Documentation for the Project

Objective of the project

This is the project which was made for an internship application. The objective of the project was to create a restful API which is a text classifier. The text classifier should take up to 150 characters of a text snippet and will tell the user what sentiment does the text display.

Scope of the Project

The scope of the project involves building a Flask API for sentiment analysis based on a pre-trained XGBoost model. The API takes text input, preprocesses it by tokenizing, removing stop words, and lemmatizing using Spacy, and then applies a profanity filter. If the text contains profanity, it is labelled as "Hate Speech"; otherwise, it is classified into sentiment classes using the XGBoost model.

Additional features include a profanity filter to censor offensive language and the ability to detect hate speech based on the presence of profanity. The API will be tested in a Flask development environment. The project aims to provide a reliable sentiment analysis tool with a profanity filter for various applications.

Dataset used

This dataset has been provided by the Company for the assessment purpose. The dataset used has 18000 records and two columns: The first one is the text column that has the text snippet and the second column has the corresponding label that depicts the sentiment of the text snippet. It was in xlsx format and the text are in raw form i.e., Natural language.

Processes involved in the project

1. **Filtering the Profanity and Duplicates**
   This is the first phase of the project and the following activities were performed during this phase:
   - The dataset was imported and examined to check for null values.
   - Then the duplicate records were identified and dropped to remove redundancy from the dataset.
   - Functions were created to remove the profanity (bad words) and replace the bad words with "*****" and clean the text of unwanted characters.
   - The functions were applied on the dataset then the dataset was transformed as per our requirements.
   - The new dataset was saved in the form of a csv file.

   Packages used: Pandas for dataset manipulation, Spacy for NLP, Profanity-filter to remove bad words, Profanity filter to remove hate speech, Re for regular expressions

2. Creating the Classifier Model

This is the second phase of the project where the classification was built. The following steps were performed:

- The filtered dataset was imported and examined for flaws.
- After that the dataset was cleaned and formatted accordingly.
- **Bonus Task:** We used Spacy Model package Doc to check for sentences with grammatical errors.
- NLP techniques like tokenization, stop word removal, lemmatization etc were used for preprocessing the text data.
- After the pre-processing, TFIDF vectorization was performed and at maximum 5000 features were kept in order to maximise efficiency.
- The dependent variable was encoded using label encoder.
- The dataset was split into X and y which is Independent and Dependent variables.
- Then 80% of the dataset was split for training and the 20% for testing.
- At first, conventional machine learning models like Multinomial naïve bayes, Support Vector Classification was used which gave an average accuracy of 82% between the three models.
- Then we moved on to ensemble learning models where we used XGBoost, Gradient Boosting and Random Forest which gave which gave an average accuracy of 87% between the three models.
- Since XGBoost gave the highest accuracy of 88.42%, it was decided that this model will be used for hyperparameter tuning to improve accuracy.
- Keeping in mind the constraints of time and processing resources only a third of the predictor variables were picked for passing into the model.
- A halving grid search was performed for hyperparameter tuning which worked within the constraints and gave the hyperparameters.
- The hyperparameters were applied on the model which shockingly gave an accuracy of 88.45% which could be called a negligible improvement.
- Owing to time and constraints of processing power, it was decided that this model should be finalised as an accuracy of approximately 90% worked well.
- Going further for neural networks would have delayed the project and would have consumed a significant amount of scarce computing resources, hence this plan was scrapped.
- The model was pickled and passed on to the next phase.

Packages used: Pandas for dataset manipulation, Spacy for NLP,sklearn for machine learning models and metrics, Tensorflow keras for helping with early stopping, nltk for NLP processes, XGboost for final model, Re for regular expressions, pickle for storing/opening model

3. Developing the REST Flask API

This is the final phase of the project where the Restful Flask API was built and tested. The following steps were performed:

- The pickled model was loaded from a file named best_xgb_classifier.pkl.
- Define a mapping between numerical labels and emotions.
- Set up a profanity filter using the profanity_filter library.
- Loaded the spaCy language model en_core_web_sm.
- Created a Flask app and an API.
- Defined a resource for sentiment analysis.
- Implemented the post() method of the resource to handle sentiment analysis requests.
- Checked the text length and filter the text using spaCy.
- Applied profanity filtering and check for hate speech.
- Made a prediction using the loaded model and map the numerical label to emotion.
- The program returned the result as JSON.
- Added the resource to the API with the desired endpoint.
- Ran the Flask app on port 5000 in debug mode and copied the link for postman testing.
- Trial runs were conducted for the API.

Packages used: Flask for web development purposes, Spacy for NLP, Profanity filter to remove hate speech, pickle for storing/opening model