Part 1:

In the reference paper, the authors found that a binary classifier yields better results than multiclass classifiers. Moreover, Naive Bayes algorithm was reported to be a suitable choice that reaches satisfactory results in a short time. Finally, adding metadata like sentiment score and rating could increase the classifier performances.

So I decided to use the reference work's findings as the guide to train the models and make technical decisions. Two algorithms, Naive Bayes and SVM, are used to train binary classifiers. SVM is used to compare how a different algorithm could have affected the outcome. With each algorithm, two sets of settings are used. One set only includes title and comment, and another includes title, comment, rating, and sentiment score. This setup is meant to see how much can metadata contribute to the overall performance.

The comments used in training come from the "stopwords_removal_lemmatization" field in the datasets.This field is chosen so that trivial words and word forms can have minimal impact on the training. The string is combined with the "title" field, and tf-idf is applied to convert each document into a feature vector. Tf-idf is set to choose a maximum of 400 features so that the models will not overfit. Additional values, "sentiScore_pos" and "rating", are added to the feature vector in settings that consider metadata.

Since I am using the one-to-all strategy to train 4 binary classifiers in each model/setting combination, the 4 datasets are combined into 1. The combined dataset is labeled 4 different ways to train each binary classifier. For instance, when training for "bug report", all the comments label as "Bug" are encoded into "1", while the others labelled as "Not_Bug", "Feature", "Not_Feature", "UserExperience", "Not_UserExperience", "Rating" and "Not_Rating" are encoded into "0". Random undersampling is used on instances with label "0" to ensure the dataset balance.

Lastly, the training set is fed into Multinomial Naive Bayes algorithm, which is a widely used Naive Bayes variant when training feature vectors, and also into SVM. Randomized search is used to tune hyperparameters in the SVM algorithm to find the best estimator.

Part 2:

Performance for the models

		Bug report	Features	User Experience	Ratings	Overall Accuracy
Model/Setting		F1-Score	F1-Score	F1-score	F1-score	
SVM	BOW lemmatization - stopwords	0.73	0.79	0.76	0.65	0.75
SVM	BOW + lemmatization – stopwords + sentiment + rating	0.70	0.72	0.74	0.55	0.70
Naïve Bayes	BOW + lemmatization - stopwords	0.73	0.71	0.68	0.62	0.70
Naïve Bayes	BOW +lemmatization – stopwords + sentiment + rating	0.76	0.75	0.75	0.63	0.72

Part 3:

a. What is the model/setting that had the best performance? Which is the factor that has contributed the most for its performance?

The SVM/BOW+lemmatization-stopwords had the best performance. It might be the algorithm that contributes to the performance.

b. What is the model/setting that had the worst performance? Which is the factor that has contributed the most for its poor performance?

The SVM/BOW + lemmatization – stopwords + sentiment + rating combination and Naive Bayes/BOW + lemmatization - stopwords combination has around the same performance.

c. You have to recommend one model to your company. Which one would you recommend and why?

Based on the experiment results, I would recommend the SVM/BOW+lemmatization-stopwords combination. This combination yields the highest f1 score and accuracy for all 4 classifiers, which will be more useful in classifying different comments automatically. Plus, not using any metadata is also a plus, since the company does not need to collect sentiment analysis data, which might be more hard to come by. But I would caution the users of the models that some manual checking is still required, especially when rating classifiers still only yield a 0.65 f1 score. As more manual checkings are done, additional comments can be added into the training set to improve classifier performance.

d. What can you conclude about the performance of your models? In other words, what did you learn from analyzing the performance of these different models?

When not including any metadata, the accuracy and f1 score for each classifier did not reach the same level as those in the reference work. The effect of adding metadata also did not increase the performance much as reported by the reference work. One reason for the performance difference might be how data are preprocessed. Although I used the “stopwords_removal_lemmatization”, “rating”, and “sentiScore_pose” fields provided in the datasets, the reference work might have used a different way to do BOW. In this assignment, I used the TfidfVectorizer class to convert documents into feature vectors. This class provides many parameters, where I mostly go with the default ones. Perhaps the accuracy and f1 score would increase if different parameters for TfidfVectorizer are used. Finally, this might suggest that, when doing text classification, the data preprocessing section is critical to the final performance. Using different algorithms or adding metadata could not automatically lead to a better classifier if data is not preprocessed properly.