# Analysis and Prediction of Question Quality on Stack Overflow

Ruohe Zhou
rzhou73@wisc.edu

Annabelle Wan
cwan22@wisc.edu

Tsz Yau Iris Chow
tchow7@wisc.edu

## Abstract

*Stack Overflow is one of the most popular Community driven Question Answering (CQA) websites for programmers. However, the quality of the questions raised concerns because low-quality questions can be misleading or make it time-consuming to obtain information. Often in blogs, there are questions that are irrelevant, off-topic, duplicate of other questions or contain wrong information. Recognizing high quality questions can improve user experience and could help Stack Overflow to provide better service. In our project, we focus on exploring the relationship between different features of questions and their qualities, as well as making predictions based on the features. To build this model, data from Kaggle containing 60 thousand observations of posts from Stack Overflow was collected. These posts are prelabeled as belonging to one of the three categories: High-quality posts without a single edit, low-quality posts with multiple community edits, and low-quality posts that were closed by the community without a single edit. Features such as question title's length, body's length, and context of title were collected to help label the quality of the post. Regular expression and natural language toolkit were used to remove punctuation, stopwords, as well as other irrelevant terms like brackets. TfidfVectorizer[2] was applied to transform text to feature vectors for model fitting. To ensure a robust model, different ensemble methods were tested and grid search was used to determine the best parameter and to achieve higher accuracy. The best model was logistic regression. Its overall accuracy is 81.983 %*

## 1. Introduction

Community driven Question Answering (CQA) websites such as Stack Overflow provide a platform to ask questions and obtain answers from other users that have experienced similar problems or are expertise in the field. From another point of view, CQA website is an interface that follows a crowd sourced model that allows experts to share their knowledge on a large scale based on a variety of topics. Stack Exchange layout networks of CQA websites that run multiple forums on various topics.

Stack Overflow is so far the most popular and the first Stack Exchange website for programmers. Stack Overflow is an Q&A website where users can ask computer programming related questions. Users on Stack Overflow can tag a question to indicate relevant topics of the question, users are allowed to edit questions and answers. Moreover, users can express their opinions on how helpful the post is by voting. This community based voting process helps Stack Overflow maintain the quality of posts on their platform. A question will be "closed" by moderators or experienced users if it is low-quality. All of those help maintain the quality of the posts to a reasonable degree and help eliminate low-quality posts on this huge collaborative platform.[3]

Despite this user interaction, it is a challenge to ensure all questions are answered or are helpful. Over the years there has been exponential growth in numbers of Stack Overflow users. According to Statistics from 2021, Stack Overflow has over 14 million registered users, and has received over 21 million questions and 31 million answers. [12] Moderators are not able to closely monitor every question due to tremendous workload. As a result, there are a lot of unanswered questions. Up to 29 percent of questions on Stack Overflow are left unanswered.[11] The reason behind the unanswered questions is not because users are not able to view the questions, but rather, it is because the questions are deemed not relevant or helpful.

As a CQA service, Stack Overflow should strive to have as many questions answered as possible. Understanding the factors that contribute to a post being labeled as high quality is extremely relevant to making sure questions have the best chance of being answered. Having almost 30 percent of questions go unanswered is evidence that there is room for improvement.

Therefore, in this project, we investigate factors that affect the quality of questions and try to label the quality of questions. Overall, we perform analysis on question title and content. We use Natural Language Processing to distinguish words that determine the quality of the post. In addition, we include length of title and content of question as our predictors. Besides performing textual analysis using natural language processing, we compare the performances of different models in categorizing the questions as having high or low quality.

## 2. Related Work

On account of the popularity of CQA websites and the importance to maintain quality of posts, evaluation and prediction of question quality has attracted researchers' attention. There are studies on question quality, deleted questions and features that determine question quality of Stack Overflow.

Different studies consider aspects and use methods that are different from our method to predict the quality of the questions. In Baltadzhieva and Chrupała's work[1], they use Ridge regression models to study the effects of each individual term to predict the quality of question and also to predict the probability of question getting answered. They conclude that terms expressing, among others, excitement, negative experience or terms regarding exceptions are related to high quality posts and posts containing spelling errors or off-topic or containing interjections are related to low quality posts. Furthermore, in Correa and Sureka's work[4], they are trying to predict low quality posts that have been removed. Their studies analyse and characterize "closed" questions on Stack Overflow and they include tags as one of the features besides using only the question and title. They use a machine learning classifier to predict whether the question will be closed. Correa and Sureka conclude that questions that are closed are less informative and less descriptive.

In comparison to previous studies, our project mainly focuses on textual analysis and identifies the quality of the questions. In our study we also identify words in the title and context of questions and but additionally, we include length as one of the features.

## 3. Proposed Method

### 3.1. Natural Language Processing (NLP)

NLP is a machine learning method to allow models to understand, analyze, manipulate, and potentially predict human language. [7] To apply NLP, there is an open-source package on Python named NLTK (Natural Language Toolkit). It contains basic NLP operation commands When we implement NLP, it is important to clean the data for a machine learning system to recognize meaningful patterns. Therefore, there is a function for removing punctuation and stop words. Furthermore, tokenization is part of NLP to separates text into words

### 3.2. K-Nearest Neighbors

K-nearest neighbors (KNN) is a supervised machine learning algorithm that involves learning a function from labeled input data to produce correct output when given new data points. Data points are close in terms of proximity when they are similar in KNN algorithm. KNN made approximations on new data points based on their distance on graphs

### 3.3. Random Forest

Random Forest is a supervised machine learning algorithm that developed from decision trees. It utilizes ensemble methods to combine classifiers. Random forest consists of decision trees that are trained by bagging or bootstrap aggregating. Finally, Random Forest generates outcomes based on results of decision trees by taking the average. Increasing the number of trees could improve accuracy

### 3.4. XGBoost

XGBoost is an algorithm developed based on gradient boosted decision trees and focuses on computational speed and model performance. XGBoost uses a gradient boosting framework. Boosting and bagging are used to reduce variances. In XGBoost, it uses gradient descent to optimizing the loss function:

$F_1(x) < -F_0(x) + h_1(x)$

Where $F_0$ is defined to predict the independent variable y. $h_1$ is the new model.$F_1$ is the boosted version of $F_0$

### 3.5. CatBoost

CatBoost is another algorithm developed based on decision trees. CatBoost stands for Category Boosting. It works well with categorical data. It can work with diverse data types and it yields accurate results without the extensive use of data training. The CatBoost library on Python handles categorical data automatically. [9]

### 3.6. Logistic Regression

Logistic regression is used to predict categorical variables. It uses a logistic function to model a binary dependent variable. Below is a simple form of logistic function:

$l = log_b \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

where l is the log-odds, b is the base of the logarithm, and beta are parameters of the mode, x1 and x2 are predictors of the model.

## 4. Experiments

### 4.1. Dataset

| | Id | Title | Body | Tags | CreationDate | Y |
|---|---|---|---|---|---|---|
| 0 | 34552656 | Java: Repeat Task Every Random Seconds | <p>I'm already familiar with repeating tasks e... | <java><repeat> | 2016-01-01 00:21:59 | LQ_CLOSE |
| 1 | 34553034 | Why are Java Optionals immutable? | <p>I'd like to understand why Java 8 Optionals... | <java><optional> | 2016-01-01 02:03:20 | HQ |
| 2 | 34553174 | Text Overlay Image with Darkened Opacity React... | <p>I am attempting to overlay a title over an ... | <javascript><image><overlay><react-native><opa... | 2016-01-01 02:48:24 | HQ |
| 3 | 34553318 | Why ternary operator in swift is so picky? | <p>The question is very simple, but I just cou... | <swift><operators><whitespace><ternary-operato... | 2016-01-01 03:30:17 | HQ |
| 4 | 34553755 | hide/show fab with scale animation | <p>I'm using custom floatingactionmenu. I need... | <android><material-design><floating-action-but... | 2016-01-01 05:21:48 | HQ |

Figure 1. dataset

Data from kaggle containing 60,000 observations of posts from Stack Overflow was collected. They are data from 2016-2020. [6]

Questions are classified into three categories based on their qualities:

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

The purpose of this analysis was to found out what words lead to higher quality questions. We made predictions by applying models on features separately.

We used different ensemble methods to achieve higher accuracy in our model. Natural Language Processing was used to tidy data and remove unnecessary words.[10]

#### 4.1.1 Data Processing and Visualization

The dataset includes 60 thousands post observations. Almost all of these observations included textual data that needed to be cleaned. Additionally, there were many observations containing null values. First, we dropped the rows having null values and also deleted all dividers '¡p¿' to obtain pure text of title and content of questions. After performing basic data cleaning and preprocessing, we began with exploratory data analysis by visualizing the Stack Overflow questions to get a general idea of words that people used the most using the WordCloud package.

#### 4.1.2 Natural language processing

We further clean our data by removing irrelevant words. Natural Language Processing is applied in this process. First, tokenized words to split sentences into individual words for analysis. Regular expression is used to recognize patterns of the words in tokenization. Afterwards, we worked on removing irrelevant words. We used a natural language toolkit to remove punctuation, stop words, HTML dividers, and urls. After that, we visualized the most frequent words using the WordCloud

#### 4.1.3 Model Fitting and Data Analysis

For the analysis part, we use different machine learning models. They include K-nearest Neighbors Classifier, Random Forest Classifier, XGBoost, CatBoost, Logistic Regression. We applied models on both cleaned and uncleaned dataset to compare their accuracy rates.

#### 4.1.4 Inspecting models

After that, we use lime to understand whether different models weight words differently in determining the quality of questions.

To compare the accuracy of various models, we performed the Mcnemar Test and computed the confusion matrix. We compared our two models from XGBoost and logistics regression.

### 4.2. Software

Python Jupyter notebook

### 4.3. Hardware

Laptops

## 5. Results and Discussion

### 5.1. Visualization with clean and uncleaned data


Figure 2. High_Quality without cleaning

Without removing HTML dividers and URL links in the body of the questions, we found that the high-frequency words on both plots were very similar. Therefore, we were curious to find out whether the accuracy of models will be improved after removing dividers and URL links. (As shown in Figure 2 and Figure 3)

After removing punctuation, stop words, as well as URL links, we found that the most frequently used words in high quality questions are 'code', 'using', 'use', and 'file'; and the words that show up most frequently in low quality questions are 'code', 'use', 'one', 'value',and 'function'. Some words like 'code', 'use', 'function', and 'example' show up


Figure 3. Low_Quality without cleaning

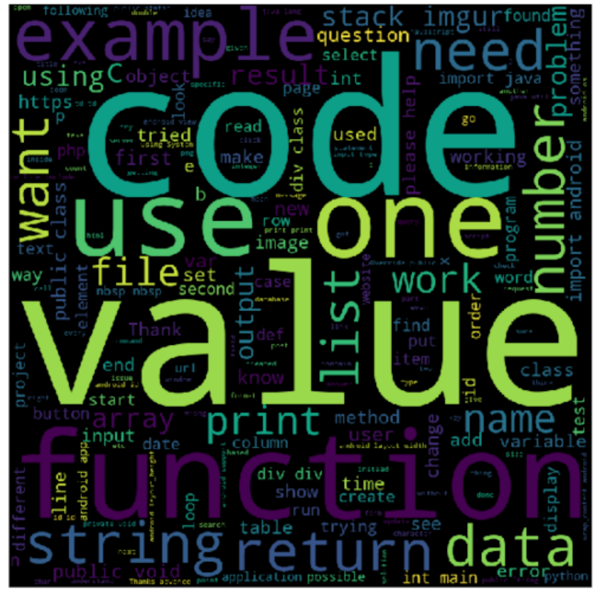


Figure 5. Low_Quality with cleaning


Figure 4. High_Quality with cleaning

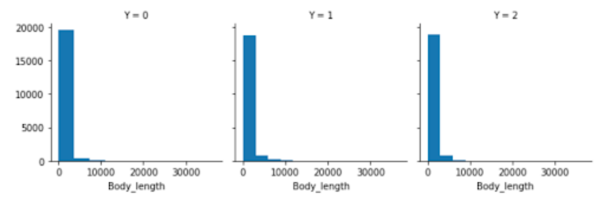


Figure 6. Section 5.2

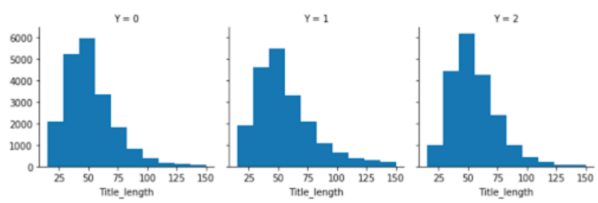


Figure 7. Section 5.2

in both levels of questions frequently, however, ‘using’ and ‘file’ are two frequently used words only in high-quality questions. (As shown in Figure 4 and Figure 5)

### 5.2. Length of body and titles and the quality of questions

In the above plots (Figure 6 and Figure 7), Y = 2 indicates high quality questions, while Y = 0 and Y = 1 indicate low quality questions. The distributions of body length in different quality categories have similar patterns, therefore, the body length is not a contributing factor in determining the quality of questions. We noticed that high quality questions have a higher number of title lengths. High quality questions have 50 to 75 words more than low quality questions on average, and the overall distribution is more concentrated, with a fewer number of title length less than 25 words or more than 100 words.

### 5.3. Model Fitting

#### 5.3.1 Check data imbalance

Before fitting models, we first check the balance of the dataset. As shown above, the three types of questions were evenly distributed in the whole dataset, and therefore the dataset is balanced.

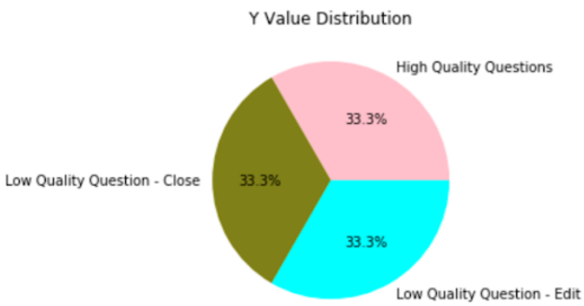


Figure 8. Section 5.3

#### 5.3.2 Fitting models without removing stop words, HTML dividers, and URL links

```
# with stopwords
from sklearn.ensemble import RandomForestClassifier
forest1 = RandomForestClassifier(n_estimators=100,
                                 random_state=123,
                                 max_depth = 50)
forest1.fit(X_train_withstop, y_train_withstop)
print(f"Train Accuracy: {forest1.score(X_train_withstop, y_train_withstop)*100:0.3f}%")
print(f"Test Accuracy: {forest1.score(X_test_withstop, y_test_withstop)*100:0.3f}%")

Train Accuracy: 93.644%
Test Accuracy: 75.817%

from xgboost import XGBClassifier
xg_classifier1 = XGBClassifier(random_state = 123)
xg_classifier1.fit(X_train_withstop, y_train_withstop)
print(f"Train Accuracy: {xg_classifier1.score(X_train_withstop, y_train_withstop)*100:0.3f}%")
print(f"Test Accuracy: {xg_classifier1.score(X_test_withstop, y_test_withstop)*100:0.3f}%")

[00:21:18] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default eval
uation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_m
etric if you'd like to restore the old behavior.
Train Accuracy: 88.990%
Test Accuracy: 82.267%

from catboost import CatBoostClassifier


boost1 = CatBoostClassifier(verbose=0)

boost1.fit(X_train_withstop, y_train_withstop)
print(f"Train Accuracy: {boost1.score(X_train_withstop, y_train_withstop)*100:0.3f}%")
print(f"Test Accuracy: {boost1.score(X_test_withstop, y_test_withstop)*100:0.3f}%")

Train Accuracy: 84.867%
Test Accuracy: 82.283%
```

Figure 9. Fitting models without removing stopwords, html dividers, and url links

#### 5.3.3 Fitting models with removing stop words, HTML dividers, and URL links

```
# without stopwords
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(n_estimators=100,
                                random_state=123,
                                max_depth = 50)
forest.fit(X_train, y_train)
print(f"Train Accuracy: {forest.score(X_train, y_train)*100:0.3f}%")
print(f"Test Accuracy: {forest.score(X_test, y_test)*100:0.3f}%")

Train Accuracy: 90.271%
Test Accuracy: 77.033%
```

Figure 10. Fitting models with removing stopwords, html dividers, and url links

Apart from random forest, models tend to perform slightly worse after removing stop words, URL links, and HTML dividers.

#### 5.3.4 Test whether models select different words in a body sentence to determine the quality of questions (Lime)

[5]

```
from xgboost import XGBClassifier
xg_classifier = XGBClassifier(random_state = 123)
xg_classifier.fit(X_train, y_train)
print(f"Train Accuracy: {xg_classifier.score(X_train, y_train)*100:0.3f}%")
print(f"Test Accuracy: {xg_classifier.score(X_test, y_test)*100:0.3f}%")

/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/xgboost/sklearn.py:1224: UserWarning:
The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warn
ing, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode
your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[23:37:54] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default eva
luation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval
_metric if you'd like to restore the old behavior.
Train Accuracy: 86.998%
Test Accuracy: 81.017%

from catboost import CatBoostClassifier


boost = CatBoostClassifier(verbose=0, random_state = 123)

boost.fit(X_train, y_train)
print(f"Train Accuracy: {boost.score(X_train, y_train)*100:0.3f}%")
print(f"Test Accuracy: {boost.score(X_test, y_test)*100:0.3f}%")

Train Accuracy: 83.921%
Test Accuracy: 81.467%

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state = 123)
lr.fit(X_train, y_train)
print(f"Train Accuracy: {lr.score(X_train, y_train)*100:0.3f}%")
print(f"Test Accuracy: {lr.score(X_test, y_test)*100:0.3f}%")

Train Accuracy: 91.629%
Test Accuracy: 81.983%
```

Figure 11. Fitting models with removing stopwords, html dividers, and url links

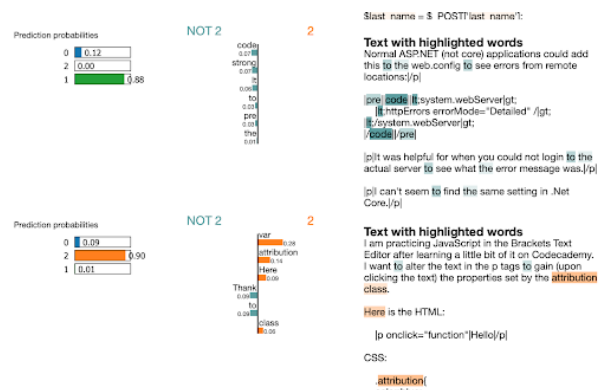

Figure 12. Random forest

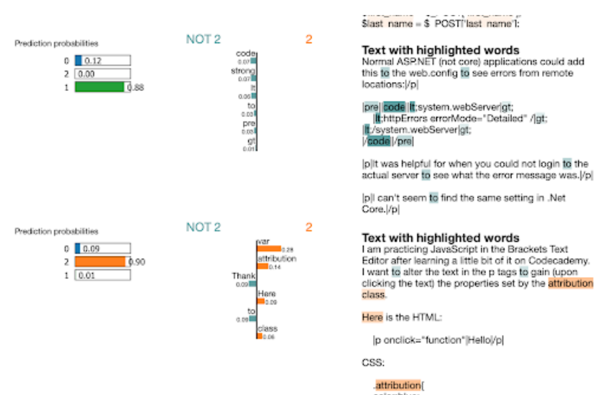

Figure 13. XGBoost

Note: 2 indicates high quality questions, and NOT 2 indicates low quality questions. We found that Random Forest and Xgboost take similar words to determine if a question is of high quality or low quality, while logistic regression selects different words.

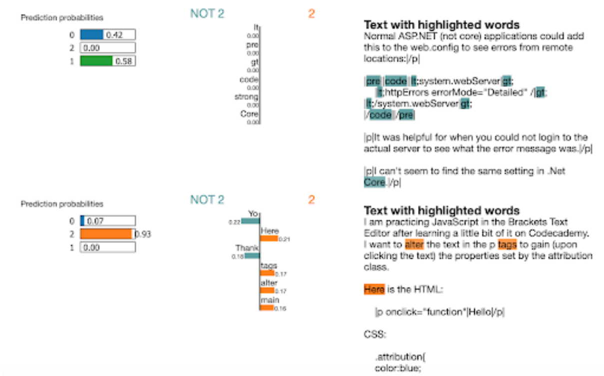


Figure 14. Logistic Regression

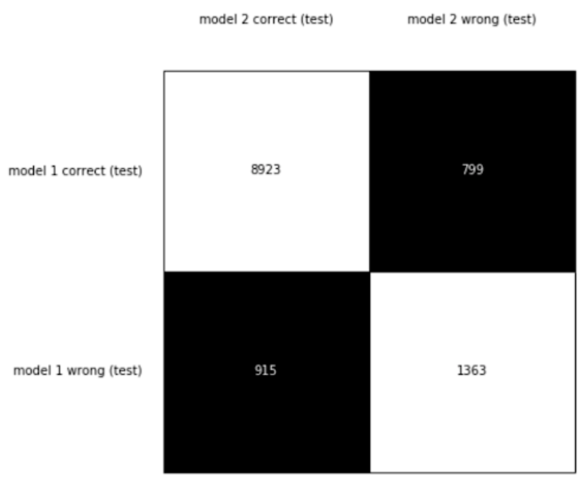


Figure 15. Model1: Xgboost

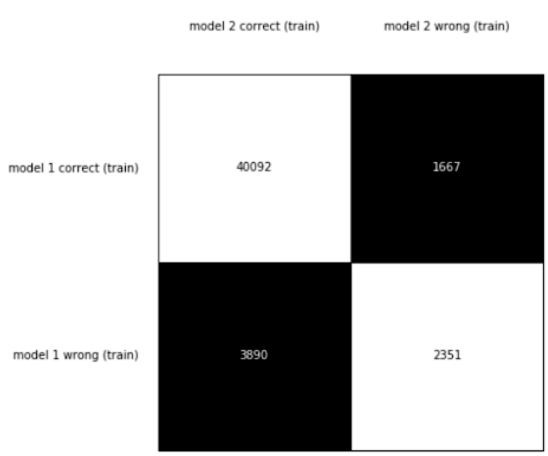


Figure 16. Model2: Logistic Regression

```
chi-squared-for-train: 888.4801151700558
p-value-for-train: 3.1341741004492237e-195
chi-squared-for-test: 7.715869311551925
p-value-for-test: 0.005473749531437572
```

Figure 17. Chi-squares and p-values

### 5.4. Compare the performances of two models – Xgboost & Logistic regression (Mcnemar's table)

On both training and testing dataset, logistic regression model makes less incorrect predictions than Xgboost. By using Mcnemar test, through the result from chi-square and p-value, there are significant differences by using two different models (model1 is xgboost, model 2 is logistic regression) with both train accuracy and test accuracy. However, percentages showing the accuracy do not illustrate a huge difference since the number of samples to examine test accuracy is not as large as that to examine train accuracy. In general, Model 2 performs better than model 1.[8]

### 5.5. Confusion Matrix

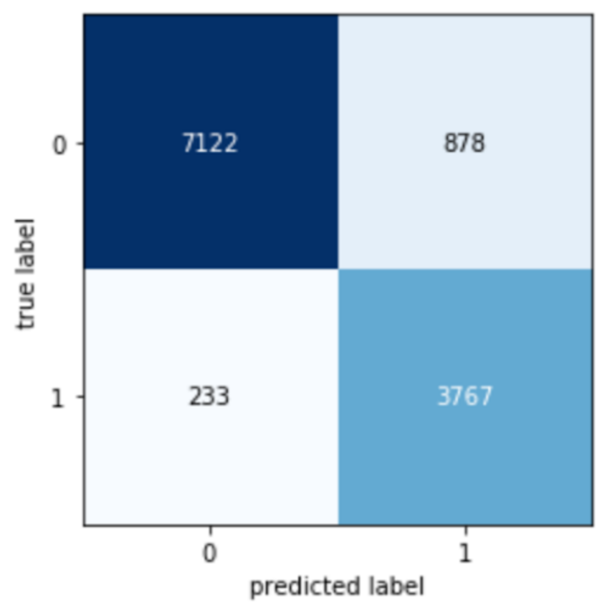


Figure 18. Xgboost

According to the plots above, both models tend to make more type 1 (false positive) errors than type 2 (false negative errors), and xgboost model has more errors on average.

## 6. Conclusions

The purpose of this analysis was to found out what words lead to higher quality questions. Broadly, we found those words to be: 'code', 'using', 'use', and 'file'. The words

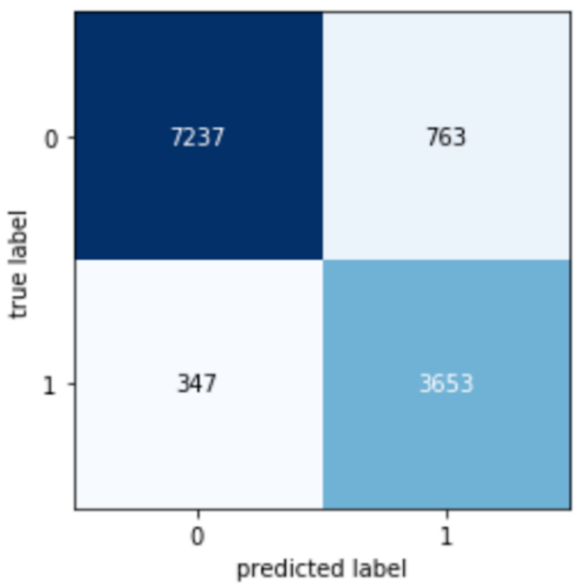


Figure 19. Logistic Regression

that show up most frequently in low quality questions are ‘code’, ‘use’, ‘one’, ‘value’,and ‘function’. In terms of length of title and body of the question, high quality questions have 50 to 75 words more than low quality questions on average, and the overall distribution is more concentrated, with a few titles with a length of less than 25 words, or more than 100 words. When examining model performance with both uncleaned and cleaned data, logistic regression gives the best accuracy of 82.283%, this happens when using uncleaned data. Surprisingly, using cleaned data does not give a more accurate result. Using lime to compare our model, Random Forest and Xgboost take similar words to determine if a question is of high quality or low quality, while logistic regression selects different words. Finally, we compare XGboost and logistic regression and find out that logistic regression gives the highest accuracy

## 7. Contributions

Ruohe Zhou cleaned the data, fit the models and wrote the experiment part of the report. Annabelle Wan also helped with cleaning the data, fitting the models and writing the result and discussion part of the report. Tsz Yau Iris Chow wrote the report excluding experiment, result and discussion.

## References


[1] Antoaneta Baltadzhieva and Grzegorz Chrupała. *Predicting the quality of questions on stackoverflow*. URL: https://aclanthology.org/R15-1005.pdf.

[2] Mukesh Chaudhary. *TF-IDF vectorizer scikit-learn*. Jan. 2021. URL: https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a.

[3] David Hin. *StackOverflow vs Kaggle: A Study of Developer Discussions About Data Science*. URL: https://arxiv.org/ftp/arxiv/papers/2006/2006.08254.pdf.

[4] Denzil Correa IIIT-Delhi et al. *Fit or unfit: Analysis and prediction of ’closed questions’ on stack overflow*. Oct. 2013. URL: https://dl.acm.org/doi/abs/10.1145/2512938.2512954.

[5] Christoph Molnar. *Interpretable machine learning*. Nov. 2021. URL: https://christophm.github.io/interpretable-ml-book/lime.html.

[6] Moore. *60K stack overflow questions with Quality Rating*. Oct. 2020. URL: https://www.kaggle.com/imoore/60k-stack-overflow-questions-with-quality-rate/version/13?select=valid.csv.

[7] Divya Raghunathan. *NLP in Python-Data Cleaning*. June 2020. URL: https://towardsdatascience.com/nlp-in-python-data-cleaning-6313a404a470.

[8] Sebastian Raschka. *Mlxtend*. URL: http://rasbt.github.io/mlxtend/.

[9] Sunil Ray. *CatBoost: CatBoost categorical features*. June 2020. URL: https://www.analyticsvidhya.com/blog/2017/08/catboost-automated-categorical-data/.

[10] *Removing stop words with NLTK in python*. May 2021. URL: https://www.geeksforgeeks.org/removing-stop-words-nltk-python/.

[11] Saikat Mondal University of Saskatchewan et al. *Early detection and guidelines to improve unanswered questions on stack overflow*. Feb. 2021. URL: https://dl.acm.org/doi/fullHtml/10.1145/3452383.3452392.

[12] *Stack overflow*. Dec. 2021. URL: https://en.wikipedia.org/wiki/Stack_Overflow#:~:text=As%5C%20of%5C%20March%5C%202021%5C%20Stack,questions%5C%20and%5C%2031%5C%20million%5C%20answers.