Technical University of Košice

Faculty of Electrical Engineering and Informatics

Department of Cybernetics and Artificial Intelligence

# Documentation for Toxic Comment Detector

Instructor:

Ing. Viliam Balara

Student:

Oleksandr Kostianets

Košice 2025

# Contents

# List of Tables

# 1 Introduction

This project is a toxic comment detection system that analyzes text for different types of problematic content. It uses machine learning to automatically classify comments as toxic, abusive, or provocative.

The aim of the project was to implement one of the machine learning algorithms discussed and to practically apply the theoretical knowledge gained. In this project, bagging algorithm (Bootstrap Aggregating) is used in combination with a Naive Bayes classifier, while the algorithms themselves were programmed from scratch without using ready-made library functions. The dataset used for the experiments contains real text data (specific comments) along with the corresponding toxicity assessments. In addition, the project contains its own implementation of metrics: accuracy, precision, recall and F1 measure, which were created from scratch, in accordance with the requirements of the assignment.

# 2 Dataset overview

The dataset used for training models comes from **kaggle** called *"youtoxic_english_1000.csv"*. Data contains 1000 entries and the following attributes:

- Text: Text of comment that program will use to train models.
- IsToxic, IsAbusive, IsProvocative: Tells if text is toxic, abusive or provocative respectively (FALSE – non-toxic, TRUE – toxic).

There is also 11 columns that will not be described such as VideoId, CommentId due to their non-usage.

**Table 2 – 1**: Dataset preview

| Text | IsToxic | IsAbusive | IsAbusive |
|---|---|---|---|
| If only people would just take a step back and not make this case about them, because... | FALSE | FALSE | FALSE |
| Law enforcement is not trained to shoot to apprehend. They are trained to shoot to kill. And I tha... | TRUE | TRUE | FALSE |
| Dont you reckon them 'black lives matter' banners being held by white cunts is kinda patronizing a... | TRUE | TRUE | FALSE |
| There are a very large number of people who do not like police officers. They are called Criminals a... | FALSE | FALSE | FALSE |
| The Arab dude is absolutely right, he should have not been shot 6 extra time. Shoot him once if hes ... | FALSE | FALSE | FALSE |

# 3 Theoretical description of algorithms

Project has 2 implemented algorithms: **Bagging** and **Naive Bayes classifier**.

## 3.1 Bagging (Bootstrap Aggregating)

Bagging is an ensemble meta-estimator method that combines multiple models to reduce variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it. [1] The basic principle consists of:

- Creating multiple bootstrap samples from the training data (random selection with repetition).
- Training multiple copies of the base model (in our case, the Naive Bayes classifier) on these samples.
- Aggregating the predictions using voting, where the final class is determined by majority decision.

## 3.2 Naive Bayes classifier

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. [2] In our implementation, a bag-of-words approach with Laplace smoothing is used. This approach allows the model to:

- Calculate the probabilities of occurrence of each word in the text for each class.
- Avoid problems with zero probabilities by using the smoothing parameter (alpha).

# 4 Description of implementation and features

The project is divided into several modules, with each module providing a specific part of the workflow:

## 4.1 Loading data

**File:** *data_loader.py*

**Functions:**

- `load_data(label: str)` – Loads a CSV file containing data, assuming that the file contains a Text column with text data and another column whose name is given as a parameter (representing a label). The output is a pair of lists - the texts and the corresponding labels.

## 4.2 Model training

**File:** *model_trainer.py*

**Functions:**

- `train_model(texts, labels)` – randomly shuffles the data, splits it into training and test sets, trains a bagging classifier and evaluates it using the implemented metrics.
- `get_trained_model(label: str)` – integrated function that first loads the data and then starts training the model.

## 4.3 Saving and loading the model

**File:** *model_saver.py*

**Functions:**

- `auto_save_best_model(model, metrics, model_filename)` – automatically saves the current model to disk if its F1 score is better than that of an already saved model.
- `load_best_model(model_filename)` – loads the best saved model along with its metrics.

These functions use the pickle library to serialize the model and also use notifications via Streamlit for the UI.

## 4.4 Model evaluation

**File:** *evaluation.py*

**Functions:**

- `accuracy_metric(y_true, y_pred)` – Proportional representation of correctly classified records.
- `precision_metric(y_true, y_pred, positive=1, average='weighted')` – Proportion of correct positives out of all predicted positives.
- `recall_metric(y_true, y_pred, positive=1, average='weighted)` – The proportion of correctly captured positives out of all true positives.
- `f1_metric(y_true, y_pred, positive=1, average='weighted)` – Harmonic mean of precision and recall.

Each of these functions is implemented manually without the use of library functions, which satisfies the specification

## 4.5 Implementation of the algorithms

- *naive_bayes.py* – A simple **Naive Bayes classifier** with `fit` (for training) and `predict` (for prediction) methods is implemented in the file. The class uses tokenization methods, word counting, and computation of log-likelihoods for each class.

- *bagging.py* – A **BaggingClassifier** class that creates multiple bootstrap samples from the training data, trains an instance of the base classifier (**Naive Bayes**) for each sample, and then aggregates the predictions using the majority voting method.

# 5 Evaluation and interpretation of results

## 5.1 Metrics

- **Accuracy:** proportion of all classifications that were correct, whether positive or negative.
- **Precision:** Determines how much of the predicted positives are actually correct.
- **Recall:** Expresses how well the model captures all true positive cases.
- **F1 Score:** A harmonic average of precision and recall that provides a balanced view of the model's performance.

Each metric has been programmed manually, ensuring transparency of calculations and complete control over the calculation process.

## 5.2 Interpretation of results

- **The higher Accuracy and F1 Score** values indicate that the model generalizes well to unseen data.
- **Precision** is important when we want to minimize the number of false positive classifications.
- **Recall** is key when we need to capture as many positive cases as possible (e.g., when detecting toxic comments).

The results obtained through validation testing (by splitting the data into training, test and evaluating sets) provide practical information about the performance of the model, allowing for possible further optimization.

## 5.3 Comparison of model performance

In this section, the best results achieved by each implemented model on evaluation metrics (Accuracy, Precision, Recall, and F1 Score) are summarized. These metrics are crucial in determining the quality and effectiveness of our models in the task of text classification.

**Table 5 – 1**: Comparation of metrics

| Model | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| IsToxic | 95,33% | 94,44% | 95,77% | 95,10% |
| IsProvocative | 95,33% | 95,42% | 95,33% | 95,37% |
| IsAbusive | 95,33% | 95,33% | 95,33% | 95,33% |

Also decided to compare the results of these custom implementations with equivalent metrics from the Scikit-learn library, specifically with the `accuracy_score`, `precision_score`, `recall_score`, and `f1_score` functions from the *sklearn.metrics* module. This comparison step served to verify the correctness of the calculations and ensure that the implemented metrics provide consistent results with the recognized standard in the field of machine learning.

**Table 5 – 2**: Comparation of metrics with sci-kit

| Model | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| IsToxic | 87,60% | 87,92% | 88,96% | 86,89% |
| IsProvocative | 89,80% | 66,85% | 68,46% | 67,64% |
| IsAbusive | 87,60% | 80,05% | 86,40% | 83,11% |

# 6 Conclusion

In this work, a **Naive Bayes classifier** combined with **Bagging** has been successfully implemented. The project meets all the specified requirements:

- Custom implementation of the algorithms without using library functions.
- Manual implementation of metrics (accuracy, precision, recall, F1).
- Use of real dataset with more than 4 attributes and more than 20 values.
- Documentation in thesis template including theoretical description, implementation description, data analysis and evaluation of results.

The project presents a comprehensive approach to solving the machine learning problem, while allowing for further extensions and optimization.

# References

[1] Scikit-learn Developers. *BaggingClassifier - Scikit-learn Documentation*, 2024.

[2] Scikit-learn Developers. *Naive Bayes - Scikit-learn Documentation*, 2024.