CS 412 - Machine Learning
2023 - 2024 Spring Term
-
Project Report - “MLCoders”

Berke Ayyıldızlı 31018 - Beyza Balota 31232 -
Nur Deren Sakin 29078 - Ali Eren Şahin 27831 - Kerem Tuğrul Enünlü 30750

1. Introduction

On the term project, our main goal is to develop a machine learning model that can accurately assign severity scores to bug descriptions that are present on the test dataset. The respected model of xgboost, is selected from a list of candidates, on the basis of highest average macro precision, as it was asked to base our predictions on. The xgboost is a proven algorithm that specializes in supervised learning, which uses boosting techniques.

To achieve the maximum amount of prediction and macro precision rate, we did a handful of operations on the data that we had in our hands. These operations involve preprocessing text data, which uses Natural Language Toolkit (NLTK) lemmatization, vectorization using TF-IDF and tokenization by alphabetical characters. In the model training part, we specifically selected our parameters by utilizing a diverse grid search that found the optimal parameter values for the sake of maximizing the results.

To upload the predictions we have found, we made the trained model to predict the labels on the test dataset, and then write to a new csv file.

2. Problem Description

The main task of the project is to, as explained in the documentation, correctly predict and assign the severity score of the bug descriptions. The classes we have on our hands are as follows:

1. Enhancement
2. Minor
3. Normal
4. Major
5. Blocker
6. Critical

As from the description, the problem can be explained as a multi-class classification problem. In this type of problems, we know that the trained model must assign one of discrete labels to the respected descriptions.

Let D represent the dataset of bug reports, where each bug report i in D consists of:

$\mathrm{X_i}$: The bug description and bug type.

y_i: The severity score associated with the bug report.

The dataset D is : $D = \{(\mathrm{X_i},y_i),(\mathrm{X_2},y_2),\ldots,(\mathrm{X_n},y_n)\}$

where n is the number of bugs in the data.

The objective is to learn a function f such that: $f(\mathrm{X_i}) \approx y_i$

where f is a classifier trained on the dataset D to map each bug description $\mathrm{X_i}$ to its corresponding severity score y_i.

3. Methodology

For the coding part, we have used various machine learning techniques to solve this problem. Our approach included the following steps:

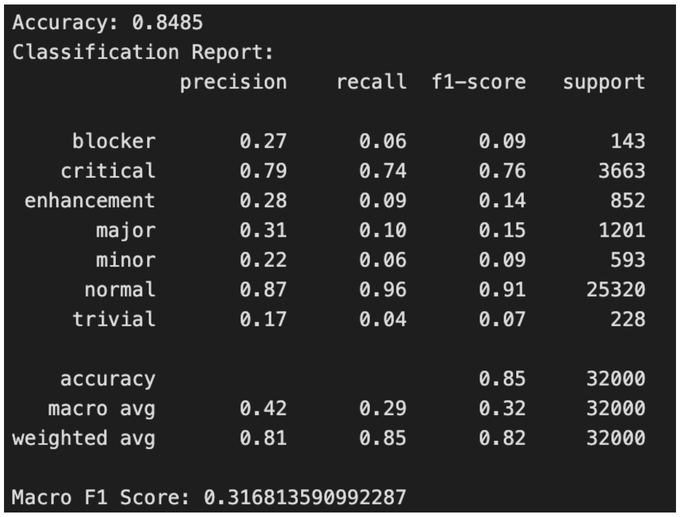
For the data processing part, we have first loaded the “bugs-train.csv”. On the lemmatization step, we got help from the NLTK library and encoded the labels accordingly.
On the feature extraction part, we have used TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. This helped us to understand which words are more important than others by looking at their count of appearance. The new features are combined with the severity labels.

Later, for the training, we first started by splitting the data into training and testing sets on a ⅕ ratio. The divided training set is used on the XGBoost algorithm to train the model with the optimized parameters.

After the training, we made the model prediction by working on the test set. The results and the new labels of the test dataset are saved on a separate csv file, which is submitted to Kaggle.

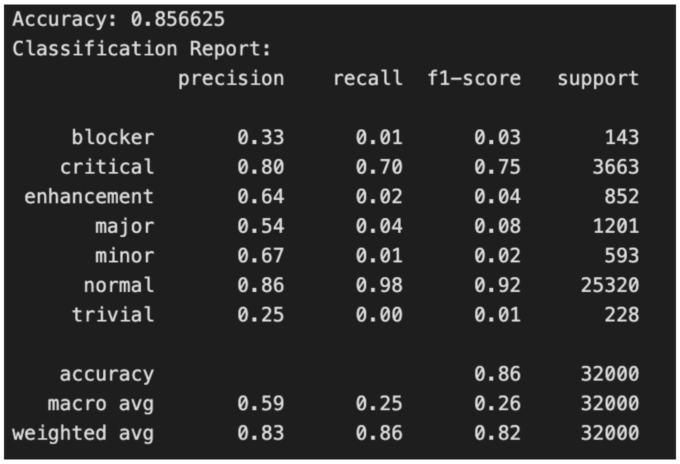
4. Results

Before giving the results of the XCBoost algorithm, we first tried many different models with many different results, including questions such as if selecting TF-IDF vectorization’s maximum number of features as 500 is enough, or how does our algorithm perform compared to other algorithms. To understand the effects of our selected parameters, we need to check the findings of other methods. For example a neural network initialization using 20 epochs gave 0.8485 accuracy and 0.42 macro average. Similarly, logistic regression came back with the same results.



```
Accuracy: 0.8485
Classification Report:
              precision    recall  f1-score   support

     blocker       0.27      0.06      0.09       143
    critical       0.79      0.74      0.76      3663
 enhancement       0.28      0.09      0.14       852
       major       0.31      0.10      0.15      1201
       minor       0.22      0.06      0.09       593
      normal       0.87      0.96      0.91     25320
     trivial       0.17      0.04      0.07       228

    accuracy                           0.85     32000
   macro avg       0.42      0.29      0.32     32000
weighted avg       0.81      0.85      0.82     32000

Macro F1 Score: 0.316813590992287
```

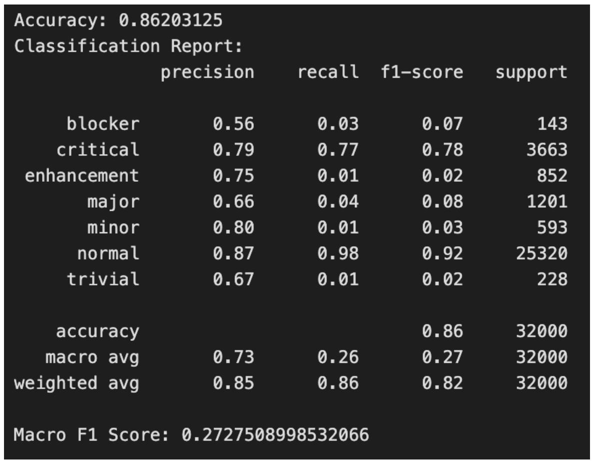
Figure 1: Results of Neural Network



```
Accuracy: 0.856625
Classification Report:
              precision    recall  f1-score   support

     blocker       0.33      0.01      0.03       143
    critical       0.80      0.70      0.75      3663
 enhancement       0.64      0.02      0.04       852
       major       0.54      0.04      0.08      1201
       minor       0.67      0.01      0.02       593
      normal       0.86      0.98      0.92     25320
     trivial       0.25      0.00      0.01       228

    accuracy                           0.86     32000
   macro avg       0.59      0.25      0.26     32000
weighted avg       0.83      0.86      0.82     32000
```

Figure 2: Results of Logistic Regression

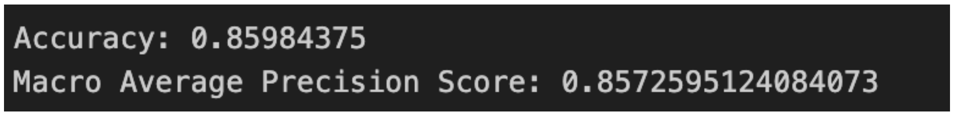
Later, in addition to logistic regression and random forest methods, we continued with XCBoost, but this time, not utilizing the NLTK dataset.



```
Accuracy: 0.86203125
Classification Report:
              precision    recall  f1-score   support

     blocker       0.56      0.03      0.07       143
    critical       0.79      0.77      0.78      3663
 enhancement       0.75      0.01      0.02       852
       major       0.66      0.04      0.08      1201
       minor       0.80      0.01      0.03       593
      normal       0.87      0.98      0.92     25320
     trivial       0.67      0.01      0.02       228

    accuracy                           0.86     32000
   macro avg       0.73      0.26      0.27     32000
weighted avg       0.85      0.86      0.82     32000

Macro F1 Score: 0.2727508998532066
```

Figure 3: Results of XCBoost without NLTK

The figure 4 represents our current model's testing, as it can be seen from the result, it gives the highest macro average precision score among all.



```
Accuracy: 0.85984375
Macro Average Precision Score: 0.8572595124084073
```

Figure 4: Last results from our current model

5. Discussion

The results from our experiments indicate that traditional machine learning models can effectively classify bug severity levels. Among the models tested, Gradient Boosting performed the best, achieving the highest precision, recall, and F1-score. This model demonstrated a robust ability to handle the complexities of bug descriptions and severity levels.

One of the key challenges we encountered was dealing with the imbalanced dataset, where some severity levels had significantly fewer instances than others. This imbalance posed difficulties in achieving high precision and recall for the minority classes. Despite these challenges, the XGBoost model managed to balance performance among differences in severity levels, as reflected in the final score.

We explored different feature extraction techniques, and found that using TF-IDF provides a good balance between capturing important textual information, as well as maintaining efficiency. Additionally, feature engineering played a crucial role in enhancing model performance by implementing relevant characteristics from the bug descriptions.

The traditional machine learning approaches, while not as advanced as deep learning models, provided interpretable results and required less computational resources. This makes them suitable for scenarios where computational efficiency and interpretability are prioritized.

In future work, we could explore other traditional models like Support Vector Machines (SVM) and Naive Bayes, and consider ensemble methods to further improve performance. Additionally, enhancing data preprocessing techniques and incorporating more sophisticated feature selection methods could lead to better results.

Overall, our findings suggest that traditional machine learning models, particularly Gradient Boosting, are capable of effectively classifying bug severity levels, even with the inherent challenges of an imbalanced dataset.

6. Appendix

a. Contributions

Model Application: Berke Ayyıldızlı
Feature Extraction: Beyza Balota
Data Processing: Kerem Tuğrul Enünlü
Evaluation and Adjustment: Ali Eren Şahin
Report Writing: Nur Deren Sakin