Report

A. Part 1

I chose to use the "stopwords_removal_nltk" flavor of comments. These are lemmatized comments with stopwords removed, taking morphological analysis into consideration and removing stopwords that might not be very helpful for classification. Lemmatized comments with no stopwords are then transformed into tf-idf (term frequency–inverse document frequency) vectors, quantifying comments for training models.

Sentiment scores of comments were also added as one of the features, next to the tf-idf vector.

For training models that could handle multiclass classification natively, I dropped all data that has not been labeled as either Bug, Feature, Rating, or UserExperience, and combined all the rest into a dataset where each comment corresponds to one of the four classes. This process results in having a combined dataset of 1183 comments for training and 222 comments for testing, where test set was set to 15% of the total dataset.

A Random Forest, and a Logistic Regression classifier (with "multinomial") were trained and tuned to classify multiclass classification natively for their popularity, strong potential to tune hyperparameters, and, most importantly, their inherent ability to do multiclass classification.

For training models that could only handle binary class classification, e.g. SVM classifier, Logistic Regression, etc., I trained a model for each of the four classes with their own dataset. During prediction, each of the four models will calculate a probability that this comments belongs to the model's corresponding class and the class with highest probability will be picked.

This One-vs-All approach was trained and tuned with SVM, Logistic Regression classifier (without "multinomial") for their popularity, strong potential to tune hyperparameters, and, most importantly, their nature of only being able to do binary class classification.

Since Scikit-learn actually provides a way to do One-vs-All method for models that might only support binary class classification via confidence level, I also trained a Logistic Regression with `mutli_class` set to "ovr" to compare with my own implementation of One-vs-All. For this model, data of four classes must be combined into one dataset, so it was fed with the combined dataset as other inherently multiclass capable models.

For this One-vs-All approach, the same test set was used as other inherently multiclass capable models.

B. Part 2

All models below have the same setting: tf-idf + lemmatization - stopwords + sentiment

Scenario	Bug F1 Score	Feature F1 Score	Rating F1 Score	UserExperience F1 Score	Accuracy
Random Forest (multiclass)	0.6047	0.3143	0.6349	0.6387	0.5766
Logistic Regression (multinomial)	0.6154	0.4250	0.6116	0.6349	0.5856
Logistic Regression (ovr sklearn)	0.5932	0.4051	0.5950	0.6190	0.5676
SVC (ovr)	0.5421	0.5149	0.5600	0.5766	0.5495
Logistic Regression (ovr)	0.3830	0.4098	0.6202	0.4444	0.4730

C. Part 3

The model with the best performance was Logistic Regression with multinomial enabled, which reached 58% accuracy. This is prabably due to the multinomial loss it tries to minimize across the entire probability distribution.

The model with the worst performance was Logistic Regression with my own One-vs-All method, which only reached 47% accuracy. This is probably due to that my implementation of One-vs-All by default will normalize the data before feeding into the model. If I remove the step of normalization, performance would dramatically increase as shown below.

Scenario	Bug F1 Score	Feature F1 Score	Rating F1 Score	UserExperience F1 Score	Accuracy
Logistic Regression (ovr, no normalization)	0.5800	0.5745	0.6029	0.5614	0.5811

However, it seems like this trick cannot be applied to SVM classifier. It will occassionally cause SVM classifier to take forever to train.

To conclude, One-vs-All method gives binary class classifiers the ability to do multiclass classification and is able to reach similar performance, though there maybe caveat here and there.

To train the best model, one thing I will probably try would be Recurrent neural networks, which has a strong potential in natural language processing, to extract features. For models selections, in the next run I will probably calibrate my models with sigmoid function so that models that do not support probability prediction, LinearSVC for example, could also be used.