Stack Overflow Question Quality

CS750 Final Project Report

Ryan Tobin, James Westbrook

Introduction

Stack Overflow is an online forum for asking and answering questions about programming and computer science. With millions of users visiting the site every day, it is no stretch to say that it has become a huge part of the programming community [1]. However, a large reason for Stack Overflow's continued success is its emphasis on asking good questions [2], as these are most likely to be answerable and received well by the community. But of course, poorly framed questions regularly slip through the cracks, and this can create a nuissance for forum regulars and moderators, as answering bad questions is typically a much harder task than answering good ones.

Currently, there is a flagging system in place, where questions deemed unfit can be flagged for further review. We assert that automation could make the flag system more efficient since a question could be automatically flagged before any human even sees it (and then thinks about it and types up a response to it). Our goal in this paper is to explore machine learning models to classify Stack Overflow questions as **good** and **bad** with the potential for automated moderation/flagging in the back of our heads. We believe such a model, if it exists, could reduce the workload of moderators and active community members.

Though our primary goal is prediction, we are also interested in seeing which features of a question are the best predictors of whether it is a good one. In this sense, we can verify Stack Overflow's guidelines for asking good questions.

Related Works

A survey of past works on similar topics will give us a hand in defining our features and target.

First, Stack Overflow's official guidelines on asking questions claim that using code snippets with a minimal reproducible example, including relevant tags, using a helpful title, and using proper grammer improves the quality and answerability of questions [2].

Duijn *et al.* analyzed code snippets in determining quality questions. They defined **good** and **bad** questions using two metrics: score and number of edits. Notably, they filtered out questions with zero score, as these questions were deemed 'not popular enough to get a reliable verdict'. In their analysis, they were able to achieve a classification accuracy of 79.8% using the score metric and 81.2% using the edits metric, both with random forest classification (note: the only features they used were based on code snippets). They found worse results for decision tree and logistic regression. Finally, they found that code/text ratio is a strong predictor of question quality [3].

Chua and Banerjee explored the answerability of Stack Overflow questions to develop what they called the Quest-for-Answer framework. For us, the most interesting result they found is that downvoted posts correlated positively with being answered [4].

Methodology

Definition of "Good" Question

The notion of "question quality" is admittedly vague. A more tangible and measurable target is *community reception*, which can be measured by Stack Overflow's builtin *score* metric, and we will assume that community reception is a good heuristic for quality. We labeled questions with scores > 0 as **good**, and ones with score < 0 as **bad**. Questions with a score of 0 were removed from the dataset since we determined the true quality of these is ambiguous. Duijn *et al.* reported a similar interpretation in their paper [3].

Data

To obtain our data, we used Stack Overflow's StackAPI library to gather data on questions between January 1, 2021 and March 2, 2021. We processed this data into features while filtering out questions with a score of 0, resulting in 13,407 datapoints.

One flaw in our dataset is that 82.24% of our sampled questions have $score > 0$, and due to this we decided to undersample **good** questions during training.

We allowed only for features which are available at the time of posting the question, so metadata such as view count was not used, but the owner's acceptance rate and reputation were.

Classification

To obtain our results, we evaluated the cross-validation score of the following machine learning algorithms on our training data:

- Logistic Regression,

- Support Vector Classifier with linear, polynomial, and RBF kernels,
- Extreme Gradient Boosted Trees.

The algorithm with the greatest cross-validation score was then trained on the training set and evaluated on the test set, and the resulting classification accuracy is our primary result. Referring back to Duijn *et. al*'s study, their 81.2% classification error will be our minimal bar for success, and 90% would be incredible given our primitive features.

Secondarily, we analyzed feature importance with three methods:

- Correlation coefficients with score,
- Normalized logistic regression coefficients,
- Presence of each feature in the XGBoost trees.

Results

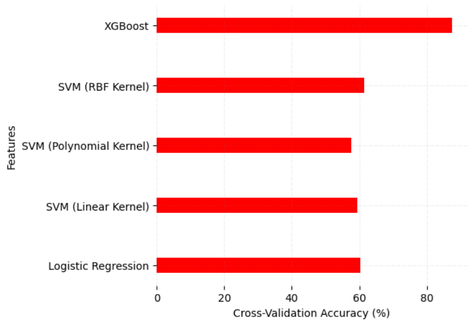


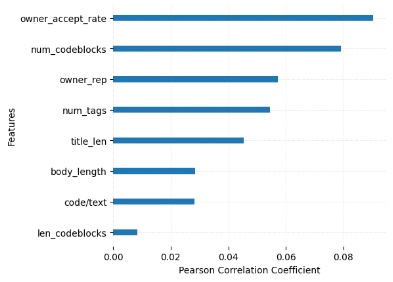
Figure 1: Cross-Validation Accuracies

Figure 1 summarizes the cross-validation scores of each method. Extreme Gradient Boosted trees performed by far the best of the 5 methods, and thus we evaluated its performance on the test set. We reported a 75.21% classification accuracy on the test set. Additionally, of the 925 **bad** predictions, 356 were truly **bad**, giving us a true negative rate of 0.385.

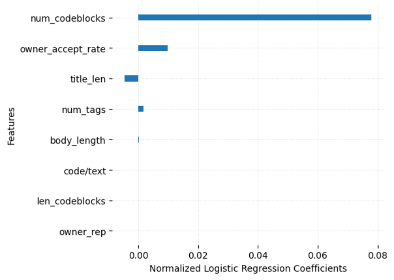
Figure 2a shows the Pearson Correlation Coefficients of each feature with raw score. We can see that the owner's acceptance rate, and number of codeblocks have some correlation, however no singular feature appears to have very high correlation.

Furthermore, we trained logistic regression on normalized data (all features were transformed to have unit mean and variance), and compared the sizes of the coefficients in Figure 2b. We see that number of codeblocks is by far the most important, and nothing else is close.

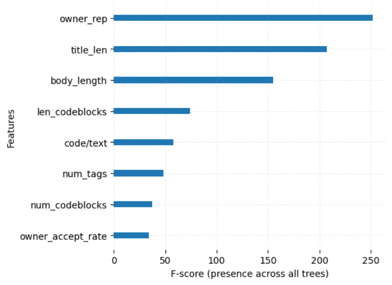
Finally, we computed the presence of each feature in our XGBoost tree as seen in Figure 2c. Here, owner reputation, title length, and body length were the three most used features.



(a) Pearson Correlation Coefficients between Features and Score



(b) Normalized Logistic Regression Coefficients



(c) XGBoost Presence

Figure 2: Feature Importance

Discussion/Conclusions

Our prediction model performed below our expected standards, both in classification accuracy and TNR. We do not believe that our model in its current form would be useful in moderating Stack Overflow, however we are hopeful that further improvements could push it to our standards. For example, we avoided using natural language processing due to time constraints, and there are certainly interactions between features that we missed. Our model is near-limitlessly scalable with features.

However, even if our approach is wrong, there is still some hope for alternatives. Perhaps score is the wrong target to be using (we are only assuming it's a good one), in which case number of edits is a replaceable metric. Or perhaps having a target is wrong, and an unsupervised approach is correct. Even KMeans could work well for binary clustering, and may allow for a richer notion of question quality than our one-dimensional *score* metric.

Finally, we are unable to report consistent results on feature importance. No features seemed to be great predictors of question quality, though some showed relevance. We are unable to verify Stack Overflows advice, although this is partly due to our features being primitive.

References

1. David C (2023) Stack overflow growth and usage statistics (2023)

2. How do I ask a good question?

3. Duijn M, Kucera A, Bacchelli A (2015) Quality questions need quality code: Classifying code fragments on stack overflow

4. Chua A, Banerjee S (2021) Answers or no answers: Studying question answerability in stack overflow. Journal of Information Science 41(5):720–731

5. Correa D, Sureka A (2013) Fit or unfit: Analysis and prediction of 'closed questions' on stack overflow

6. Flag posts

7. Yang J, Hauff C, Bozzon A, Houben G-J (2014) Asking the right question in collaborative q&a systems