Project Proposal Report of Emotion Classification using Machine Learning on a Multi-Class Dataset

Members:
- Carlos Rabat: czr0067@auburn.edu, Project Coordinator
- Samarth Kumar: szk0187@auburn.edu
- Jeronime Houndonougbo: jah0268@auburn.edu

Link of the Dataset: https://huggingface.co/datasets/dair-ai/emotion

The dataset we will be working on is focused on the primary application domain of emotion classification of English Twitter messages. The inputs to the dataset are English Twitter messages, which consist of text data. These messages are collected from Twitter, a popular social media platform. The goal of the project is to classify the input messages into one of the six basic emotions: anger, fear, joy, love, sadness, and surprise. The dataset will serve as the foundation for training and evaluating machine learning models to accurately predict and assign the appropriate emotion label to each Twitter message. By analyzing and classifying emotions in Twitter messages with the algorithms to be implemented, valuable insights can be gained into the emotional responses and sentiments expressed by users in the context of social media interactions.

Please see below a screenshot example of the dataset.

`i cant help but feel that i need to be delicate`	`2 love`
`i just feel for my hubbie all this rubbish is really starting to knock his confidence in the people hes supposed to be…`	`1 joy`

Machine learning techniques will be applied to this problem by training models on the provided dataset to learn the patterns and features associated with different emotions in the text data. The models can then be used to predict the emotions of new, unseen Twitter messages. In this project, there are several potential challenges that may arise. Firstly, handling text data can be challenging due to the need for preprocessing and cleaning. This involves dealing with noise, special characters, and tokenizing the text into meaningful units. Selecting appropriate features to represent the text data is another challenge, as options like bag-of-words, TF-IDF, or word embeddings need to be carefully considered. Class imbalance is another potential challenge in the dataset, as different emotions may have varying frequencies, leading to imbalanced class distributions. Addressing this issue and ensuring fair representation of all emotions during training is important. Model selection and evaluation pose additional challenges. Choosing the most suitable machine learning algorithms for emotion classification and evaluating their performance using appropriate metrics require careful consideration. Overfitting or underfitting the training data is a concern, and striking the right balance in model complexity is crucial. Ensuring the generalization of the trained models is essential. The ability to accurately classify emotions in new, unseen Twitter messages is important for the practical application of the project. Therefore, steps must be taken to ensure that the models generalize well beyond the training data. Overall, these potential challenges (which are not exhaustive) should be taken into account during the project's development and appropriate strategies should be employed to address them effectively.

The dataset found is an existing dataset of English Twitter messages with labeled emotions. It can be used as the primary data source.

- **Given Information about the dataset :**
    - "Unsplit" configuration: 416,809 examples for training.

- "Split" configuration: 16,000 examples for training, 2,000 examples each for validation and testing.
- 6 classes: anger, fear, joy, love, sadness, surprise.
- No attributes mentioned. Planning to generate 20 attributes (excluding the class label) ourselves.

In the "unsplit" configuration, there are 416,809 training examples, which exceeds the requirement of more than 100K training samples. To meet the requirement of more than 10K testing samples, we will perform a split on the "unsplit" configuration dataset. By using a desired ratio, such as 80% for training and 20% for testing, we will randomly select 20% of the examples from the "unsplit" dataset, resulting in a testing set with more than 10K samples. The remaining 80% of examples will be used for training. This splitting process ensures that the dataset meets the requirement of more than 10K testing samples. The dataset consists of 6 classes (anger, fear, joy, love, sadness, surprise), which satisfies the requirement of having more than 2 classes. The dataset information does not specify the number of attributes, but we will generate the 20 attributes required ourselves (excluding the class label).

Overall therefore, based on the provided information, the dataset meets the requirements. It has more than 100K training samples, more than 10K testing samples, more than 2 classes, and it is possible to generate at least 20 attributes (excluding the class label) to meet the requirement. Moving forwards, in order to create or build the dataset, our first step will be to download the dataset from the source mentioned in the metadata or follow any specific instructions provided in the dataset documentation. Once we have obtained the dataset, preprocessing steps may be necessary to clean the text data. This involves handling any missing values or noisy data and ensuring consistency, accuracy, and completeness of the dataset. Furthermore, we may need to apply feature engineering techniques to represent the text data in a format suitable for the machine learning algorithms that will be implemented in the project. These techniques will help in transforming the raw text into meaningful and informative features that can be effectively utilized by the algorithms. By performing these preprocessing and feature engineering steps, we aim to enhance the quality and usability of the dataset for our emotion classification project.

We will use different popular python libraries to build our models. For text processing we will use NLTK (Natural Language Toolkit), this library will be used to preprocess the text data like tokenization and removing stop words or filler words like “the”, “in”, “is” that dont have too much impact on sentiment of the text and it will help decrease the complexity of the model. Then we will use other Machine Learning Libraries to build our models once the data has been preprocessed such as Scikit-learn, TensorFlow or PyTorch.

The hyperparameters will vary depending on the model that we are trying to build. For each of the models we will try to finetune our parameters through trial and error to find the optimal parameters. Below is a list of the models that we will implement and the main hyperparameters that we will finetune for each model.

1. Decision Tree:
    a. Maximum Depth of the tree: Not limiting this parameter can lead to overfitting.
    b. Minimum Number of Samples to split: reduces the size of the tree by preventing the creation of nodes.
    c. Minimum Samples to be a Leaf: This parameter also helps in controlling the model complexity.
2. Naive Bayes Classifier
    a. Alpha: This is the value of the probability that you assign when the feature was not in the training data to avoid a zero probability condition.
3. Logistic Regression
    a. Solver: Algorithm used for optimization.
    b. Penalty: The norm used in the penalization, this help feature selection and model complexity.
4. Support Vector Machine (SVM)
    a. Penalty Parameter of the error term.
    b. Kernel: What type of kernel to use. (linear, sigmoid, poly).

5. Multi-layer Perceptrons (MLP_
    a. Hidden Layer Size: the amount of hidden layers we want, this will increase complexity.
    b. Activation Function: What activation function we want, sigmoid, ReLu or others.
    c. Solver: What optimizer we are going to use (SGD, ADAM, RMSProp).
    d. Learning Rate: Initial learning rate of the model, this control the step size of each iteration.
    e. Maximum Iterations: Has a ceiling of how many iterations to do if the model does not converge..

Many steps need to be taken in order to develop a useful and correct classifier. First, the dataset needs to be partitioned into testing, training, and validation sets. Splitting the dataset will enable the model to have an unbiased evaluation of unseen data. We will then use metrics such as accuracy, precision, and f1 score to assess the performance of our classifier. Accuracy measures the ratio between the number of correct precisions and the total number of predictions. Precision will be used to account for true positives and false negatives. F1 score will account for precision and recall, and is therefore useful when comparing models. These measurements will illustrate how well our classifier predicts the emotional sentiments from a set of texts.

Project Timeline

Task	**Due Date**	**Responsibility**
Pre-Process Data	April 7th	Everyone
Multi-Layer Perceptrons (MLP)	April 14th	Carlos
Logistic Regression	April 14th	Samarth
Naive Bayes Classifiers	April 14th	Jeronime
Project Progress Report	April 16th	Everyone
Support Vector Machine	April 26th	Everyone
Decision Tree	April 26th	Everyone
Project Final Report Draft	April 28th	Everyone
Project Final Report	April 30th	Everyone