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 for the site 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 the possibility of classifying Stack Overflow questions as **good** and **bad** using machine learning, 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 we are primarily focused on 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 score > 0 as **good**, and ones with score < 0 as **bad**. Questions with a score of 0 were removed from the dataset since these are uninterpretable as **good** or **bad** due to low popularity or being highly controversial. 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 basic features while filtering out questions with a score of 0, resulting in 13,407 datapoints. Examples of features include number of tags, number of codeblocks, and number of non-whitespace characters in the body. Metadata such as view count was available, but we restricted ourselves to metadata that would be available at the time of posting.

It is noteworthy that 82.24% of our sampled questions have *score* > 0, causing a major imbalance in our classes. However, as we expect the distribution of Stack Overflow questions to roughly follow this in general, we did not account for this during training.

Classification

To obtain our results, we evaluated the 5-fold cross-validation score of the following machine learning algorithms on our training data:

- Logistic Regression,
- Support Vector Machines 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. Based on the class distributions, we expect a significantly higher classification accuracy than 82.24%. We place the cutoff for success at 95% classification accuracy.

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

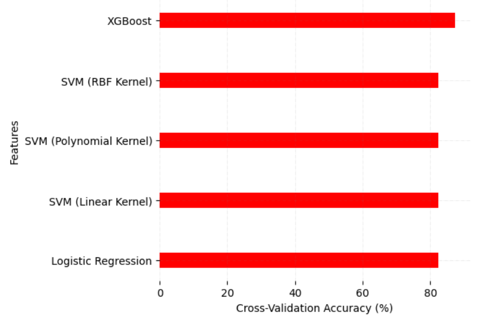


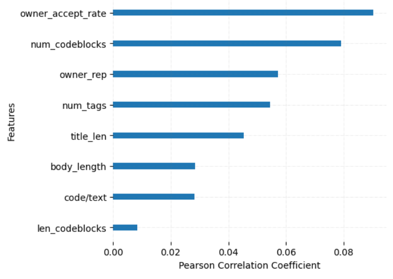
Figure 1: Cross-Validation Accuracies

Figure 1 summarizes the cross-validation scores of each method. The performance of each algorithm is strikingly similar, and this is likely due to the class imbalance - everything besides XGBoost is simply predicting **good** every time. Since XGBoost performs the best, we evaluated its performance on the test set and reported a classification accuracy of 87.26%.

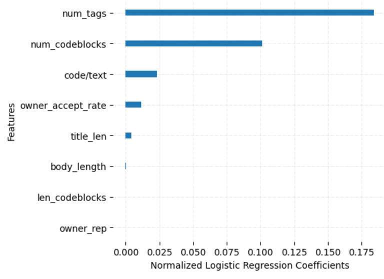
Figure 2a shows the Pearson Correlation Coefficients of each feature with raw score. We can see that the owner's acceptance rate, and number of codeblocks are most highly correlated, however no singular feature appears to have very high correlation.

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 tags and number of codeblocks are considered most highly, and the rest of the features were not used very much.

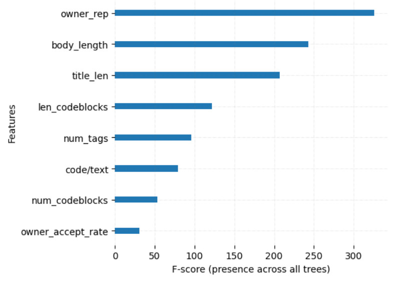
Finally, we computed the presence of each feature in our XGBoost tree as seen in Figure 2c. Here, owner reputation, body length, and title length were the three most used features.



(a) Pearson Correlation Coefficients between Features and Score



(b) Normalized Logistic Regression Coefficients



(c) XGBoost Presence

Figure 2: Feature Importance

Discussion/Conclusions

Though a classification accuracy of 87.26% is somewhat high, it is only 5% better than if we had just predicted **good** every time. As such, 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 - our features and target are extremely flexible. Extending our scope beyond this project, we would like to incorporate polynomial features, interaction features, and natural langauge processing (in that order) into our model. Our 87.26% result should be regarded as a *baseline* performance on this problem.

There are also alternative approaches to this problem. A cost-sensitive approach is more appropriate than what we have done (see [5]), as false-positives and false-negatives are not quite weighed equally in the context of automated moderation. Additionally, an unsupervised clustering approach could yield a richer notion of quality than we can characterize.

Finally, we are unable to report consistent results on feature importance - no features seemed to be great predictors of question quality, but some were relevant. We are unable to verify Stack Overflow's guidelines, although our features are at best simplified representations of the guidelines in the first place.

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. Ling CX, Sheng VS (2008) Cost-sensitive learning and the class imbalance problem. Encyclopedia of Machine Learning

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

7. Flag posts

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