment 38](#_Toc188900626)

[Figure 50: Implementation of Feature Engineering and Prediction of Real-world Input Data 39](#_Toc188900627)

[Figure 51: Interface of Streamlit Web App 39](#_Toc188900628)

[Figure 52: Test Prediction One 40](#_Toc188900629)

[Figure 53: Test Prediction Two 40](#_Toc188900630)

[Figure 54: Test Prediction Three 41](#_Toc188900631)

# **List of Tables**

[Table 1: Column Description 17](#_Toc188896159)

[Table 2: Sample Tweets Before and After Removal of Unwanted Texts and Symbols 19](#_Toc188896160)

[Table 3: Overall Performance Comparison of Different Algorithms 37](#_Toc188896161)

# **Task 01: Theory Exercise**

## **Introduction to Machine Learning**

A paper with text on it

AI-generated content may be incorrect.

## **K-Means Clustering**

K-Means Clustering is an unsupervised machine learning algorithm widely used for partitioning datasets into distinct groups (clusters) based on their similarities. This method is valuable for covering patterns and relationships within data without relying on labelled datasets or examples (GeeksforGeeks, 2025). K-Means is a distance-based or centroid-based algorithm that assigns data points to clusters by calculating distances. Each cluster in K-Means is represented by a centroid. It minimizes the variance within each cluster while maximizing the variance between clusters. Optimization is a key aspect of the K-Means clustering algorithm. Its objective is to identify the optimal set of centroids that minimizes the total squared distances between data points and their nearest centroid (Sharma, 2025). The K-Means Clustering algorithm primarily involves two key steps:

* Iteratively determining the optimal positions for K centroids (center points)
* Assigning each data point to the nearest centroid, with points close to a particular centroid forming a cluster.

A diagram of a diagram of a number of dots

AI-generated content may be incorrect.

Figure 1: Figure Showing K-Means Clustering (JavaPoint, 2024)

### **Working Principles of K-Means:**

We have the purpose of categorizing items based on specified attributes and their related values (expressed as vectors). To do this, we use the K-Means technique, with 'K' denoting the number of groups or clusters into which we want to classify the items. The algorithm divides the objects into K groups or clusters according to their similarity. To measure this similarity, the Euclidean distance is used as the metric (Rinkal, 2023). The working principles of K-means follow the following steps:

1. **Initialization**

The initial step involves selecting the number of clusters, K, which is a predefined parameter. Randomly chosen K cluster centroids are then assigned to the dataset, acting as the starting points for each cluster.

1. **Assignment Step**

For each data point in the dataset, the distance to each of the K centroids is calculated, and the point is assigned to the cluster with the closest centroid, which is typically determined using Euclidean distance. This mechanism leads to the creation of K clusters.

1. **Update Step**

After assigning all data points to clusters, the centroids are updated by calculating the average of all data points inside each cluster.

1. **Iteration**

The assignment and update procedures are performed iteratively until the centroids stabilize (i.e., their positions don't fluctuate considerably) or the maximum number of iterations is reached.

1. **Convergence**

The algorithm ends when the assignments remain unaltered or a predetermined number of iterations are reached. After convergence, it generates the final cluster centroids and allocates each data point to the appropriate cluster.

The figure below shows the working principles steps of K-means described above:

A diagram of a step

AI-generated content may be incorrect.

Figure 2: Steps of Working Principles of K-Means Clustering (Madushan, 2017)

### **Advantages of K-Means Clustering**

1. **Simplicity and Efficiency:**

One of the primary benefits of the K-Means algorithm is its simplicity. Its straightforward approach of splitting data into clusters based on similarity makes it easy to grasp and apply (Sharma, 2025). Additionally, K-Means is computationally efficient, making it well-suited for processing large datasets. Its low complexity enables quick and effective data processing.

1. **Scalability and Flexibility:**

The algorithm's efficiency scales effectively with increasing data points, making K-means suitable for small to large datasets using variations like Mini-batch K-means. This algorithm works well for various types of clustering problems and provides intuitive results.

1. **Fast Convergence:**

The data point converges relatively quickly compared to some other clustering algorithms.

1. **Customizable Number of Clusters:**

The number of clusters, k, is configurable based on the dataset. Although choosing the best k can be difficult, tools like the elbow method and silhouette score help data analysts make well-informed decisions on the optimum number of clusters.

1. **Interpretable Results:**

K-Means generates clusters that are straightforward to interpret, with each cluster representing a distinct group of data points sharing similar characteristics. This helps in extracting meaningful insights from the results.

### **Disadvantages of K-Means Clustering**

1. **Dependence on Number of Clusters**

The performance of K-Means is heavily reliant on selecting the optimum number of clusters (k). An erroneous selection of k might lead to clusters that lack meaning or fail to deliver significant insights.

1. **Initialization Dependency**

K-Means convergence is influenced by the initial location of cluster centroids, as different starting sites might result in different end cluster arrangements. To solve this, strategies like the K-Means++ initialization method are utilized to increase clustering stability and quality.

1. **Assumption of Equal-sized Clusters and Spherical Shapes**

K-Means assumes clusters are equal in size and spherical in shape. This assumption can result in suboptimal clustering outcomes when working with datasets containing clusters of varying sizes or non-spherical shapes.

1. **Sensitive to Outliers**

K-Means is extremely sensitive to outliers, which can distort the placement of cluster centroids and significantly affect the overall clustering findings.

1. **Not Suitable for Non-Linear Data**

K-Means assumes that clusters are separated by linear boundaries, which can limit its effectiveness on datasets with complex or non-linear cluster structures.

### **Real World Applications**

1. **Customer Segmentation**

* Businesses use K-means to categorize clients based on purchase habits, demographics, or preferences.
* For example, a retailer segments its customer base into “budget-conscious,” “frequent buyers,” and “premium shoppers.”

1. **Image Compression**

* K-means reduces image colour count by clustering related hues.
* Example: Compressing an image from millions of colour to just a few dominant ones.

1. **Anomaly Detection**

* Identify anomalies in datasets, such as fraudulent transactions or network intrusions.
* Example: Fraud detection in banking transactions.

1. **Document Clustering**

* Used in text mining to group similar documents based on content or keywords.
* Example: Categorizing news articles into topics such as sports, politics, and technology.

1. **Genomics**

* Cluster genes with similar expression patterns for understanding biological processes or identifying diseases.
* Example: Cluster DNA sequences to understand genetic similarities.

1. **Market Basket Analysis**

* Identifies groups of frequently bought products to improve recommendation systems.
* **Example**: Cluster products in a supermarket into groups like "snacks," "beverages," and "dairy."

K-means clustering is a sophisticated and efficient algorithm for grouping data into meaningful categories. While it is simple to create and versatile, users must overcome problems such as determining the optimal number of clusters, dealing with outliers, and assuring proper initialization. Despite its shortcomings, K-means remains a cornerstone in machine learning and data analysis because of its broad applicability, which ranges from customer segmentation to biological data analysis and urban planning.

## **Performance Evaluation Metrics of Classification Algorithms**

The classification algorithm is a supervised learning technique used in machine learning and statistics to predict the category or class of an input data point (Keita, 2024). The goal of these algorithms is to map input data to one of the predefined discrete labels or categories based on the labeled training data. There are three types of classification problems: Binary Classification - predicting whether an email is spam or not (Spam/Not Spam), Multi-Class Classification - Identifying the type of fruit in an image (Apple, Banana, Orange, etc.), and Multi-Label Classification - Predicting multiple labels for one input (e.g., tagging an image with multiple objects like "Dog" and "Car").

A question mark above a mail

AI-generated content may be incorrect.

Figure 3: Email Spam Detection (Binary Classification Problem) (Zheng, 2015)

Performance evaluation metrics are strategies and tools for assessing the efficacy of a machine learning model. They help to make educated judgments about model selection and optimization by providing insights about the model's performance (Melanie, 2023). For classification algorithms, different types of performance evaluation metrics are described below:

### **Confusion Matrix**

A confusion matrix is a table that assesses the performance of a classification model. It summarizes the prediction findings by comparing the actual labels to the expected labels (Kumar, 2024). This matrix helps in identifying how well the model is performing and which types of errors it is making.

**A diagram of positive values

AI-generated content may be incorrect.**

Figure 4: Confusion Matrix (Kumar, 2024)

The confusion matrix shows the number of:

* **True Positive (TP):** Correctly predicted positive cases, which means we predicted positive and it’s true.
* **True Negative (TN):** Correctly predicted negative cases. It means we predicted negative and it’s true.
* **False Positive (FP):** Incorrectly predicted as positive (Type I error). We predicted positive and it is false.
* **False Negative (FN):** Incorrectly predicted as negative (Type II error). We predicted negative and it’s false.

It provides the foundation for most other evaluation metrics like precision, recall, F1-score, etc.

### **Accuracy**

Accuracy represents how frequently a classifier makes correct predictions. It is calculated as the proportion of correct predictions out of the total predictions made. It is a fundamental metric for assessing the performance of a classification model, offering a simple overview of how effectively the model makes correct predictions. This is calculated as:

Accuracy is a simple and intuitive metric but can be misleading for imbalanced datasets (e.g. when one class dominates the dataset). While a model achieving 99% accuracy may appear to perform exceptionally well, this metric can be deceptive and fail to reflect true performance in certain scenarios. So, we can use accuracy as a metric when the dataset is well-balanced and not skewed much.

### **Precision**

Precision measures the fraction of successfully predicted positive cases among all positive predictions produced by the model. It assesses how accurate the model's positive predictions are. Precision is calculated as the number of true positives divided by the sum of true and false positives.

Precision is especially important in scenarios where False Positives pose a greater risk than False Negatives. It plays a critical role in applications like music or video recommendation systems and e-commerce platforms, where inaccurate recommendations can lead to a poor user experience, loss of trust, customer churn, and potential harm to the business.

### **Recall**

Recall is the fraction of accurately predicted positive observations to total positive observations. It is calculated using the following formula:

Recall is a valuable metric in situations where **FNs** are more critical than **FPs**. This is especially significant in medical contexts, where overlooking true positive instances may have major effects, whereas raising a false alert is less of a problem (e.g., detecting fraud or cancer).

### **F1-Score**

The F1-score is the harmonic mean of precision and recall, offering a balanced measure that takes both metrics equally. Its range is [0, 1]. This metric indicates how precise our classifier is (correctly classifying a significant number of instances) while also being robust by minimizing the number of missed instances. It is calculated by:

The F1 score penalizes extreme values more heavily. It serves as an effective evaluation metric in the following situations:

* When FP and FN carry equal costs.
* When adding more data does not significantly impact the outcome
* When the number of True Negatives is high.

The F1-score is useful when you need to balance precision and recall, especially in circumstances of class imbalance.

### **AUC ROC Curve**

This is one of the most commonly used metrics, primarily for binary classification. The AUC (Area Under the Curve) indicates the probability that the classifier would rank a randomly chosen positive example higher than a randomly chosen negative example. It is a probability curve that compares the True Positive Rate (TPR) to the False Positive Rate (FPR) at various threshold values to help separate the' signal' from the 'noise'. Typically, the false positive rate is computed as:

The true positive rate is computed as:

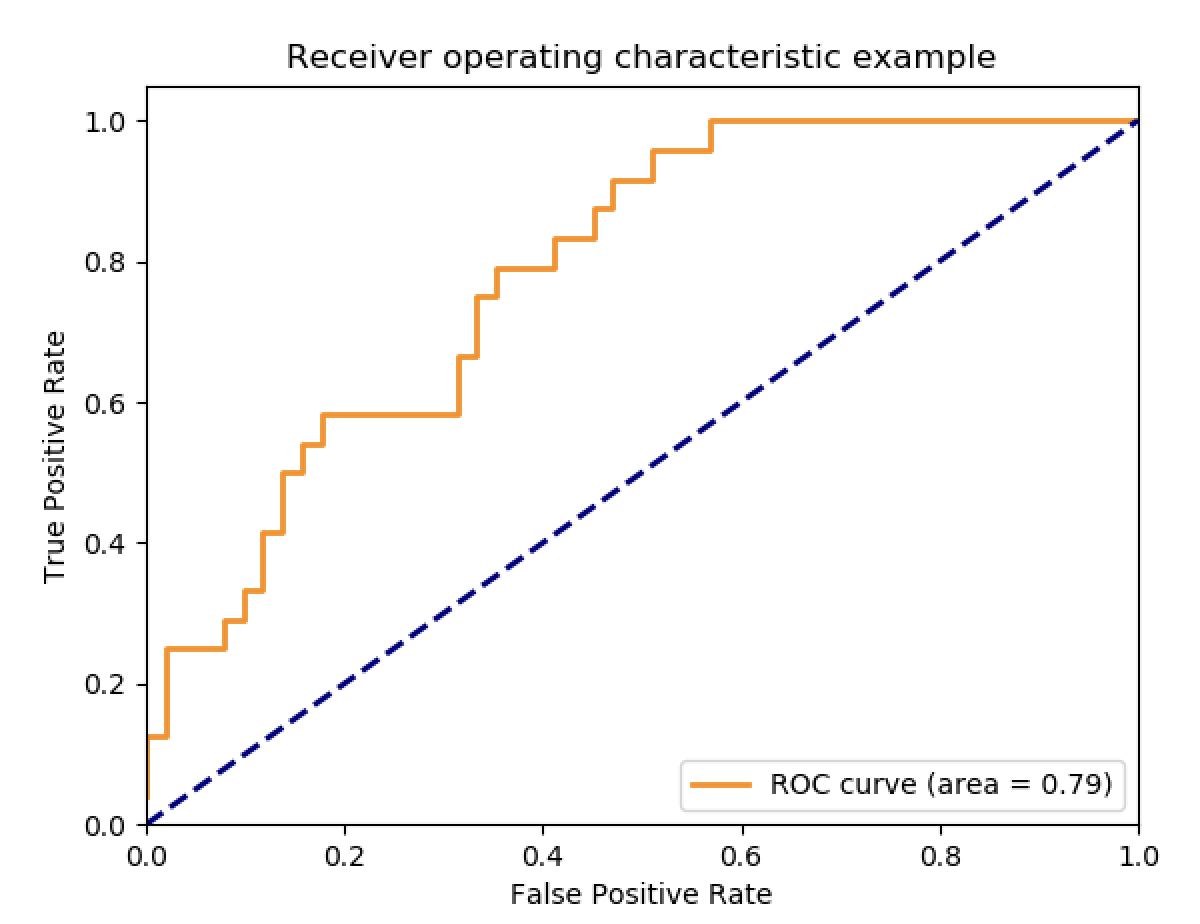


Figure 5: ROC AUC (GeeksForGeeks, 2024)

From the graph above, a higher AUC indicates better model performance across various threshold points for distinguishing between positive and negative classes. Specifically:

* **AUC = 1**: The classifier completely isolates all positive and negative class points.
* **AUC = 0**: The classifier predicts that all negatives are positive, and all positives are negative.
* **AUC = 0.5**: The classifier performs no better than random guessing, demonstrating no capacity to distinguish between positive and negative classes.

AUC-ROC is threshold-independent and effective for evaluating the overall performance of a binary classifier. It is used when the data is not balanced. It can also be used to compare various classification algorithms.

### **Logarithmic Loss (Log Loss)**

This metric, also referred to as Log Loss, operates by penalizing incorrect classifications, particularly **False Positives**. It is commonly used in **multi-class classification** problems. In Log Loss, the classifier assigns a probability to every class for each sample, ensuring a probabilistic approach to predictions. Log loss evaluates the prediction probabilities instead of the hard classifications. It measures the uncertainty of predictions:

Where yi is the true label and pi is the predicted probability for the positive class. This formula penalizes predictions that deviate from the true label, with larger penalties for probabilities further from the correct class. Log loss is beneficial when you need to evaluate probabilistic models.

# **Task 02: Programming Exercise**

## **Hate Speech and Offensive Language Classification in Twitter Dataset**

In the age of digital communication, the proliferation of hate speech and offensive language has become a pressing concern. Online platforms such as social media, forums, and messaging apps are often exploited to spread harmful content that can incite violence, foster discrimination, or create a toxic environment. Effectively identifying and addressing such content is critical to maintaining safe and inclusive online spaces. This is where machine learning emerges as a powerful solution.

Machine learning approaches use Natural Language Processing (NLP) techniques to detect and classify hate speech and abusive words in text. By evaluating massive amounts of labeled data, machine learning models can catch linguistic patterns, recognize contextual nuances, and adjust to language evolution. This project intends to construct a robust system for hate speech and objectionable language classification, utilizing machine learning approaches..

For this task, the dataset was taken from the Kaggle website (link - [https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive- language-dataset](https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-%20language-dataset)) which was taken from Twitter that is built to support research and development in detecting and analyzing hate speech and offensive language on social media, distinguishing them from ordinary slang and neutral content. Using this dataset, we trained and built our model for hate speech and offensive language classification.

### **Data Understanding and Preprocessing**

Data understanding refers to gathering insights about the dataset, such as its dimensions, columns, structure, and formats, to prepare for model development and analysis. Converting raw data into a comprehensible and usable format is known as data preprocessing (Joby, 2021). The key steps in data understanding and preprocessing include:

#### **Importing Libraries**

The required Python libraries are imported first, as we are using Python for this project as the programming language.

* 1. **Numpy and Pandas:** These are fundamental Python libraries essential for addressing any machine learning problem. The Pandas library focuses on data importing and manipulation, whereas NumPy, short for Numerical Python, is primarily used for creating and working with arrays and matrices in machine learning models.
  2. **Matplotlib and Seaborn:** These libraries are used for data visualization.
  3. **Scikit-learn:** It is a Python-based, open-source machine learning library. It offers a variety of functions and methods for tasks such as model selection, data imputation, preprocessing, feature selection, and more.
  4. **NLTK:** The Natural Language Toolkit (NLTK) is a prominent open-source Python package used for natural language processing (NLP) activities. It offers numerous tools and resources for text analysis, including tokenization, stemming, lemmatization, part-of-speech tagging, parsing, etc.

**A screenshot of a computer program

AI-generated content may be incorrect.**

Figure 6: Importing Required Libraries

#### **Importing Dataset**

The dataset is imported as a Pandas DataFrame, enabling us to work with the data in tabular format.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 7: Dataset Import and First 5 Rows of Dataset

#### **Data Understanding**

The original dataset contains 24783 rows of data and 6 columns. The figure below shows the code to get the shape and column information of the dataset.

A screen shot of a computer

AI-generated content may be incorrect.

Figure 8: Information about Dataset Shape and Columns

The column description is provided in the table below:

Table 1: Column Description

A screenshot of a table

AI-generated content may be incorrect.

But for our hate speech and offensive language classification model, we just need two columns – tweet and class. So, we dropped all other columns like count, hate\_speech, offensive\_language and neither.

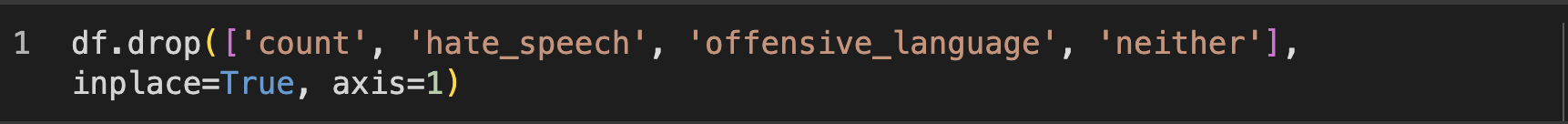


Figure 9: Dropping Unwanted Columns

The **df.info()** method provides a summary of the dataset, including details about columns, non-null value counts, data types, and memory usage. The figure includes all the columns before dropping unwanted columns.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 10: Dataset Information

We don’t have any null values in our dataset, which can be seen in the figure below:

A black screen with a black border

AI-generated content may be incorrect.

Figure 11: Checking Null Values

#### **Data Cleaning**

As we can see in the tweet column, there are a lot of unwanted texts and symbols which introduce noise in the data.

A screenshot of a phone

AI-generated content may be incorrect.

Figure 12: Sample Tweet Text

This can confuse the model, leading it to learn patterns that are not relevant to the task. Also, many text classification models rely on converting text into numerical features, so unwanted symbols can dilute meaning information and increase sparsity. The unwanted text and symbols also can cause the model to mistakenly learn false associations between text features. So, these text and symbols should be removed to make it clean data. We have used Python’s regular expression library to remove these text and symbols, which is shown in the figure below:

A computer screen shot of text

AI-generated content may be incorrect.

Figure 13: Code to Remove Unwanted Text and Symbols

The above figure contains the code to remove the retweet indicator ‘RT’, URLs, Twitter handles sign, HTML tags, non-alphabetic characters, etc., using regular expressions. The table below presents the tweets before and after the removal of unwanted text and symbols.

Table 2: Sample Tweets Before and After Removal of Unwanted Texts and Symbols

A screenshot of a social media post

AI-generated content may be incorrect.

#### **Exploratory Data Analysis**

Exploratory Data Analysis (EDA) is the process of analyzing and summarizing datasets to identify patterns, detect anomalies, test hypotheses, and verify assumptions through statistical and visual techniques. It is one of the initial and most critical steps in any data analysis or machine learning workflow, providing insights to guide further data preprocessing, feature selection, or modelling. We have performed some EDA on the dataset, taking the tweet and class columns.

1. **Class Distribution of Tweets**

We have around 24783 tweet data, and these data are classified into three labels in class – hate speech, offensive language, and neither, which means no hate or offensive. So, the bar chart below visualizes the class distribution in a dataset containing these three categories. The "Offensive Language" class dominates the dataset with nearly 20,000 samples, while the "Neither" class contains significantly fewer samples, and the "Hate Speech" class has the smallest number of samples. This imbalance suggests potential challenges in training a balanced model and highlights the need for strategies to address the class imbalance, such as data augmentation or re-sampling techniques.

A graph with different colored squares

AI-generated content may be incorrect.

Figure 14: Class Distribution in Dataset

1. **Word Cloud of Different Class**

A **word cloud** is a visual representation of text data where the size of each word reflects its importance, frequency, or relevance in the dataset. Word clouds are often used for **text analysis**, **data visualization**, or to provide a quick summary of the most prominent terms in a document or corpus.

**A close up of words

AI-generated content may be incorrect.**

Figure 15: Word Cloud of Class 2 (No Hate or Offensive Language)

The above word cloud illustrates the most frequently occurring words in the "Neither" category of a dataset, with larger words representing higher frequencies. Words like "trash," "bird," "Yankee," and "yellow" appear prominently, indicating they are used often in this context.

A close up of words

AI-generated content may be incorrect.

Figure 16: Word Cloud of Offensive Language Class

A close up of words

AI-generated content may be incorrect.

Figure 17: Word Cloud of Hate Speech Class

1. **Text Length Distribution**

**A graph of a number of columns

AI-generated content may be incorrect.**

Figure 18: Text Length Distribution

The above histogram depicts text length distribution. It shows a right-skewed distribution, with most of the text lengths concentrated around the 20–60 range.

1. **Top 10 Word Frequency in Different Class**

The bar charts below show each class category’s top 10 repeated words.

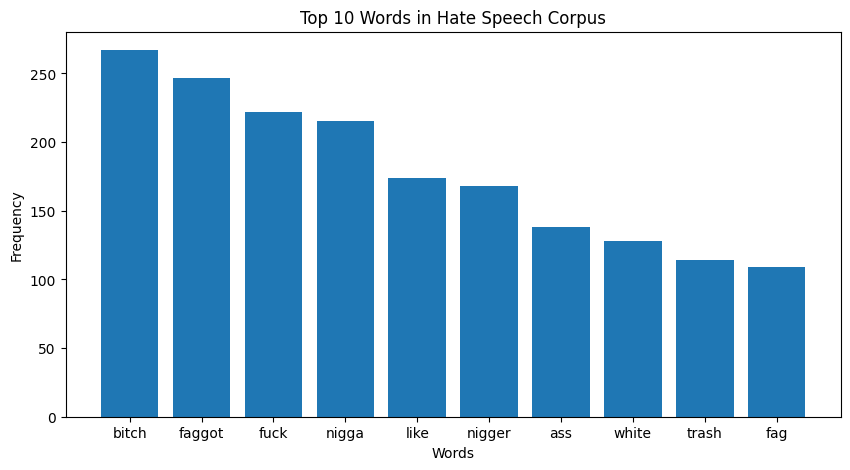


Figure 19: Top 10 Repeated Words in Hate Speech Class

This bar chart shows the top 10 most frequent words in the "Hate Speech" category of a dataset. The words are ranked by their frequency, with "bitch" being the most common, followed by "faggot," "fuck," and "nigga." This distribution highlights the offensive and derogatory nature of language within this category, underscoring the severity of the content in the dataset.

The bar chart below shows the top 10 most repeated words in the offensive class of the dataset. In this class also, “bitch” is the most common word, followed by “hoe”, “like”, “pussi”, “fuck”, etc.

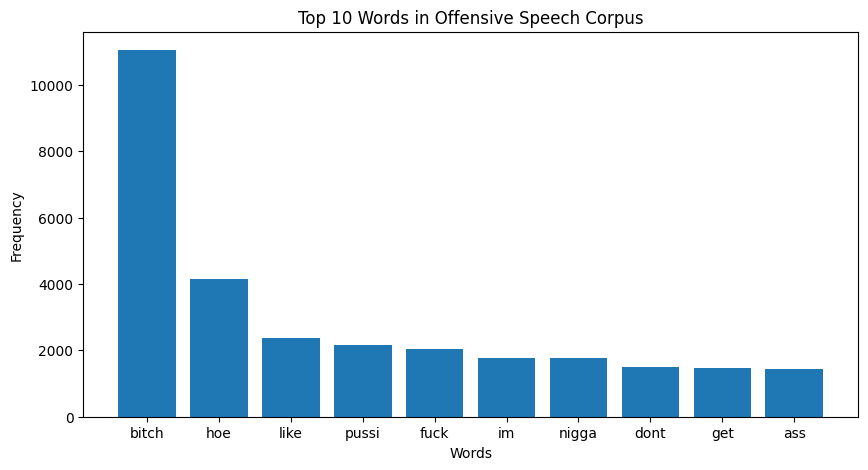


Figure 20: Top 10 Most Repeated Words in Offensive Language Corpus

A graph of blue rectangular bars with white text

AI-generated content may be incorrect.

Figure 21: Top 10 Most Frequent Words in No Hate or Offensive Class Corpus

The bar chart above shows the top 10 most frequent words in the dataset with no hate or offensive language. In this class, “trash” is the most common word, followed by “bird”, “like”, “yanke”, “charli”, “im”, etc.

### **Feature Engineering Process**

This is the crucial step in the machine learning pipeline. In this step, raw data is transformed into meaningful features that can enhance model performance. We balance the dataset, extract features from textual data, scale the features, and reduce their dimensionality.

#### **Stop Words Removal and Stemming**

**A black background with white text

AI-generated content may be incorrect.**

Figure 22: Stop Word Removal and Stemming

Stop words are the commonly used words in a language that are often removed during text preprocessing in NLP tasks which are not critical for understanding the main content of the text. For eg. “a”, “an”, “the”, “is” etc. We are not removing “not” but it is also in the list of stop words as it has a crucial meaning in our model.

Stemming is the process of reducing a word to its **root form** or **stem**, which may not necessarily be a valid word. The purpose of stemming is to group different forms of a word so they can be analyzed as a single item in natural language processing (NLP) tasks. We are using Porter Stemmer in our project.

1. **Dataset Balancing**

As from the above EDA (Figure 14), our dataset is imbalanced as the offensive language class has a large number of data (19190), and the hate speech class has a low number of data (1430). This will lead our model to have biased predictions, misleading evaluation metrics, poor learning from minority classes and overfitting in minority classes.

So, to balance the dataset, we resample the dataset into the number of hate speech data i.e. 1430 for every class.

A computer screen shot of a program

AI-generated content may be incorrect.

Figure 23: Code to Balance the Dataset

#### **Feature Extraction using TF-IDF Vectorizer**

The **TF-IDF Vectorizer** (Term Frequency-Inverse Document Frequency) is a popular technique in Natural Language Processing (NLP) used to convert text data into numerical representations for machine learning models. It assigns a weight to each word in a document based on its importance, calculated as the product of term frequency (how often the word appears in a document) and inverse document frequency (how unique the word is across all documents). This helps emphasize important words while down-weighting common ones, making it ideal for text classification, sentiment analysis, and information retrieval tasks.

The figure below shows the code to implement the TF-IDF vectorizer for our project which will convert our text data into numerical representations. We have limited the vectorizer to use only the top 5,000 features (or terms) based on their importance (e.g., frequency or TF-IDF score).

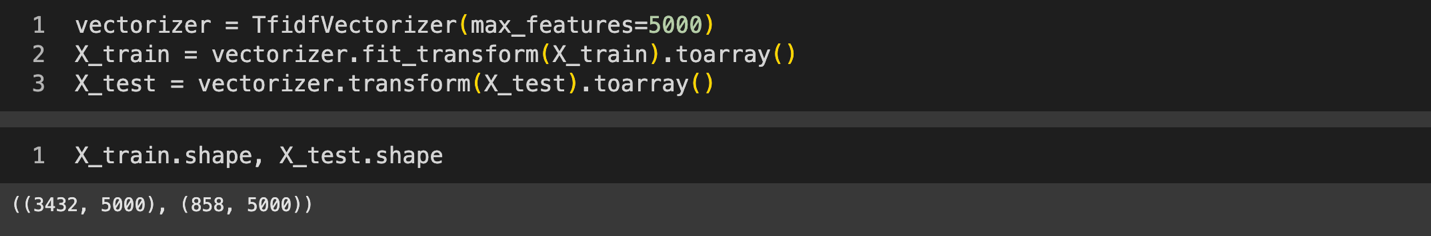


Figure 24: Code to Implement TF-IDF Vectorizer

#### **Feature Scaling**

After the feature extraction, the features are scaled using Standard Scalar, which scales the features in such a way that it has a value of mean as zero and a standard deviation of one which is based on the below formula:

Where X = Original value of the feature

µ = Mean

𝜎 = Standard Deviation



Figure 25: Feature Scaling Using Standard Scaler

#### **Dimensionality Reduction using PCA**

Dimensionality reduction with PCA (Principal Component Analysis) is a technique for lowering the number of features in a dataset while preserving as much variance as possible. PCA converts the data into a new collection of orthogonal features, known as principal components, that are linear combinations of the original features. The following graphic depicts the code for achieving dimensionality reduction using PCA.

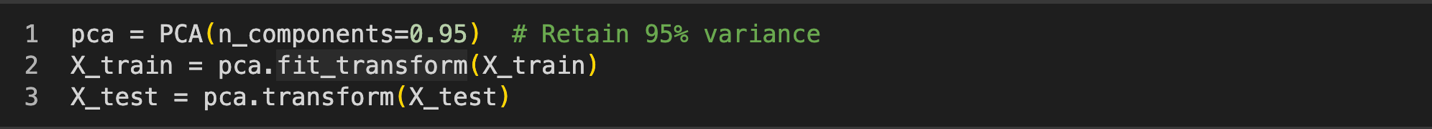


Figure 26: Dimensionality Reduction using PCA

Instead of specifying a fixed number of components, we choose PCA to choose as many components as needed to preserve **95% of the total variance** in the data. By retaining 95% of the variance, we reduce the feature space without significant loss of information.

### **Model Selection and Training**

After the feature engineering process, the data can be used to train and test our machine learning models. But for our project, we split our dataset first before feature engineering because this prevents data leakage, unbiased model predictions, mimics real-world scenarios and avoids overfitting in test data. We split our 80% dataset into a train set used to train the model and the remaining 20 % data as a test set used to test the built model.

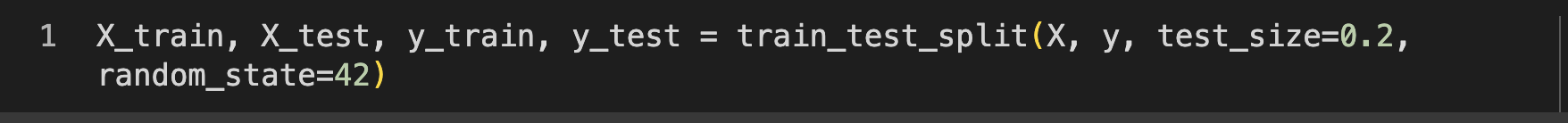


Figure 27: Dataset Splitting

The dataset was divided into two parts: a training set and a test set, and several machine learning methods were used to create models for hate speech and offensive language classification. Three algorithms were utilized for this purpose: Logistic regression, random forest, and support vector machines.

#### **Logistic Regression**

Logistic regression is commonly used for classification problems in machine learning. It analyzes historical data to predict binary outcomes, such as yes or no, or to classify data into distinct categories. This technique determines the dependent variable by examining its relationship with one or more independent variables, making it a go-to solution for binary classification tasks. The logistic function is expressed as:

A diagram of a curve

AI-generated content may be incorrect.

Figure 28: Graph of Logistic Regression (Tech-AI-Math, 2023)

The figure below shows the code to implement Logistic Regression with the Scikit Learn Python library. We called the Logistic Regression model and trained the model with a training set of data.

A black rectangular object with a black stripe

AI-generated content may be incorrect.

Figure 29: Implementing Logistic Regression

The prediction and accuracy score can be checked with the following code where the accuracy of logistic regression before hyperparameter tuning is 81.7%.

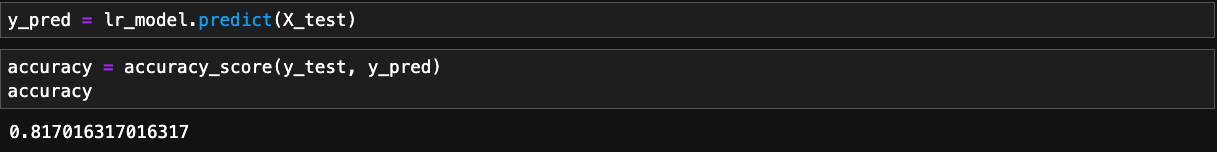


Figure 30: Prediction and Accuracy of Logistic Regression

#### **Random Forest**

Random Forest is a robust and adaptable ensemble machine learning technique that is mostly utilized for classification and regression applications. It works by creating many decision trees during training and integrating their predictions to provide a more accurate and consistent result. Each tree is trained on a random subset of the data using a technique known as bagging. At each decision split, a random subset of characteristics is considered, which helps to reduce overfitting and improve generalization. In regression, the final output is the average of predictions, whereas in classification, it is chosen by majority voting among all trees.

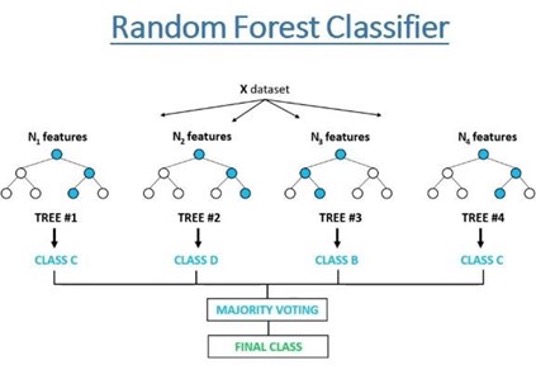


Figure 31: Working of Random Forest Classifier (Khushaktov, 2023)

The code to implement Random Forest, prediction and accuracy score is shown in the figure below. The accuracy score of the Random Forest Classifier for our project is 73.54%. This is the accuracy score before the hyperparameter tuning. We will tune the hyperparameter in the next step.

A black and yellow text

AI-generated content may be incorrect.

Figure 32: Implementing Random Forest Classifier

A black and white striped background

AI-generated content may be incorrect.

Figure 33: Prediction and Accuracy Score of Random Forest Classifier

#### **Support Vector Machine**

Support Vector Machine (SVM) is a supervised machine learning technique that is widely used for classification and regression applications, with a particular emphasis on classification problems. The core idea of SVM is to find a hyperplane that best separates data points from distinct classes in a high-dimensional environment. SVM aims to maximize the margin, which is the distance between the hyperplane and the nearest data points in each class, also known as support vectors. These support vectors are necessary for defining the decision boundary. When data is not linearly separable, SVM uses the kernel trick to translate it to a higher-dimensional space with a linear boundary. Linear, polynomial, and radial basis functions are among the most widely used kernel functions.

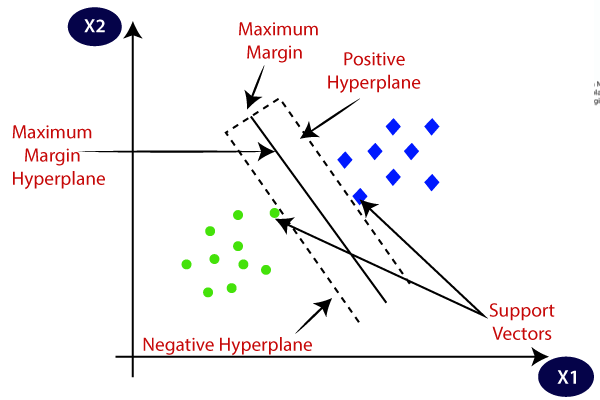


Figure 34: Graph Showing SVM (Java Tpoint, 2024)

The figure below shows the code to implement the support vector classifier, prediction and accuracy score of the algorithm.

A black and grey background with blue text

AI-generated content may be incorrect.

Figure 35: Implementing SVC

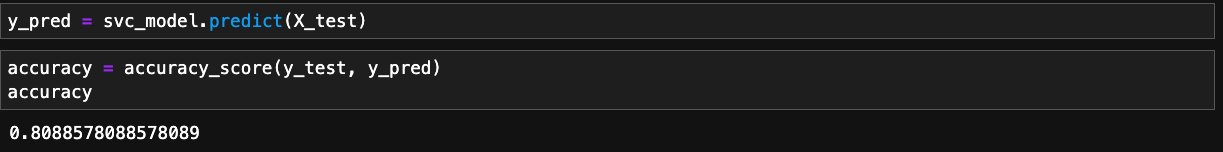
****

Figure 36: Prediction and Accuracy of SVM

The accuracy score of the SVM for our hate speech model is 80.8%.

### **Hyperparameter Tuning**

Hyperparameter tuning involves providing specific parameters as arguments to the constructor of an estimator class. **Hyperparameter tuning** is essential in machine learning because it helps optimize the performance of a model by finding the best combination of hyperparameters (Pandian, 2024). For hyperparameter tuning, we used Grid Search CV, which evaluates all possible combinations of specified hyperparameter values and selects the one that results in the best model performance, as determined by a specified scoring metric. We have used the scoring metric “accuracy” and cross-validation of 5.

#### **Tuning Hyperparameter of Logistic Regression**

The below figure shows the code to tune the hyperparameter of Logistic Regression.

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 37: Tuning Hyperparameter of Logistic Regression

Here, C is an inverse of regularization strength, and max\_iter is the Maximum number of iterations for solvers to converge. By using the parameter grid as above, we tune the hyperparameter of Logistic Regression and train the model.

#### **Tuning Hyperparameter of Random Forest**

From the above figure, the parameter grid has n\_estimators which denotes the number of trees in the forest, max\_depth which has the values for the maximum depth of each tree and min\_samples\_leaf which denotes the minimum number of samples required to split an internal node.

A computer screen shot of a program

AI-generated content may be incorrect.

Figure 38: Tuning Hyperparameter of Random Forest

#### **Tuning Hyperparameter of SVM**

The parameter **C** is a regularization factor that balances the trade-off between minimizing training error and maximizing the margin. **Kernel** specifies the type of hyperplane used to separate the data, while **gamma** determines the impact of each individual training example. The code for implementing hyperparameter tuning of SVM is shown in the figure below:

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

Figure 39: Hyperparameter Tuning of SVC (SVM)

### **Model Evaluation**

Model evaluation in machine learning is important to determine whether the model is accurate and reliable and helps to generalize unseen data. We evaluate our different machine learning models with different performance evaluation metrics like prediction accuracy, classification reports like precision, recall and F1 score, confusion matrix, AUC-ROC curve and Logarithmic Loss.

#### **Performance Evaluation of Logistic Regression**

From the classification report, we have a detailed report regarding prediction accuracy, precision, recall, and F1-score. Our Logistic Regression model has a **prediction accuracy** of **82%**. The overall precision, recall and F1 score of Logistic Regression are:

1. **Prediction Accuracy:** 82%
2. **Precision:** 0.81
3. **Recall:** 0.81
4. **F1 Score:** 0.81

The detailed classification report is presented in the figure below, where information about the precision, recall and f1-score of individual classes is shown:

A screenshot of a computer

AI-generated content may be incorrect.

Figure 40: Classification Report of Logistic Regression

1. **Confusion Matrix:**

The figure below is the confusion matrix heatmap of the Logistic Regression model, which shows the TP, TN, FP and FN values predicted by the model.

A blue squares with white text

AI-generated content may be incorrect.

Figure 41: Confusion Matrix Heatmap of Logistic Regression

According to the heatmap,

A white background with black text

AI-generated content may be incorrect.

1. **AUC – ROC Curve:**

The areas under the curve (AUC) scores are high, reflecting strong predictive performance: 0.87 for Hate Speech, 0.90 for Offensive Language, and 0.97 for Neutral (Neither).

A graph of a line graph

AI-generated content may be incorrect.

Figure 42: AUC-ROC Curve for Logistic Regression

1. **Logarithmic Loss:**

The logarithmic loss for our Logistic Regression model is **0.567.**

#### **Performance Evaluation of Random Forest Classifier**

From the classification report, we got a detailed report about prediction accuracy, precision, recall, and F1-score. Our Random Forest Classifier model has a **prediction accuracy** of **77%**. The overall precision, recall and F1 score of the Random Forest Classifier are:

* + 1. **Prediction Accuracy:** 77%
    2. **Precision:** 0.76
    3. **Recall:** 0.76
    4. **F1 Score:** 0.76

The detailed classification report is presented in the figure below, where information about the precision, recall and f1-score of individual classes is shown for the Random Forest Classifier:

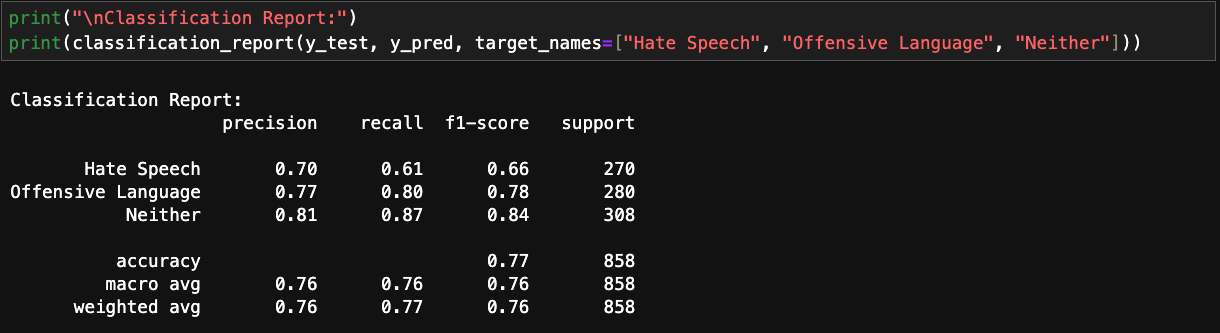


Figure 43: Classification Report of Random Forest Classifier

* + 1. **Confusion Matrix:**

The figure below is the confusion matrix heatmap of the Random Forest Classifier model.

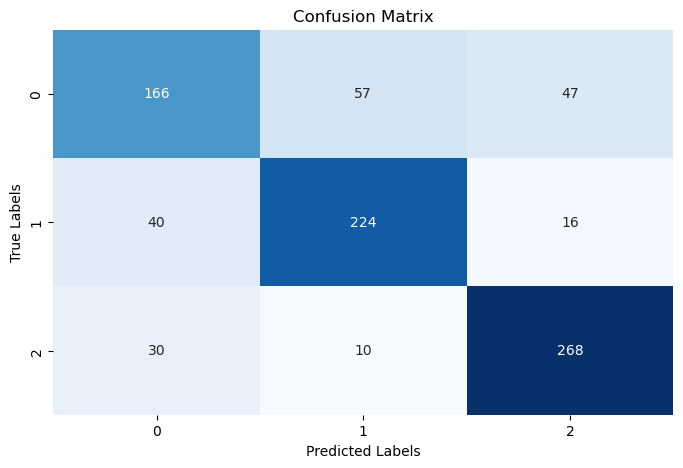


Figure 44: Confusion Matrix Heatmap of Random Forest Classifier

A close-up of a number

AI-generated content may be incorrect.

1. **AUC – ROC Curve:**

The areas under the curve (AUC) score are high, indicating strong predictive performance: 0.83 for Hate Speech, 0.90 for Offensive Language and 0.94 for Neither i.e. Neutral.

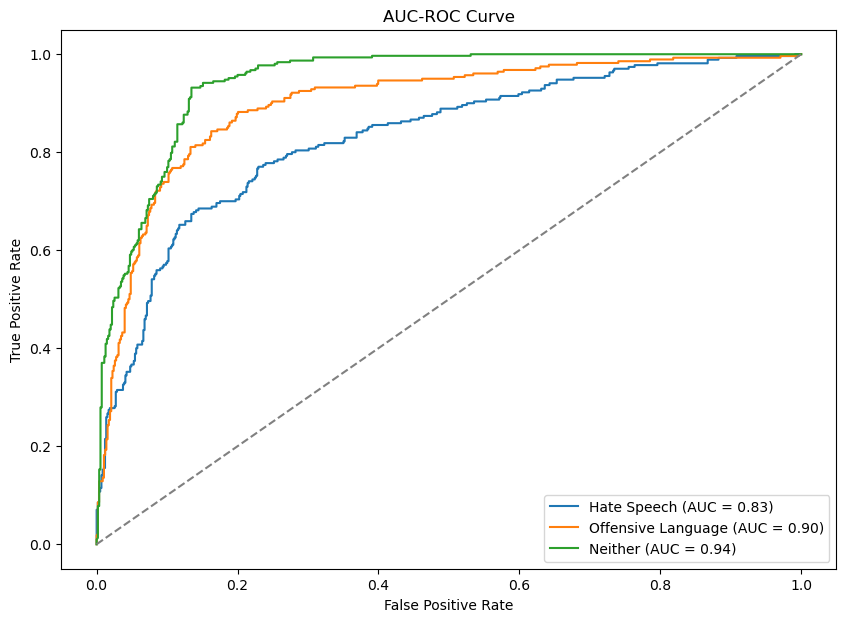


Figure 45: AUC-ROC Curve for Random Forest Classifier

1. **Logarithmic Loss:**

The logarithmic loss for our Logistic Regression model is **0.779.**

#### **Performance Evaluation of Support Vector Machine (Classifier)**

Our SVM model has a **prediction accuracy** of **81%**. The overall precision, recall and F1 score of SVM are:

* + 1. **Prediction Accuracy:** 81%
    2. **Precision:** 0.80
    3. **Recall:** 0.80
    4. **F1 Score:** 0.80

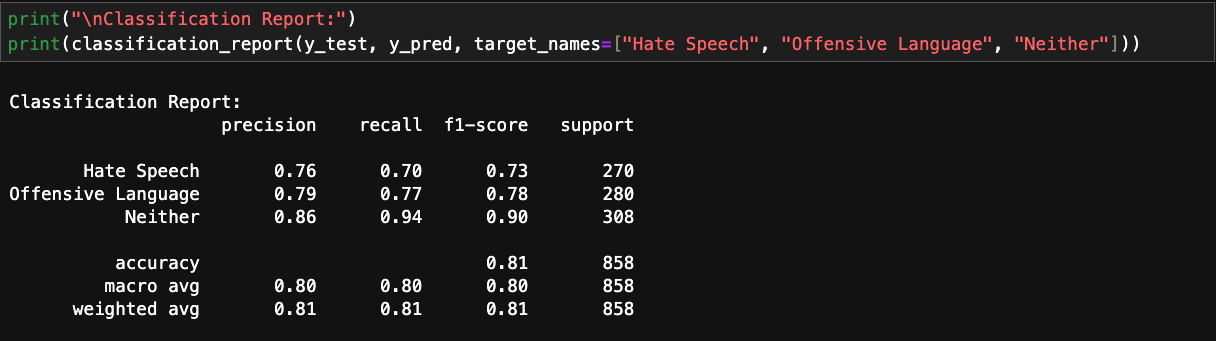


Figure 46: Classification Report of SVM

The detailed classification report is presented in the figure above, where information about the precision, recall and f1-score of individual classes is shown.

* + 1. **Confusion Matrix:**

The figure below is the confusion matrix heatmap of the SVM model, which shows the TP, FP, FN, and TN values predicted by the model.

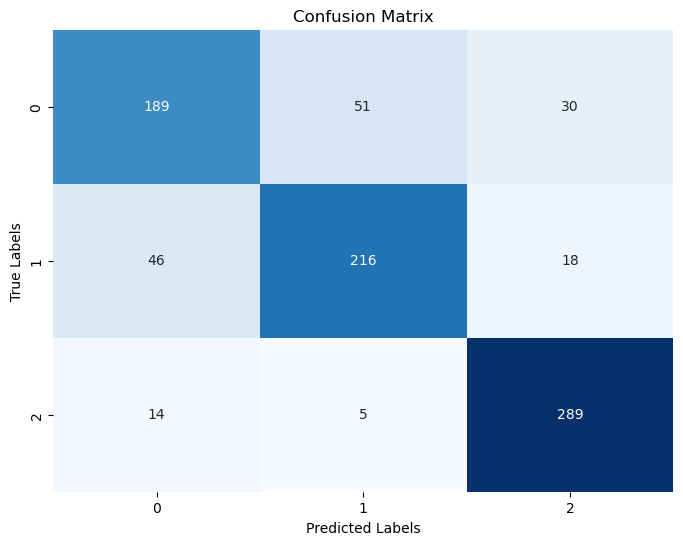


Figure 47: Confusion Matrix Heatmap of SVM

A black text on a white background

AI-generated content may be incorrect.

1. **Logarithmic Loss:**

The logarithmic loss for our Logistic Regression model is **0.524.**

1. **AUC – ROC Curve:**

The areas under the curve (AUC) score are high, indicating strong predictive performance: 0.87 for Hate Speech, 0.91 for Offensive Language and 0.97 for Neither i.e. Neutral.

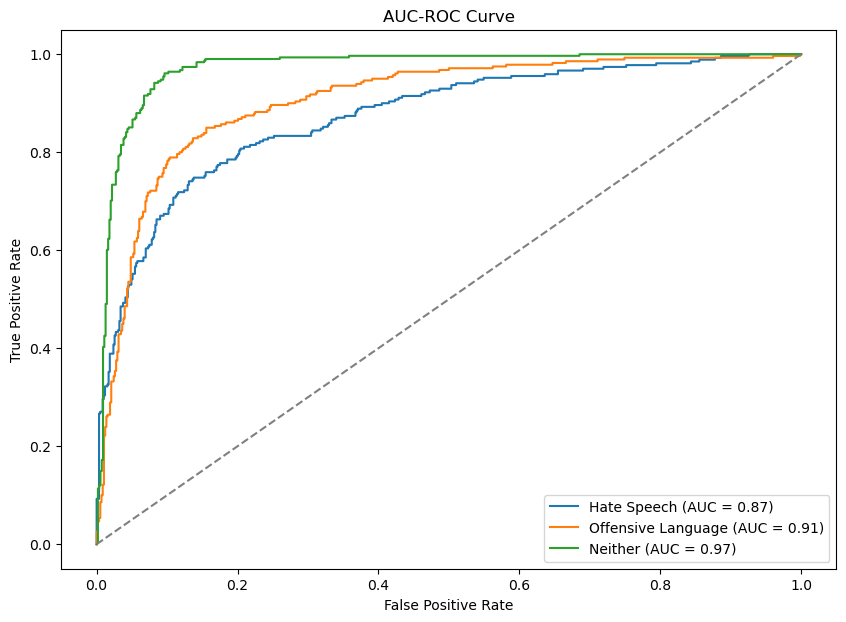


Figure 48: AUC-ROC Curve for SVM

#### **Performance Comparison of Different Models**

Table 3: Overall Performance Comparison of Different Algorithms

A table with numbers and text

AI-generated content may be incorrect.

* + 1. From the above model evaluation, we have Logistic Regression as the best-performing model with 82% accuracy, followed by SVM (81%).
    2. We have SVM with the smallest logarithmic loss (0.524) followed by Logistic Regression (0.567).

### **Model Deployment**

Model deployment is essential because it allows us to make machine learning models accessible for practical use, enabling real-world applications like predicting outcomes, automating processes, or making decisions in real-time.

We used Logistic Regression for the deployment, as this has the best accuracy (82% accuracy) among three of the algorithms we used. For the model deployment, we used a Streamlit library, which is an open-source Python library used to create interactive and user-friendly web applications for machine learning and data science projects with minimal coding effort. The following figure shows the code to implement Streamlit to deploy our model as a web app.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 49: Code Showing Preprocessing and Feature Engineering in Model Deployment

In the above figure, the required libraries, trained model, vectorizer, and PCA are imported. The code to preprocess the data is shown in the figure.

The figure below shows the code to implement feature engineering of input data and the trained model classifying the input data, whether it is hate speech, offensive language or neutral.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 50: Implementation of Feature Engineering and Prediction of Real-world Input Data

**Results of Model Deployment**

**A black and white screen with white text

AI-generated content may be incorrect.**

Figure 51: Interface of Streamlit Web App

A screenshot of a computer

AI-generated content may be incorrect.

Figure 52: Test Prediction One

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 53: Test Prediction Two

A screenshot of a computer

AI-generated content may be incorrect.

Figure 54: Test Prediction Three

### **Conclusion and Recommendations**

* + 1. **Summary of Key Findings**

In our analysis of hate speech classification, we trained and evaluated three machine learning models: **Logistic Regression**, **Random Forest**, and **Support Vector Machine (SVM)**. The goal was to identify which model performed best in classifying text as hate speech or not. Here are the detailed findings:

#### **Model Performance**

* **Logistic Regression** was the best model, with a predictive accuracy of 82%. This strong performance is due to its ability to effectively handle high-dimensional data, which is crucial for text features derived from techniques like TF-IDF.
* **Support Vector Machine (SVM)** closely followed with an accuracy of **81%**, demonstrating its strength in creating decision boundaries for complex datasets.
* **Random Forest**, while robust in many applications, performed slightly worse in this context, likely due to its less effective handling of sparse and high-dimensional text data.

#### **Insights Gained**

* **Effectiveness of Linear Models**: Logistic Regression's superior performance highlights that linear models are often well-suited for text classification tasks, especially when paired with appropriate feature extraction techniques like TF-IDF.
* **SVM's Competitiveness**: SVM’s performance being close to Logistic Regression indicates its capability in effectively separate classes, even in high-dimensional feature spaces.
* **Random Forest’s Limitation**: Despite its ensemble nature, Random Forest struggled to achieve comparable accuracy, suggesting that tree-based models may not generalize as well for sparse text data.

#### **Importance of Preprocessing**

#### The quality of text preprocessing played a pivotal role in the overall performance of the models. Steps such as cleaning the data, removing stop words, tokenizing, and applying TF-IDF vectorization allowed the models to focus on meaningful patterns in the text.

* + 1. **Future Recommendations**

For future work, additional steps could be taken to enhance the model's performance and scalability.

1. **Explore Advanced Algorithms**: Future work could involve implementing more advanced deep learning approaches like Recurrent Neural Networks (RNNs) and Transformer-based models (e.g., BERT) to capture contextual information and improve classification accuracy.
2. **Data Augmentation**: Expanding the dataset with additional samples, especially for underrepresented categories, could help models generalize better and reduce bias in predictions.
3. **Fine-Tuning Feature Engineering**: Experimenting with advanced text representation techniques, word embeddings (Word2Vec, GloVe), or sentence embeddings (BERT embeddings) could improve model performance by capturing semantic relationships more effectively.
4. **Hyperparameter Optimization**: Conducting a more extensive hyperparameter tuning process using techniques like Bayesian Optimization could enhance model accuracy and stability.
5. **Real-World Application Testing**: Deploying the models in real-world scenarios, such as social media platforms or content moderation systems, would provide valuable insights into their practicality and scalability. Regular retraining with fresh data would ensure the models remain relevant.
6. **Explainability and Fairness**: Incorporating explainable AI (XAI) methods and bias-detection frameworks can make the models more transparent and fairer, particularly in sensitive contexts like hate speech detection.

Our analysis shows that **Logistic Regression** is the most effective model for hate speech classification in this setup, achieving the highest accuracy of **82%**, closely followed by **SVM** with **81%**. These results reinforce the idea that simpler models can perform exceptionally well on text-based tasks when paired with appropriate preprocessing techniques. However, there is room to explore advanced methods like transformers to push the performance further.

# **Bibliography**

Crabtree, M., 2024. *What is Machine Learning? Definition, Types, Tools & More.* [Online]   
Available at: https://www.datacamp.com/blog/what-is-machine-learning  
[Accessed 20 January 2025].

GeeksForGeeks, 2024. *Evaluation Metrics in Machine Learning.* [Online]   
Available at: https://www.geeksforgeeks.org/metrics-for-machine-learning-model/  
[Accessed 21 January 2025].

GeeksforGeeks, 2025. *Types of Machine Learning.* [Online]   
Available at: https://www.geeksforgeeks.org/types-of-machine-learning/  
[Accessed 20 January 2025].

IBM, 2023. *Five Machine Learning Types to Know.* [Online]   
Available at: https://www.ibm.com/think/topics/machine-learning-types  
[Accessed 20 January 2025].

Java Tpoint, 2024. *Support Vector Machine Algorithm.* [Online]   
Available at: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm  
[Accessed 25 January 2025].

JavaPoint, 2024. *K-Means Clustering Algorithms.* [Online]   
Available at: https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning  
[Accessed 20 January 2025].

Joby, A., 2021. *What Is Data Preprocessing? 4 Crucial Steps to Do It Right..* [Online]   
Available at: https://learn.g2.com/data-preprocessing  
[Accessed 23 January 2025].

Keita, Z., 2024. *Classification in Machine Learning: An Introduction.* [Online]   
Available at: https://www.datacamp.com/blog/classification-machine-learning  
[Accessed 21 January 2025].

Khushaktov, M. F., 2023. *Introduction to Random Forest Classification with Example.* [Online]   
Available at: https://medium.com/@mrmaster907/introduction-random-forest-classification-by-example-6983d95c7b91  
[Accessed 24 January 2025].

Kumar, S., 2024. *Evaluation Metrics For Classification Model.* [Online]   
Available at: https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/  
[Accessed 20 January 2025].

Madushan, D., 2017. *Introduction to K-Means Clustering.* [Online]   
Available at: https://medium.com/@dilekamadushan/introduction-to-k-means-clustering-7c0ebc997e00  
[Accessed 20 January 2025].

Melanie, 2023. *Classification algorithms: Definition and main models.* [Online]   
Available at: https://datascientest.com/en/classification-algorithms-definition-and-main-models  
[Accessed 21 January 2025].

Pandian, S., 2024. *A Comprehensive Guide on Hyperparameter Tuning and its Techniques.* [Online]   
Available at: https://analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/  
[Accessed 25 January 2025].

Rinkal, J., 2023. *Introduction to K-Means Clustering.* [Online]   
Available at: https://www.bombaysoftwares.com/blog/introduction-to-k-means-clustering  
[Accessed 20 January 2025].

Sharma, P., 2025. *What is K-Means Clustering?.* [Online]   
Available at: https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/  
[Accessed 20 January 2025].

Tech-AI-Math, 2023. *Logistic Regression in Depth.* [Online]   
Available at: https://ai.plainenglish.io/logistic-regression-543c8424595d  
[Accessed 24 January 2025].

Wakefield, K., 2024. *A Guide to The Types of Machine Learning Algorithms and Their Applications.* [Online]   
Available at: https://www.sas.com/en\_gb/insights/articles/analytics/machine-learning-algorithms.html  
[Accessed 20 January 2025].

Zheng, A., 2015. *Evaluating Machine Learning Models A beginner's guide to key concepts and pitfalls..* [Online]   
Available at: https://www.oreilly.com/content/evaluating-machine-learning-models/  
[Accessed 20 January 2025].