### Miimansa Assignment Submission

### | Archana Trivedi | [archanatrivedi93@gmail.com|](mailto:archanatrivedi93@gmail.com|)

### Approach 1: Traditional Machine Learning (Code in MachineLearning\_approach.py)

**1. Data Loading and Preprocessing:**

* The training and validation datasets are loaded from CSV files and preprocessed to remove special characters, extra spaces, and convert text to lowercase for normalization.

**2. Data Splitting:**

* Validation data is split further into validation and test sets to ensure models are evaluated on unseen data.

**3. Feature Extraction:**

* TF-IDF vectorization is employed to convert text data into numerical features, capturing word importance relative to the entire dataset.

**4. Model Initialization:**

* Various classification models are initialized, including Logistic Regression, Naive Bayes, SVM, Random Forest, Gradient Boosting, and a Voting Classifier combining all models using soft voting.

**5. Model Training and Evaluation:**

* Each model is trained on the training dataset and evaluated on the validation and test datasets to calculate weighted F1 scores. Detailed classification reports are generated for each model.

**6. Results Storage and Comparison:**

* Results such as validation/test F1 scores and classification reports are stored and saved to CSV files for future reference and analysis.

**Assumptions:**

* Assumes clean, correctly labeled datasets.
* Assumes consistent labeling without typos.
* Assumes TF-IDF captures sufficient textual features.
* Assumes models generalize well to unseen data.
* Assumes the weighted F1 score handles class imbalances adequately.

**Future Scope:**

* Hyperparameter tuning for improved performance.
* Exploration of advanced text representations (e.g., embeddings, transformers).
* Experimentation with ensemble methods beyond simple voting.
* Additional feature engineering techniques.
* Integration of deep learning models for sequential data analysis.
* Implementation of data augmentation for improved model robustness.
* Utilization of cross-validation for more robust performance metrics.
* Incorporation of explainability techniques for model insights.
* Development of pipelines for real-time model deployment and monitoring.

### Approach 2: Transformer-based Deep Learning (Code in Transformers\_approach.py)

**1. Data Loading and Preprocessing:**

* Data undergoes preprocessing including lowercasing, removing stopwords, stemming, and filtering based on word length.

**2. Data Splitting:**

* Validation data is split into validation and test sets. Training and validation sets are concatenated for final model training.

**3. Label Encoding:**

* Labels are encoded numerically using LabelEncoder for model compatibility.

**4. Model and Tokenizer Setup:**

* Tokenizers and configurations are set up for transformer models like BERT, DistilBERT, RoBERTa, and XLM-RoBERTa. Custom functions and classes facilitate dataset creation.

**5. Training:**

* Models are fine-tuned using Trainer from the transformers library, with predefined training parameters like epochs, batch size, and learning rate.

**6. Evaluation:**

* Models are evaluated on the test set using the best-performing checkpoint based on F1 score. Results are stored for comparison.

**Assumptions:**

* Preprocessing steps (e.g., stopword removal, stemming) enhance model performance.
* Selected transformer models (BERT, etc.) are suitable for the classification task.
* Training parameters (epochs, batch size, learning rate) are optimized for the task.
* Evaluating multiple checkpoints aids in identifying the best model.

**Future Scope:**

* Exploration of additional transformer models (e.g., T5, Longformer, ELECTRA).
* Hyperparameter tuning using grid search or other techniques.
* Implementation of data augmentation strategies for enhanced training.
* Adoption of advanced preprocessing techniques (e.g., lemmatization, named entity recognition).
* Integration of ensemble methods to combine predictions.
* Evaluation using additional metrics and cross-validation.
* Deployment optimization for production environments.

#### **Analysis of Best Performing Model| Performance Metrics** (Detail report with each class is attached in the folder Classification\_code\_results)

|  |  |  |
| --- | --- | --- |
| **Model** | **Validation F1** | **Test F1** |
| Logistic Regression | 0.912699799 | 0.902051532 |
| Naive Bayes | 0.897614841 | 0.889029554 |
| SVM | 0.916463047 | 0.906735702 |
| Random Forest | 0.851919822 | 0.843185298 |
| Gradient Boosting | 0.871625196 | 0.864223437 |
| Voting Classifier | 0.916940606 | 0.910886485 |
| Bert | 0.899880346 | 0.896860346 |
| Distilbert | 0.899648123 | 0.892748123 |
| Roberta | 0.899636473 | 0.894636473 |
| XLM-Roberta | 0.909102839 | 0.899102839 |

Based on the provided validation and test F1 scores, the Voting Classifier appears to be the best performer among the traditional machine learning models, while XLM-RoBERTa stands out among the transformer-based models.

### Voting Classifier

* **Validation F1 Score:** 0.9169
* **Test F1 Score:** 0.9109

The Voting Classifier combines predictions from multiple base models (Logistic Regression, Naive Bayes, SVM, Random Forest, Gradient Boosting) using soft voting. It achieves high performance on both validation and test sets, indicating robustness and generalizability. The ensemble approach likely benefits from leveraging diverse learning algorithms that complement each other, reducing the risk of overfitting and capturing different aspects of the data's complexity.

### XLM-RoBERTa

* **Validation F1 Score:** 0.9091
* **Test F1 Score:** 0.8991

XLM-RoBERTa, a transformer-based model pretrained on multilingual data, demonstrates strong performance but slightly lower than the Voting Classifier. The model's effectiveness lies in its ability to understand contextual nuances and semantic relationships in text across different languages. The slightly lower performance compared to the Voting Classifier may be attributed to factors such as dataset characteristics, fine-tuning parameters, or the specific nature of the classification task.

### Possible Reasons for Performance Differences:

* **Model Complexity:** The Voting Classifier benefits from combining simpler models that collectively perform well across various aspects of the data, while XLM-RoBERTa, being a single complex model, might struggle with specific nuances not captured in its pretrained weights or fine-tuning process.
* **Data Characteristics:** If the dataset has characteristics (e.g., specific linguistic nuances, domain-specific jargon) that favor ensemble learning approaches like the Voting Classifier, it could outperform a single transformer model like XLM-RoBERTa.
* **Fine-Tuning and Hyperparameters:** The effectiveness of each model heavily depends on fine-tuning parameters such as learning rates, batch sizes, and number of epochs. Optimal tuning might differ between the Voting Classifier and transformer models, influencing their respective performances.