Udacity MLE Nanodegree - Capstone Report

Zhaoyi Hou (Joey)

1. Introduction

As a critical component of the developer community, Stack Overflow (www.stackoverflow.com) detailedly documented countless communication among developers via question-answering activities. Not only did it solve critical problems the developers had, but it also paved the way for all the future developers to save their time on debugging and, therefore, to make a bigger impact with the help from the pioneers. Because of that, Stack Overflow has become a "go-to" place and even a culture for the developers. Therefore, it is intuitive to investigate the quality of the questions on the Stack Overflow, given the impact it has on the entire developer community. If there were too many low-quality questions, developers would be overwhelmed and find it difficult to navigate a clear error case that they have.

To address this problem, I proposed a machine learning classification model to assess the question quality of a given question text. To build this model, I incorporate an XGBoost Model and train it on previously labeled Stack Overflow question texts with official labels from the website. With this model, accurate classification of Stack Overflow questions is possible and thus people can use it as guidance when posting questions.

2. Dataset

2.1 Overview

The dataset I use for this project comes from Kaggle[1]. It consists of 60,000 pieces of question text data and corresponding labels. All of those questions were posted between 2016 and 2020.

2.1 Data Fields

The dataset consists of five different data fields and a label (see Table 1).

[1] https://www.kaggle.com/imoore/60k-stack-overflow-questions-with-quality-rate

Attribute	Id	Title	Body	Tags	CreationDate	Y (*question label*)
Type	Int	String	String (*HTML Text*)	String	String	String

Table 1. Dataset Fields

2.2 Labels

As mentioned above, this is a multi-class dataset with three different labels (definition given by Kaggle[2]):

- "HQ": High-quality posts with a total of 30+ scores and without a single edit;
- "LQ_EDIT": Low-quality posts with a negative score, and multiple community edits. However, they remain open after those changes;
- "LQ_CLOSE": Low-quality posts that were closed by the community without a single edit.

Unexpectedly, the labels are perfectly, evenly distributed, as shown in Figure 1. Therefore, manual balancing of class weight is not needed during training.

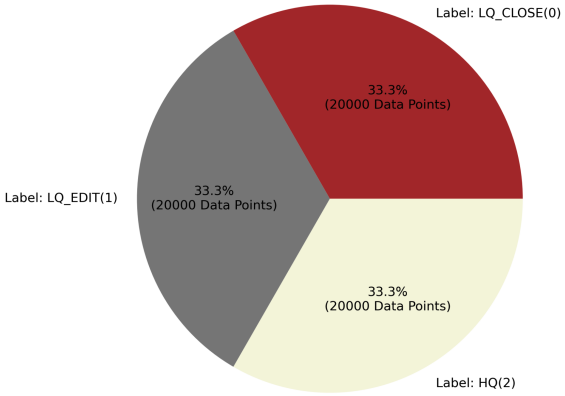

Figure 1. Label Distribution of the Dataset

2.3 Other Details

2.3.1 Frequent Word Analysis

[2] https://www.kaggle.com/imoore/60k-stack-overflow-questions-with-quality-rate

After filtering out all the stopwords by NLTK[3], I analyze the frequency distribution of the common words across three different classes. As shown in Figure 2, most of the top-frequency words across three different classes are the same. However, there is a notable difference between LQ_Close and the other two classes that the word "would" is used less and the word "lt" and "gt" are used much more frequently. In other words, the appearance of certain words and phrases might be useful in classification.

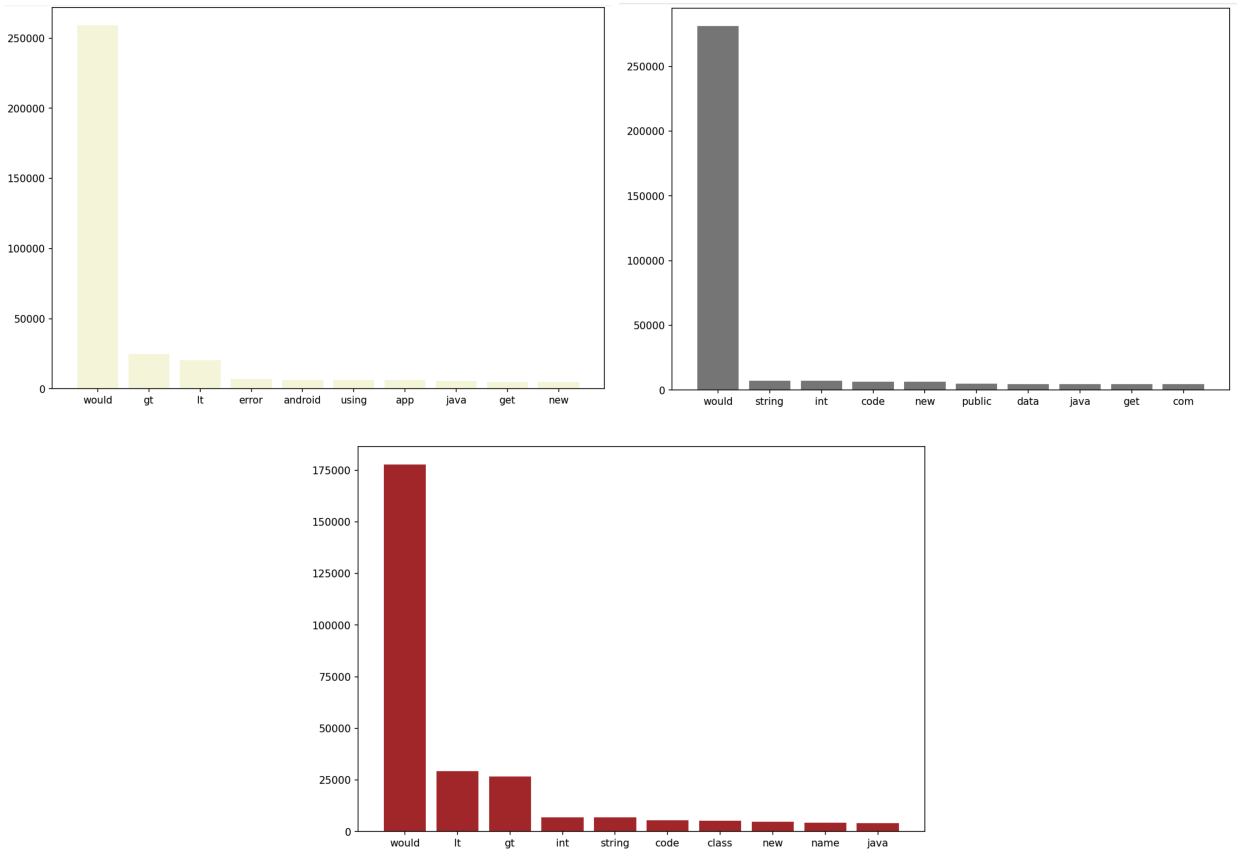

Figure 2. Frequent Words Distribution

Top-left: HQ(2); Top-right: LQ_Edit(1); Bottom: LQ_Close(0)

2.3.2 Question Length Distribution

In addition to the common words analysis, I also calculate and analyze the total length of the title and content of each question across three different classes. As shown in Figure 3, the HQ(2) questions tend

[3] Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc."

to have a higher value for total length (keep in mind that the total number of questions for each label is the same!)

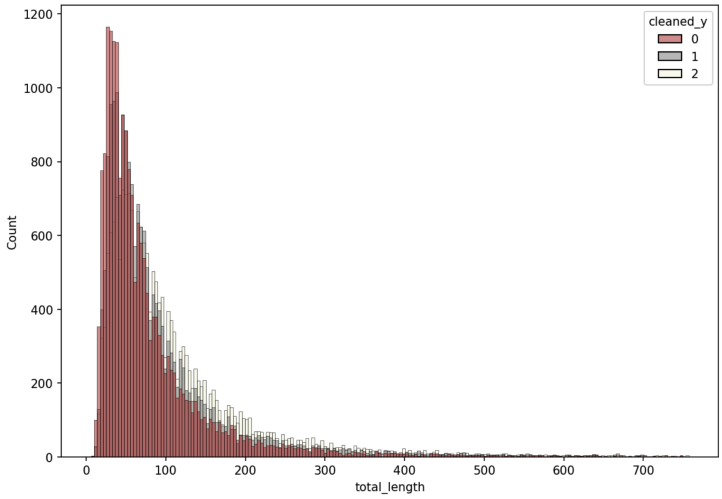

Figure 3. Label Distribution of the Dataset

3. Techniques and Algorithms

3.1 Pre-processing

Given the cleanness of the given dataset, the pre-processing step is straightforward. All the texts were first brought down to lower case characters; after that, commonly used "short-term" are replaced by the full version of them; then, commonly used stopwords are removed from the text; and lastly, the title and body of every question are combined together into one single field, which will be the main focus of the modeling steps.

3.2 Baseline Model

The baseline model for this question is defined as followed:

- Input Feature: title length, content length
- Model: Logistic Regression from Scikit Learn[4]

3.3 Tokenize & Encoding

[4] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Tokenization of the cleaned questions is done through the Python NLTK framework. After tokenization, a bag of word feature representation is constructed. The sentences (in form of "lists of words") are converted into "count vectors" using Scikit-Learn CountVectorizer.

3.4 XGBoost Model

For this classification task, I use the standard XGBoost Model built-in in AWS Sagemaker[5]. XGBoost is a decision-tree-based ensemble Machine Learning algorithm that handles classification tasks. As a widely used boosting method, it has the advantage of the boosting algorithm: efficiency, which means the training time is relatively short. It also employs gradient descent for optimization, which improves its classification performance.

3.5 Hyperparameters

For hyperparameters, I conduct a hyperparameter tuning task with AWS SageMaker and picked the best combination available. The detailed hyperparameters used in the final model are shown in Table 2.

Hyperparameter	**Value**
max_depth	11
eta	0.2622
gamma	1.0375
min_child_weight	3
subsample	0.8544
objective	multi:softmax
num_class	3
early_stopping_rounds	10
num_round	500

Table 2. XGBoost Hyperparameters

[5] https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost.html

3.6 Evaluation Metric

As mentioned in section **2.2**, the label for this dataset is equally distributed. Therefore, there is no need to accommodate for any one of the labels and it is appropriate to use subset accuracy (or, exact match ratio) as the evaluation metric. The formula for this metric is defined as followed. In other words, it measures the proportion of the prediction labels that exactly match the target labels.

$$ExactMatchRatio = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = Z_i)$$

4. Result Analysis

4.1 Accuracy and Performance Distribution

The overall accuracy of the XGBoost model on the test set is **72.22%**, compared to the **38.62%** from the baseline model.

The model's performance on data points from different classes is also evaluated. As shown in Table 3, the XGBoost model does a great job of classifying the label for HQ (2), i.e. high-quality questions, compared to other labels. In other words, if the XGBoost model predicts that a question is of high quality, it is highly likely that it does have a high quality.

Class Label	LQ_CLOSE (0)	LQ_EDIT (1)	HQ (2)	Overall
Baseline	63.10%	31.24%	21.52%	38.62%
XGBoost	65.54%	71.50%	**79.64%**	**72.22%**

Table 3. Model Performance on Different Subsets of Data (Test Set)

4.3 One-VS-All Performance

In some cases, there is a need to perform One-VS-All classification. For example, simply predicting if a question will be closed or will be considered as a high-quality question. In these One-VS-All

classification cases, the XGBoost Model gives even more impressive performances. As shown in Table 4, the prediction accuracy for high-quality questions versus non-high-quality is nearly 87%. In other words, when using this model in the One-VS-All task, it can identify high-quality questions accurately.

Class Label	LQ_CLOSE (0) V.S. All	HQ (2) V.S. All
Baseline	52.82%	64.86%
XGBoost	76.84%	**86.92%**

Table 4. Model Performance on Different One-VS-All Task (Test Set)

5. Related Work

In this field, multiple researchers have made contributions in the past. For example, a group of researchers in Arizona State University have conducted a research back in 2017[6]. In this research, they built a machine learning with linguistic-based features and achieved around 60% accuracy on a 4-label prediction task.

6. Conclusion & Future Work

In conclusion, although the focus for this project is to use AWS SageMaker to solve a real problem, using tokenization, word count encoding and XGBoost model can successfully classify questions from Stack Overflow into different quality groups with high accuracy. Due to the time limit of the Nanodegree program, only minimum feature engineering and hyperparameter tuning are conducted. However, compared to the simple baseline model based on question and title length, my simple method yields significant improvements on the baseline model. In the future, it would be promising to use a more complex NLP model such as Transformer and to incorporate more hand-crafted features such as lexical diversity in the model. In that way, built-in softwares that give suggestions to people posting questions on Stack Overflow would be possible.

[6] Assessing Question Quality using NLP (2017)

Work Cited

1. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
2. Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc."
3. Kristopher J. Kopp, Amy M. Johnson, Scott A. Crossley, and Danielle S. McNamara, Assessing Question Quality using NLP (2017), 18th International Conference on Artificial Intelligence in Education (pp. 523-527). Wuhan, China