MACHINE LEARNING PROJECT

1.Title:

Detecting Toxicity in Online Comments- Toxic Comment Classification

2.Justification

In an increasingly digital world, the ability to engage in constructive and respectful online discourse is crucial. However, the prevalence of toxic comments- such as insults, threats, and identity-based hate—poses significant barriers to open communication and can lead to a decline in user engagement on online platforms. I chose the Toxic Comment Classification dataset for several compelling reasons:

Societal Relevance

- Mitigating Online Harassment
- Encouraging Healthy Online Communities

The Toxic Comment Classification dataset provides an invaluable resource for addressing the pressing issue of online toxicity. By leveraging machine learning to analyse and classify toxic comments, we can contribute to the development of tools that foster respectful and productive online conversations. The insights gained from this analysis not only have the potential to improve individual user experiences but also to shape broader societal conversations around online safety and community engagement.

3. Understanding of the Dataset

This task involves building a **multi-headed classification model** to detect multiple types of toxicity simultaneously in text. Specifically, the project requires to develop a model that predicts the **probability of each type of toxicity** (rather than simple binary classification) for each comment in a dataset.

The types of toxicity in the dataset are:

1. toxic – General toxicity (e.g., abusive language, hostile behaviour).
2. severe_toxic – Highly aggressive or severe toxic content.
3. obscene – Use of profane or vulgar language.
4. threat – Comments that contain threats or warnings of harm.

5. insult – Personal attacks or name-calling.
6. identity_hate – Hate speech directed at specific identities (race, gender, etc.).

Description of data

- **Source of the Data**

  The dataset for this project is derived from the Wikipedia talk page edits which is available in Kaggle. The primary goal of this project is to enhance the detection of toxic comments in online discussions, thus fostering more respectful and constructive conversations.

- **Dataset Overview**
  - Training Data:
    - Rows: 159,751
    - Columns: 8

    train.csv - The training set, containing:

    1. Comments (text data)

    2. Binary labels (0 or 1) indicating whether each type of toxicity is present in the comment. For example:

    - A comment may have toxic = 1 and threat = 0.
    - Another may have multiple toxic types labelled (e.g., insult = 1 and identity_hate = 1).
  - Test data:
    - Rows: 153,164
    - Columns: 2
    - The test set, containing only comments.

Features/Variables

The training dataset consists of the following key features:

1. comment_text
   - Type: Text

   - Description: The actual comment text from the Wikipedia talk pages that needs to be analyzed for toxicity.

2. toxic:
   - Type: Binary (0 or 1)
   - Description: Indicates whether the comment is classified as toxic (1) or not (0).
3. severe_toxic:
   - Type: Binary (0 or 1)
   - Description: Indicates whether the comment is classified as severely toxic (1) or not (0).
4. obscene:
   - Type: Binary (0 or 1)
   - Description: Indicates whether the comment contains obscene language (1) or not (0).
5. threat:
   - Type: Binary (0 or 1)
   - Description: Indicates whether the comment includes a threat (1) or not (0).
6. insult:
   - Type: Binary (0 or 1)
   - Description: Indicates whether the comment is classified as an insult (1) or not (0).
7. identity_hate:
   - Type: Binary (0 or 1)
   - Description: Indicates whether the comment expresses hate towards a specific identity group (1) or not (0).

Missing Values, Outliers, and Data Imbalances

- Missing Values: The dataset does not have any missing values in the training set, as each comment has been labelled for all toxicity types.
- Outliers: The text data may contain outliers in the form of unusually short or long comments, but since the focus is on the classification of toxicity, these will be handled through preprocessing steps (such as text normalization).
- Data Imbalances: The training dataset is likely to exhibit some level of class imbalance, as toxic comments are generally less frequent than non-toxic ones. Addressing this imbalance may be necessary during the modelling process to ensure the model does not become biased towards predicting the majority class.
    - Percentage of non-toxic comments is 89.83211235124176

Data Cleaning and Preprocessing:

1. Handling Missing Data

- Missing Values: The training dataset was inspected for missing values, and it was found that there were no missing entries in the relevant columns. Since every comment was labelled for all toxicity types, no imputation or removal of rows was necessary.

2. Outlier Treatment

- Outliers in Text Data: Although outliers in text data (e.g., comments that are unusually short or long) can be identified, no specific outlier treatment was performed. Instead, the focus was on text normalization to ensuring consistent processing of all comments.

3. Text Normalization

To prepare the text data for modelling, several normalization steps were applied:

- Lowercasing: All text in the comment_text column was converted to lowercase to ensure uniformity and reduce the dimensionality of the vocabulary.

- Contraction Expansion: A dictionary of common English contractions was used to expand contractions in the comments (e.g., "what's" to "what is"), which helps in understanding the context better.
- Removing Non-Word Characters: Non-word characters (e.g., punctuation) were removed to focus on the actual words within the comments. This was accomplished using regex replacements.
- Removing Extra Spaces: Any instances of multiple consecutive spaces were replaced with a single space to ensure that the text is clean and consistently formatted.

4. Encoding Categorical Variables

- Target Variables: The target variables (i.e., toxicity labels) in the training dataset were already in binary format (0 or 1) and thus did not require any further encoding. Each label represented whether a specific type of toxicity was present or not.

5. Feature Engineering

- **Creating Additional Features:**
    - None Class: A new column named none was created, indicating whether a comment was classified as non-toxic. This was calculated by taking the maximum value across the toxicity columns (toxic, severe_toxic, obscene, threat, insult, and identity_hate). If none of these labels were active (i.e., all were 0), the none column would be set to 1.
- **TF-IDF Vectorization:** After preprocessing the comments, TF-IDF (Term Frequency-Inverse Document Frequency) vectorization was applied to transform the cleaned text data into a numerical format suitable for machine learning. This step included:
    - Max Features: The TF-IDF vectorizer was configured to limit the number of features to 200,000, focusing on the most informative words while ignoring common stop words in English.

6. Splitting the Data

- The dataset was then split into training and validation sets using the train_test_split function, ensuring that both sets contained balanced distributions of toxicity labels for proper evaluation of model performance.

4. Algorithm usage

- **Logistic Regression**
  Logistic Regression was chosen for the Toxic Comment Classification Challenge due to its effectiveness in handling binary and multi-class classification problems. This algorithm is particularly suitable for the dataset as it provides interpretable results, allowing for easy understanding of the relationship between input features and the output labels. Given the nature of the problem, where comments can be classified into various toxicity categories, Logistic Regression can effectively model these relationships. Additionally, it computes probabilities that are well-calibrated, enabling us to understand the confidence in the predictions. Its simplicity and computational efficiency make it a practical choice for initial experiments, allowing for rapid iteration and evaluation of model performance.
- **Naive Bayes**
  Naive Bayes was also selected for this due to its strengths in text classification tasks. This algorithm operates on the principles of conditional probability and is particularly advantageous when working with high-dimensional data, like the comments in this dataset. The independence assumption of features allows Naive Bayes to be computationally efficient, making it suitable for handling large volumes of text data quickly. It excels in situations where the data is represented in a bag-of-words or term frequency-inverse document frequency (TF-IDF) format, as it can effectively calculate the probabilities of various classes based on the presence of words in the comments. This makes Naive Bayes a robust choice for multi-class classification problems, as it can efficiently handle the multiple types of toxicity present in the dataset.
- **Random Forests**
  Random Forest is a powerful ensemble learning method that combines multiple decision trees to improve prediction accuracy and control

overfitting. In the case of toxic comment classification, Random Forest performs classification by building multiple decision trees, each trained on a random subset of the training data and features. For each decision tree, a decision is made by splitting the data based on feature values, for example specific words or combinations of words in the comment) that best separate the classes (toxic categories).

When a new comment is fed into the trained Random Forest model, it is passed through each of the decision trees, where each tree makes a prediction that is a label or class for the comment. Thus, Random Forest not only classifies the comments but also gives insight into the likelihood or confidence of each class, making it useful for situations where understanding the model's certainty is important as in our case.

- **SVM**

  SVM is well-suited for problems with high-dimensional data, such as text classification tasks, where features (e.g., word frequencies or TF-IDF scores) can be numerous. In text classification, each word or term could be treated as a feature, leading to a high-dimensional feature space. SVMs are effective at handling such feature spaces, making them a strong choice for text-based datasets. With the use of the **kernel trick**, SVM can handle both linear and non-linear classification problems. In the case of toxic comment classification, the relationships between words and the toxicity of comments might not always be linear, so the ability of SVM to use non-linear kernels (like the radial basis function) allows it to capture complex patterns in the data.

  For each class (in this case, toxicity categories), the SVM algorithm computes the decision boundary (hyperplane) that separates comments of one class from those of another. Once the hyperplane is determined, new comments are classified by determining on which side of the hyperplane they lie.
- **RNN (Recurrent Neural Networks)**

  Recurrent Neural Networks (RNNs) are especially well-suited for sequence data, making them a strong choice for text classification tasks like toxic comment classification. Unlike traditional feedforward neural networks, RNNs have loops that allow information to persist, enabling them to maintain a memory of previous inputs. This is particularly advantageous for text data, where the context and sequence of words can greatly influence the meaning and classification of a comment.

In the case of toxic comment classification, RNNs process input text one word at a time while maintaining a hidden state that reflects the context accumulated from previous words. This hidden state allows the RNN to remember important features from earlier parts of the comment, enabling it to capture dependencies across long sequences. For example, if a comment has complex phrasing or contains toxicity indicators spread throughout, an RNN can learn to identify these patterns effectively.

When an RNN processes a comment, it outputs a final state that summarizes the entire sequence. This output is then passed to a fully connected layer, which applies an activation function like sigmoid for multi-label classification. The sigmoid activation function outputs probabilities between 0 and 1 for each class, representing the likelihood that the comment belongs to each toxic category. These probabilities help determine which labels to assign, allowing the model to predict whether a comment falls into one or more toxicity categories simultaneously.

Model Performance

- **Evaluation Metrics**: The models were evaluated using various metrics, including:
    - **Accuracy**: This metric provides an overall success rate of the model but can be misleading in imbalanced datasets.
    - **F1 Score**: This is crucial for understanding the balance between precision and recall, especially in scenarios where false positives and false negatives have different costs.
    - **ROC-AUC**: This metric was particularly useful in evaluating the performance of both models across different thresholds, indicating how well the models distinguish between classes.

5. Inferences Drawn

Comparison of Results on validation data

- **Logistic Regression**

Evaluation metric	obscene	insult	toxic	Severe toxic	identity hate	threat
accuracy	0.946691	0.95018	0.904830	0.989681	0.99043	0.997243
f1 score	0.013148	0.01329	0.012142	0.042636	0.00868	0.204819
precision	0.566667	0.53333	0.933333	0.366667	0.06667	0.566667

recall	0.006651	0.00667	0.006111	0.022634	0.00460	0.125000
roc_auc	0.924179	0.90807	0.875979	0.968606	0.92768	0.974199

- Naïve Bayes classifier

Evaluation metric	obscene	insult	toxic	Severe toxic	identity hate	threat
accuracy	0.946629	0.95017	0.904307	0.989827	0.990955	0.997180
f1 score	0.000782	0.000837	0.000436	0.000000	0.000000	0.014599
precision	1.000000	1.00000	1.000000	0.000000	0.000000	1.000000
recall	0.000391	0.000419	0.000218	0.000000	0.000000	0.007353
roc_auc	0.697713	0.702732	0.679744	0.788297	0.728007	0.785881

- SVM

Evaluation metric	obscene	insult	toxic	Severe toxic	identity hate	threat
accuracy	0.979215	0.970651	0.960018	0.990245	0.991999	0.997389
f1 score	0.782419	0.660218	0.766357	0.369771	0.351946	0.358974
precision	0.886961	0.781787	0.869529	0.537255	0.654088	0.593220
recall	0.699922	0.571369	0.685072	0.281893	0.240741	0.257353
roc_auc	0.979939	0.964069	0.963470	0.966348	0.955356	0.966797

- Random Forest

Evaluation metric	obscene	insult	Severe toxic	toxic	identity hate	threat
accuracy	0.964760	0.962838	0.989367	0.938524	0.979612	0.997180
f1 score	0.603898	0.483599	0.182986	0.599537	0.039370	0.105960
precision	0.755138	0.788826	0.416058	0.796169	0.034247	0.533333
recall	0.503130	0.348681	0.117284	0.480794	0.046296	0.058824
roc_auc	0.949268	0.931916	0.880230	0.916295	0.800963	0.779934

- Recurrent Neural Networks with LSTM
    - ROC-AUC for obscene: 0.9833587728447952
    - ROC-AUC for insult: 0.9737725318316577
    - ROC-AUC for toxic: 0.9611289125475071
    - ROC-AUC for severe_toxic: 0.9817814867442747
    - ROC-AUC for identity_hate: 0.953181845684217

  - ROC-AUC for threat: 0.9610004226678103

The dataset is imbalance with approximately 90% of them are non-toxic comments. So purely basis on accuracy evaluation metric is not suitable.
ROC-AUC is a robust metric for evaluating how well a model distinguishes between classes. From the above collected data report RNN and SVM would be the better choice for the toxic comment's dataset.

RNNs are specifically designed to handle sequential data, where the order of the data points matters. This is especially important for text-based tasks like text classification, so going with RNN for making predictions on test data.

<u>Detailed Work plan</u>

1. Data Upload:

- Loaded training and test datasets using pandas from CSV files.

2.Data Preprocessing

3.Train-Validation Split:

- Split the training data into training and validation sets to evaluate model performance using train_test_split.

4.Feature Extraction:

- Used TF-IDF Vectorization to convert preprocessed comments into numerical features for model training.

5.Model Training:

- Implemented Logistic Regression and Naive Bayes classifiers.
- Trained both models on the training set.

6.Model Evaluation:

- Made predictions on the validation set using both models.
- Calculated evaluation metrics (accuracy, F1 score, precision, recall, ROC-AUC) for each class.
- Compared model performance on the validation set.

7.Model Selection:

- Selected the model that performed best on the validation set for predictions on the test data.

8.Test Data Predictions:

- Used the selected model to predict labels for the test dataset.

9.Visualization:

- Created visualizations, including:
    - Bar plots showing the count of each toxicity label.
    - Distribution plots of comment lengths to analyse text characteristics.
    - Correlation matrix heatmap to understand relationships between different toxicity labels.

10. Output

- The output that is predicted on test data is been saved in a csv file (test_predictions.csv).

Conclusion

The analysis and model development provided valuable insights into the process of predicting multiple labels for text data. Deploying this multi-label classification model in real-world applications can bring significant improvements in how businesses, content platforms, and customer service teams process and respond to user-generated text. The ability to predict multiple labels from a single piece of text allows for more sophisticated and timely actions, making the process of content moderation, feedback analysis, and customer interaction more automated, accurate, and scalable.

If deployed in a content moderation system, the model could be used to automatically categorize and flag comments according to their content (e.g., offensive language, spam, or inappropriate content). For example, the model could predict multiple types of harmful content in a comment, allowing moderation teams to review and take action accordingly. This can significantly improve user experience and reduce the workload on human moderators.

In business contexts, such as customer service or feedback analysis, this model could classify customer comments or survey responses into multiple categories, such as satisfaction, issues, product feedback, etc. This could help businesses prioritize responses, identify frequent concerns, and improve products or services more effectively.