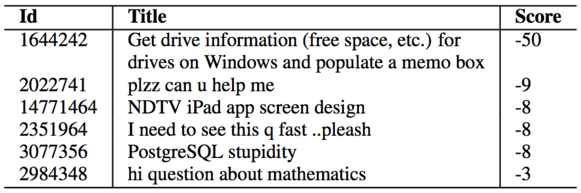
Stack Overflow Query Outcome Prediction

Robbie Jones[1] and David Lin[2]

Abstract— **Stack Overflow's core mission is to create an online encyclopedia for all programming knowledge. In order to ensure quality content in the face of rapid growth, community moderators frequently close low quality questions, often asked by newcomers. In order to alleviate moderator burden and ease newcomers' transition, we devise two classifiers to predict 1) whether a question will be closed and if close 2) its reason for closure. We train our models using logistic regression, SVMs, and boosting before selecting the optimal classifier. We found that the adaptive boosting algorithm best classified whether a question would be closed, whereas lasso-regulated logistic regression best classified the reason for closure. Our next steps to improve our classifiers include using word vectors, splicing the data by time period, and extracting more features from code segments.**

I. INTRODUCTION

Stack Overflow (SO) is the largest online question and answer forum for professional and novice programmers, boasting over 10 million questions and 4 million users in its database [5]. Despite its remarkable growth, new users increasingly report instances of being harassed by older users for not conforming to SO community standards or asking "low quality" questions [5]. In many cases, community moderators will close the question, citing one of five reasons (as defined by SO): duplicate, off topic, too broad, primarily opinion-based, or unclear. Many of these questions are rightfully closed (see Table 1), as they reflect low effort on the asker's part and do not contribute to meaningful online discourse [6].



Id	Title	Score
1644242	Get drive information (free space, etc.) for drives on Windows and populate a memo box	-50
2022741	plzz can u help me	-9
14771464	NDTV iPad app screen design	-8
2351964	I need to see this q fast ..pleash	-8
3077356	PostgreSQL stupidity	-8
2984348	hi question about mathematics	-3

Fig. 1. Examples of closed questions. Figure courtesy of Correa and Sureka 2013 [3].

In other cases, questions that reflect a strong effort but a lack of familiarity with SO guidelines are still met with the same level of hostility. A 2013 study found that 77 percent of users only ask one question and 65 percent of users only answer one question [9], a direct consequence of this hostile culture and lack of community engagement for newcomers. Ultimately, this cultural epidemic stems from a conflict of interest between older users trying to prevent the dilution of quality content and unfamiliar newcomers trying to get their programming questions answered [5]. Thus, our goal is to bridge this community divide and remedy the user experience for new users.

Given historical query data, we aim to develop two develop two separate models to 1) predict whether question will be closed and if closed, 2) its most likely reason for closure. For each classification problem, we establish a baseline model with L1 (lasso) and L2 (ridge) logistic regression. We then apply support vector machines (SVMs) and adaptive boosting to identify the optimal classifier in each case. Through automated quality control, we hope this tool will help new users get acclimated to the site, improve question quality, and reduce moderator burden.

II. RELATED WORK

Most literature in this field has been inspired by the Kaggle competition sponsored by Stack Overflow to predict closed questions [6]. Using the Kaggle data set, Lezina and Kuznetsov use a Vowpal Wabbit implementation of multi-class logistic regression to classify a question as answered or a particular reason for closure [8]. However, their model significantly overpredicts the number of closed questions. In order to overcome this obstacle, we will break the problem into two separate classifiers - the prediction of closure and the the reason for closure - in order to better capture the trends in each.

Correa and Sureka focuses on the first of these problems [2][3]. They expand the feature set with more user information and build a binary classifier to predict whether a question will be closed. However, since the publication of all these studies, the categories for closed questions have been redefined and the percentage of closed questions has increased [8]. Our study accounts for both these changes and uses a more detailed data set than Kaggle.

III. DATA AND FEATURE EXTRACTION

A. Dataset

We use the StackExchange Data Explorer [10] to directly pull historical questions, answers and user profiles from the SO database. In order to maintain a balanced data set, we include 40,000 answered questions and 40,000 closed questions (10,000 off topic, 10,000 too broad, 10,000 primarily opinion-based, 10,000 unclear). We excluded duplicate questions since classification would depend on previous questions and the time of publication. Our data set also excludes all closed questions categorized by old standards. In addition to

[1] rmjones@stanford.edu, Stanford University, Department of Computer Science

[2] dlin15@stanford.edu, Stanford University, Department of Physics

the question body, title and tags, we also include abundant data on each asker's user profile and question history.

B. Text Representation

In order to prevent the text from drowning out other features, we wanted to first streamline our text data. We removed all stop words, or frequently appearing words like ”the” that don't have strong standalone meaning. Then, we converted the text to lowercase, stripped all punctuation and code segments, and removed all special symbols such as links, emails and numbers. Since we wanted all words with different suffixes but the same root to be treated the same, we used a stemming algorithm to truncate the suffixes off each word. Finally, we represent the remaining text using a bag-of-words model that keeps track of the frequency of occurrence of each word, disregarding word order. Each individual word count is then fed into our classifier as a unique feature.

C. Feature Extraction

The bulk of our features can be come from the text, user profile, and query metadata. Specifically, our primary features included unigrams, bigrams, word count body, word count title, number of lines in the body, and account age. Before putting them into the feature vector, we normalized each set of feature values to one.

Since most closed questions are usually asked by new users, we expect user information like account age as well as question metadata to be the most important features for our first classifier[5]. The bag-of-words model and remaining textual features will likely play more of a role in distinguishing between reasons for closure.

IV. METHODS

A. One vs. Rest

We tackle the question of whether a question is closed with standard binary classification algorithms. However, to predict the reason for closure in our second classifier, we use the one-vs-rest strategy of breaking apart a multiclass problem into multiple binary classification problems. One classifier is created for each class, where each classifier labels members of k-th class positive and member of all other classes negative. We can predict the label of a new training example by selecting the classifier with the highest confidence and reporting that classifier's positive label.

The one-vs-rest method is computationally efficient compared to alternatives like one-vs-one. However this method has two shortcomings. First, the method ignores all interclass dependence. Second, even though we balanced our data set, there will be a constant factor more negative examples than positive examples for each binary classifier.

B. Baseline Models

Logistic regression creates a decision boundary by minimizing the cost function used to fit parameters. Thus, our baseline models for each classifier are L1 (lasso) and L2 (ridge) logistic regression. Both methods add a regularization term to combat over fitting during training:

$$L1 : R(\theta) = ||\theta||_1 = \sum_{i=1}^{n} |\theta_i| \tag{1}$$

$$L2 : R(\theta) = ||\theta||_2 = \sum_{i=1}^{n} \theta_i^2 \tag{2}$$

The L1 regularization tends to set many parameters to zero, thereby eliminating redundant features and functioning as a natural feature selection algorithm. L2 regularization does not have implicit feature selection but is better suited for learning algorithms outside logistic regression. Naturally, L1 tends to outperform L2 regularization when only a subset of the features correlate to the label.

C. SVM

Support Vector Machines (SVMs) work by creating a hyperplane that maximizes the functional margin. Kernel methods are often used with SVMs to detect nonlinear boundaries not easily picked up by logistic regression. In this project, we use three kernels: the linear kernel, polynomial kernel, and the radial basis function (RBF) kernel. They are defined as follows:

Linear:

$$K(x_i, x_j) = x_i^T x_j \tag{3}$$

Polynomial:

$$K(x_i, x_j) = (x_i^T x_j + c)^d \tag{4}$$

(where c is the free parameter and d is the exponent. d is usually less than 4 to avoid overfitting)

RBF:

$$K(x_i, x_j) = \exp(-\frac{|x_i - x_j|_2^2}{2\sigma^2}) \tag{5}$$

where x_i and x_j represent training examples. Compared to other kernels, the linear kernel is fastest and easiest to optimize. Linear kernels are often useful in text classification problems where there is a high volume features and training data. Such problems are often linearly separable, in which case the linear kernel is sufficient. If the problem is not linearly separable, using the RBF or polynomial kernel to map our data to a higher dimensional space can often improve predictive power.

D. Adaptive Boosting

Adaptive boosting (AdaBoost) works by combining weak learners to optimize performance. Given a training set, every example is initially weighted equally. During each iteration, we train a binary classifier on the weighted examples. For each classifier, we define

$$\alpha_i = \frac{1}{2} ln \frac{1 - \epsilon_i}{\epsilon_i} \tag{6}$$

in terms of the error ϵ_i.

We then up-weight all the misclassified examples and down-weight all the correctly classified examples by a constant factor proportional to the accuracy of the classifier. After iterating through every weak learner, the final hypothesis is

$$H(x) = \text{Sign}(\sum_{i=1}^{m} \alpha_i h_i(x)) \tag{7}$$

E. Performance Metrics

For each type of question outcome, we randomly sample 70 percent for the training set and leave the remainder for the test set. Our performance can be evaluated in two ways: accuracy (ACC) and area under curve (AUC). ACC is defined as one minus the misclassification error on the test set.

$$ACC = 1 - \frac{\text{Number of Misclassified Labels}}{\text{Number of Total Labels}} \tag{8}$$

The second performance metric is AUC, which refers to the error under the receiver operating characteristic (ROC) curve, a plot of true positive rate against false positive rate. A strong classifier will have a high true positive rate and therefore produce an AUC near one. On the other hand, a score of 0.5 implies that the classifier is no better than a random classifier. For our first binary classification problem (closed or not closed), the AUC score is intuitive. However, we must generalize this metric for our multiclass prediction problem of determining reasons for closure. Each of these k classifiers in our one-vs-rest strategy has its own ROC curve, so to summarize the multi-class results we use the micro-average (averages individual true positive and negative scores) and macro-average (averages individual errors).

The micro-average is biased towards correct predictions for larger classes whereas the macro-average is biased towards correct predictions for smaller classes. Since we balanced the size of each question category, we can use either, so we opt to report the AUC micro-averages for this study.

V. CLASSIFIER 1: TO CLOSE OR NOT TO CLOSE

A. Baseline Models

To determine whether a question should be closed, we first establish two baseline models with regularized logistic regression. We partitioned our data set (40,000 closed, 40,000 not closed) into a randomly selected training set with 70 percent of the examples and a test set with the remaining 30 percent. Our results are summarized as follows:

TABLE I
BASELINE RESULTS

Model	Train ACC	Test ACC
Lasso (L1) Logistic	0.761	0.734
Ridge (L2) Logistic	0.997	0.711

Both models have an ACC score about 0.7, although the L2 regression shows more sign of overfitting. This overfitting likely stems from the abundance of features coming from our bag-of-words model. Since the L1 model has implicit feature selection, it uses fewer parameters and is less susceptible to noise.

Although the L2 norm usually gets the nod over L1 in practice, there is a distinct case in which L1 finds more optimal solutions than L2: sparse feature spaces.Because of the square in the L2 norm, larger weights are punished more heavily than smaller ones, and thus the L2 norm seeks to shrink larger weights over smaller ones. Conversely, the L1 norm is just as likely to shrink smaller weights as it is larger weights, so it is more apt to shrinking weights very close to zero, thus finding more optimal solutions in a sparse feature space. Word features (i.e. counts or existence) are one of the canonical examples of a sparse feature space, and thus a good explanation for the advantage of the L1 norm in this problem.

B. Intermediate Models

Our two primary sets of intermediate models were SVMs and boosting. The accuracy scores of our three SVMs and adaptive boosting are summarized as follows:

TABLE II
INTERMEDIATE RESULTS

Model	Train ACC	Test ACC
Linear SVM	0.999	0.601
Polynomial SVM	0.507	0.487
RBF SVM	0.503	0.505
AdaBoost	0.847	0.766

Surprisingly, all the SVM kernels performed worse than the baseline. Out of the SVMs, the linear SVM had the best test accuracy, which implies that our problem is linearly separable. Since linear SVM is the most comparable to logistic regression, it makes sense that the linear SVM had the closest ACC to the logistic models among the SVM models. Generally, speaking the RBF and polynomial SVMs are more prone to over-fitting. One explanation is that our data set has high levels of noise, making it difficult to separate with a hyper plane. By far the strongest intermediate model is adaptive boosting, the only model to outperform the baseline. There are a couple possible explanations. Unlike SVMs where we have to select a kernel, adaptive boosting is a non-parametric algorithm, so we make very few assumptions on our data. While computationally less inefficient, adaptive boosting is capable of detecting non-intuitive decision boundaries while avoiding overfitting, which is very uncommon.

C. Final Model

Based on the accuracy scores, we select boosting as our final model. We graph the ROC curve for each of the classifiers in the one-vs-rest model as well as the microaverage and macroaverage.

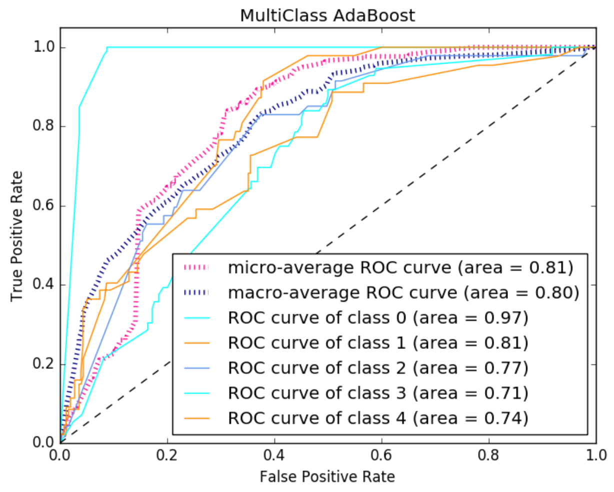


Fig. 2. ROC Curve for L1 logistic regression

Since our dataset was balanced, our micro- and macro-averages were nearly the same, as expected. Given a microaverage AUC of .81, we expect our classifier to be robust to new data.

VI. CLASSIFIER 2: REASON FOR CLOSURE

A. Baseline Models

As before, we run two regularized logistic regression models to establish a baseline. We tackle the issue of multi-class predictions using a one-vs-rest strategy. Our results are summarized as follows:

TABLE III
BASELINE RESULTS

Model	Train ACC	Test ACC
Lasso (L1) Logistic	0.633	0.570
Ridge (L2) Logistic	0.992	0.506

As expected, with more classes both baselines are lower than that those of our previous binary classification problem. Like before, the L1 regularization model proves relatively robust, whereas the L2 model is more susceptible to noise, as evidenced by its dramatic drop in test error.

Although the L2 norm usually gets the nod over L1 in practice, there is a distinct case in which L1 finds more optimal solutions than L2: sparse feature spaces. Because of the square in the L2 norm, larger weights are punished more heavily than smaller ones, and thus the L2 norm seeks to shrink larger weights over smaller ones. Conversely, the L1 norm is just as likely to shrink smaller weights as it is larger weights, so it is more apt to shrinking weights very close to zero, thus finding more optimal solutions in a sparse feature space. Word features (i.e. counts or existence) are one of the canonical examples of a sparse feature space, and thus a good explanation for the advantage of the L1 norm in this problem.

B. Intermediate Models

Continuing with the one-vs-rest strategy, we also try classification with SVMs and boosting.

TABLE IV
INTERMEDIATE RESULTS

Model	Train ACC	Test ACC
Linear SVM	0.998	0.344
Polynomial SVM	0.338	0.334
RBF SVM	0.322	0.333
AdaBoost	0.551	0.541

Like before, boosting proved to be the strongest intermediate model and the SVMs proved underwhelming. Surprisingly, none of these four models could surpass our baseline. This result again implies that we made a couple of incorrect assumptions about our data. Since the StackOverflow community is driven by human moderators, its very plausible that the moderators themselves close questions that should not be closed, effectively mislabeling examples in our training set. Another assumption we made when we used the one-vs-rest strategy of multi-classification was the independence of classes. However, since many newcomers face the same multitude of obstalces when joining StackOverflow, there is very likely a degree of interdependence among classes. Using a one-vs-one strategy could potentially improve this problem.

C. Final Model

Ultimately, the simplest model was the best one. Hence, we show our L1 regression results below.

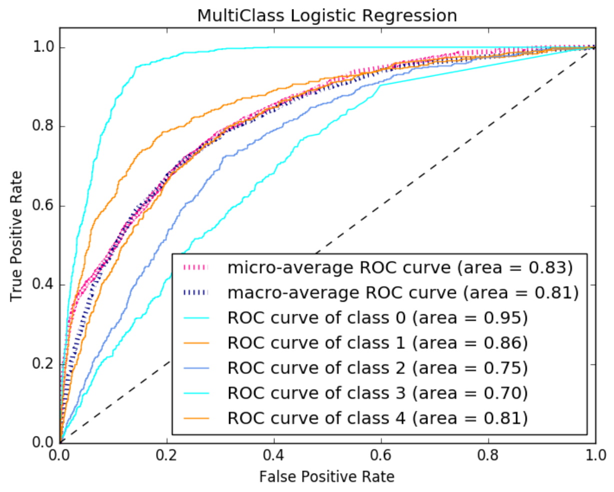


Fig. 3. ROC Curve for L1 logistic regression

VII. CONCLUSIONS

For both classification problems, the regularized logistic regression models proved to be more robust than other predictive models. When predicting whether a question would be closed the AdaBoost algorithm slightly outperformed the baseline. However, for the multiclass problem of determining the reason for closure, none of our intermediate models could outperform the baseline.

For future work, we plan to employ the following changes to improve our models:

- including accounting for the temporal aspect of the question. The percentage of closed questions increases annually, so we can make our model more robust by accounting for this changing community culture. Specifically, we can splice our data into different time periods and training the model such that features are weighted differently by time.
- adding a feature set for code segments. The presence of code often indicates that the asker made an attempt to solve their problem before asking although too much code can be a sign of an overly-localized question.
- using word vectors to represent text. Unlike the bag-of words-model, implementations of Word2vec retains context and similar words from the text. Word vectors are often useful for training neural networks.

ACKNOWLEDGMENT

Thank you to Nihit Desai for advice and feedback on the project.

REFERENCES

[1] C. Asawa, M. Gomez, and V. Mehta, ”Finding Your Way, Courtesy of Machine Learning,” in Stanford CS 229, 2016. [Online]. Available: http://cs229.stanford.edu/proj2016spr/report/055.pdf.

[2] D. Correa and A. Sureka, ”Fit or Unfit : Analysis and Prediction of Closed Questions on Stack Overflow,” in Proceedings of ACM, 2013. [Online]. Available: https://arxiv.org/pdf/1307.7291v1.pdf.

[3] D. Correa and A. Sureka, ”Chaff from the Wheat : Characterization and Modeling of Deleted Questions on Stack Overflow,” in Proceedings of ACM, 2013. [Online]. Available: https://arxiv.org/pdf/1401.0480v1.pdf

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research, vol. 12, pp. 28252830, 2011.

[5] J. Slegers, ”The Decline of Stack Overflow,” in HackerNoon, 2014. [Online]. Available: https://hackernoon.com/the-decline-of-stack-overflow-7cb69faa575d.vndi5cnpq.

[6] G. Lezina and A. Kuznetsov, ”Predict Closed Questions on StackOverflow,” CEUR, vol. 1031, 2013.

[7] ”Predict closed questions on stack overflow,” in Kaggle, 2013. [Online]. Available: https://www.kaggle.com/c/predict-closed-questions-on-stack-overflow.

[8] L. Ponzanelli, A. Mocci, A. Bacchelli, M. Lanza, and D. Fullerton, ”Improving Low Quality Stack Overflow Post Detection,” in University of Lugano, 2014. [Online]. Available: http://www.inf.usi.ch/phd/ponzanelli/profile/publications/2014e/Ponz2014e.pdf.

[9] S. Wang and D. Lo, ”An Empirical Study on Developer Interactions in StackOverflow,” in Singapore Management University, 2013. [Online]. Available: http://www.mysmu.edu/faculty/lxjiang/papers/sac13stackoverflow.pdf.

[10] ”Stack Exchange Data Explorer,”. [Online]. Available: https://data.stackexchange.com/stackoverflow/query/new.