Mahyar Sabouniaghdam

BrainStation

BrainStation Capstone Project: Stack Overflow Questions Quality Rating

Project Report

Introduction:

The goal of this final capstone project was to develop a machine learning model to maintain a high-quality, organized repository of programming knowledge for all users to benefit from and predict the quality of questions asked on Stack Overflow, a widely used platform for software developers to ask and answer questions. The project was based on a dataset of 60,000 Stack Overflow questions and their corresponding quality ratings.

problem statement:

Stack Overflow receives more than 6,000 new questions every weekday and millions of questions every year and determining the quality of these questions is important for maintaining the platform's reputation as a reliable source of information. Currently, about 6% of all new questions end up "closed". Questions can be closed as off-topic, not constructive, not a real question, or too localized. More in-depth descriptions of each reason can be found in the Stack Overflow FAQ. The exact duplicate close reason has been excluded from this contest since it depends on previous questions.

Background on the subject matter area:

The project adds business value by improving the user experience and reducing the time and effort required to filter through low-quality questions and the quality prediction of questions on question-and-answer platforms is a relevant problem area to apply data science techniques.

Details on the dataset:

This is an original dataset, published under MIT License. Please cite the dataset for your usage as the following:

{Annamoradnejad, I., Habibi, J., & Fazli, M. (2022). Multi-view approach to suggest moderation actions in community question-answering sites. Information Sciences, 600, 144-154.}

Data is collected from: https://github.com/Moradnejad/StackOverflow-Questions-Quality-Dataset

The dataset is also accessible at: https://www.kaggle.com/imoore/60k-stack-overflow-questions-with-quality-rate

I downloaded it from the Kaggle website. This is a dataset containing 60,000 Stack Overflow questions from 2016-2020. Questions are classified into three categories:

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

- Notes:
  - Questions are sorted according to Question Id.
  - The question body is in HTML format.
  - All dates are in UTC format.

Data Dictionary:

The dataset is divided to train (45k) and validation (15k) datasets.

Independent Variables:

- Id: Id of the question
- Title: Question Title
- Body: Text body of the question
- Tags: Tags included for each question (Max= 5 tags)

- Creation Date: From 1 JAN 2015 to 29 FEB 2020

Dependent Variable:

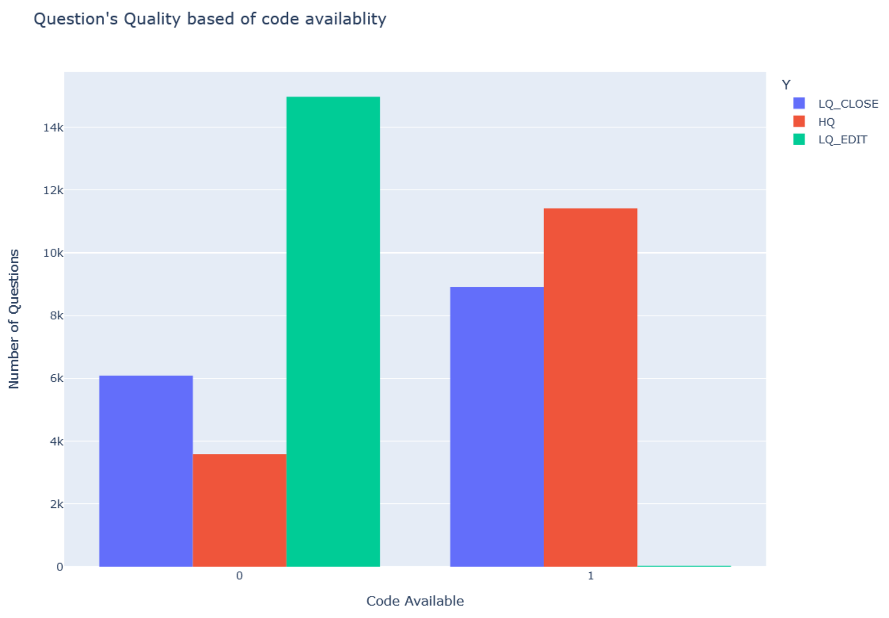
- Y: 3 Unique values HQ, LQ_EDIT, LQ_CLOSE

Summary of cleaning and preprocessing:

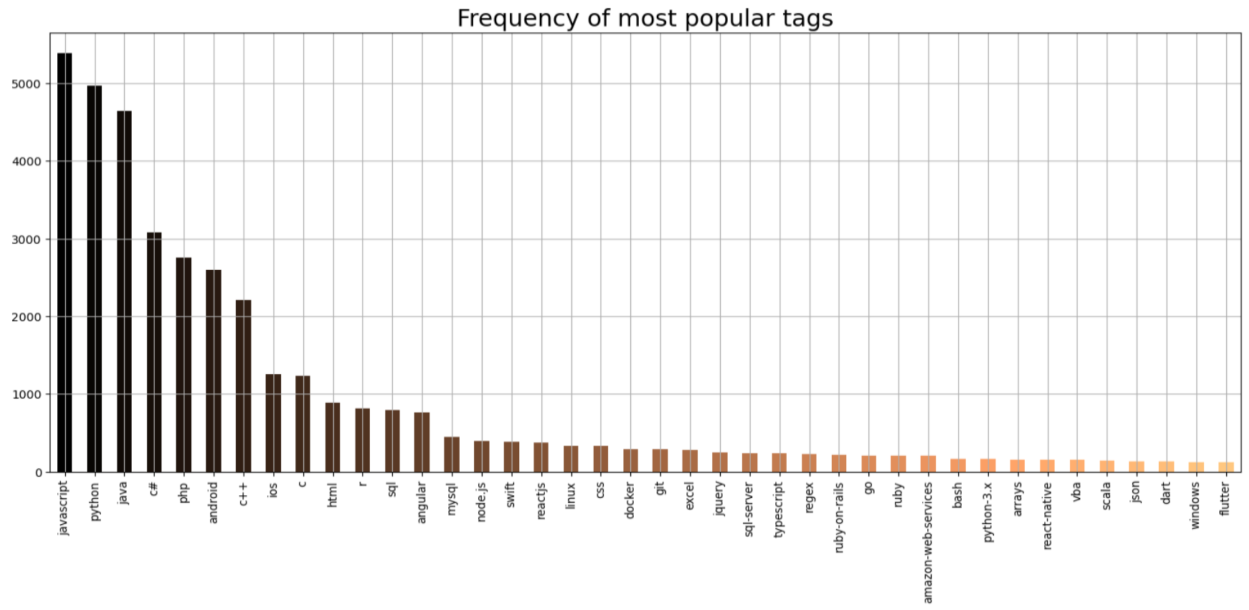
I started to convert my target variable 'Y' to HQ and closed questions by adding LQ_EDIT rows to HQ. I had only 5 feature columns so I started to create new columns out of these 4 columns. The id column had no use so I dropped it. I created Hour, weekday, month, and year columns from the Creation Date column. I created the number of words column for the body and title text because I thought it may have a connection with the target column. I created the number of tags column and my body data had <code> HTML tag in it so I created a column named code availability. I came to 17 columns from 6 columns.

Observations:

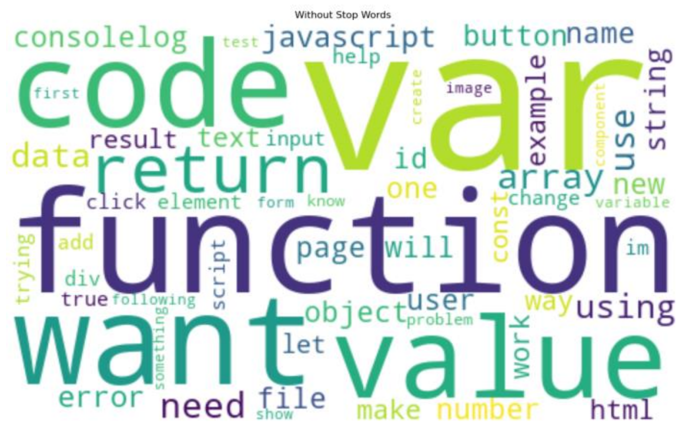
- Most of the questions are created between 10-14 and most between 12-1 pm.
- People created more questions on weekdays compared to weekends.
- number of questions in the beginning months is higher. Maybe because the 2020 year has only Jan and Feb months.
- 2016 has the highest number of questions.
- Most of the titles had between 4-6 words.
- most questions had 2 tags.
- the lower number of questions had codes in them.
- the shorter and fewer words and characters, the more chance of having a low-quality question.
- the number of titles is denser than the body and they are all gathered between 4-7.
- the questions created at night are more likely to be closed.
- the more tags we have, the more chance to have a high-quality question.
- when we have code in our question, the chance of having a High-Quality question is highest and when we do not have code in our question, the chance of having a High-Quality question is lowest:



I had so much trouble with the Tags column because I had 10,000 tags and I ran into Memory Error once I wanted to OHE them. So, I found the codes repeated more than 300 times and wrote a hardcode to categorize them:



Most repeated words in text of the question for words with JavaScript tag, which is my top tag:



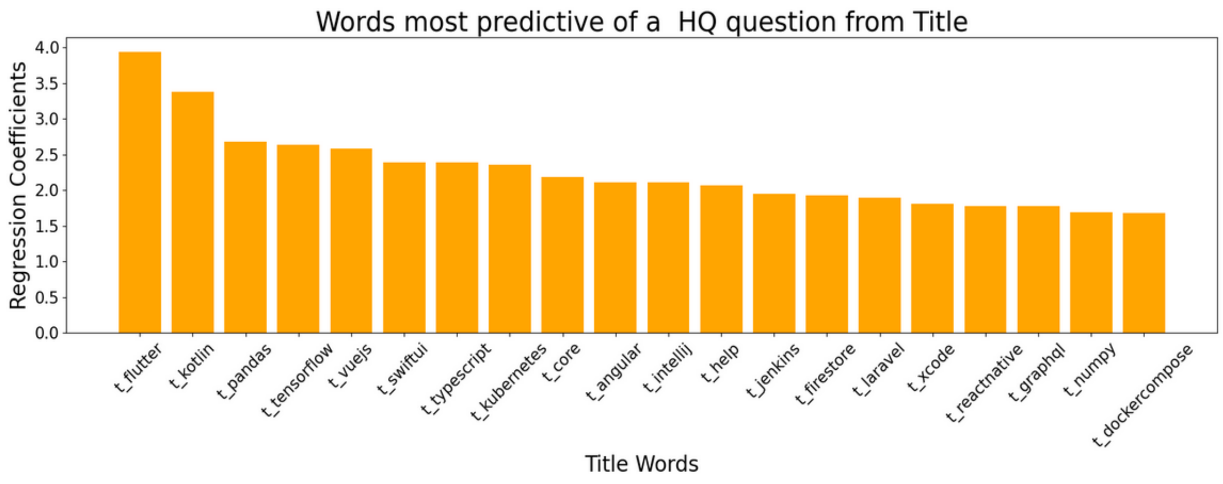
Insights, modeling, and results:

I created a correlation matrix and cleaned the text of my body and title column. Since the text length of our body column is so large, after vectorizing we will have too many columns and words and we will face Memory Error. Moreover, as we mentioned earlier, I am going to subset my data to tackle the imbalance issue I mentioned earlier. To avoid these problems, I took a sample of the data. I used TF-IDF to vectorize my data and then ran my baseline model. 74.7% was my test score for my baseline Regression model. This is my insights from it:

1. most predictive words for high-quality questions for the title are advanced and technical words for different branches of the programming community.

2. top 20 titles for closed questions are general words compared to technical and advanced words for HQ questions predictive words.

3. It is interesting that JavaScript and PHP titles are among the top 20 most predictive words for closed questions. The reason may be because of the high number of questions related to JavaScript, so there would be a higher chance of creating low-quality questions too.



Then I started to Optimize my models. I scaled it with Min Max Scaler and I did a PCA model. The optimal number of components based on the TF-IDF Vectorized Data was 1237. Since there was no significant improvement in the results of the PCA model, I did not use dimension reducers when I move on to advanced modeling for running an ML Pipeline with Grid Search for finding the best model with optimal hyperparameters. the models that will be run for optimization are: The models that will be run for optimization are:

1. Logistic Regression: With a C_value of 0.1 we got a 77.31% accuracy on the remainder set and a 75.07% accuracy on the test.
2. SVM (Support Vector Machines): SVC accuracy on the test set with a C_value of 0.01 is 75.24%
3. Random Forrest (Decision Trees): got a ~84% accuracy on the remainder and ~73%. There is a ~10% difference between test and remainder accuracy with max_depth= 21, n_est=13
4. XG Boost: We get 84.6% accuracy on the remainder, and 79% accuracy on the test with n_estimators = 40 / max_depth = 6

XG Boost was my best model in manual optimization. Then I ran some advanced modeling and Grid Search which gave me the same results and my best model was XG Boost again. Then I created another notebook and tried some deep-learning models. I tried ANN and Multi Class NNs. The test Accuracy score for the XG Boost model in that notebook was 79.43% and it was better than the Multi Class NNs model at 79.36%. The score is almost the same as the XG Boost model in the previous notebook. My NNs models indeed have lower test scores than my XG Boost model in this notebook. NNs models are more complex and I tried and wanted to run some other NNs models but because of computational limits, I could not.

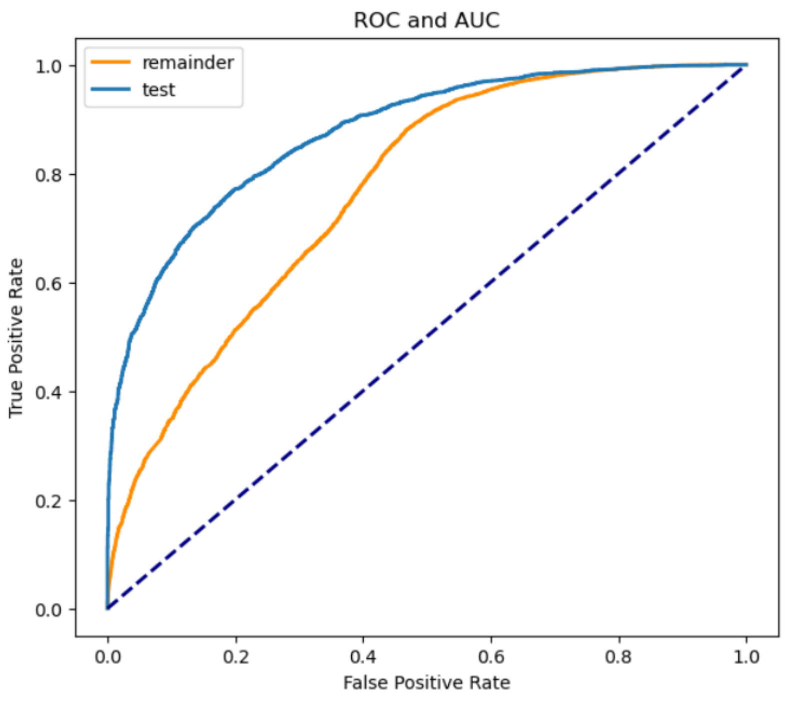
Findings and conclusions:

As we saw on the supplementary notebook for deep learning XG Boost is still our best model, The next check for my XG Boost model is to plot out the change in the precision and recall scores based on a range of threshold values. By doing this I aim to find the optimal threshold value which would lead to the highest recall rate. It looked like the optimal threshold value might be between 0.2 and 0.4. As a final check for comparing the true positive rates (TPR - how many HQ questions are correctly being classified as HQ AKA recall) and false positive rates (FPR - how many closed questions are being misclassified as HQ question), the Receiver Operating Characteristic (ROC) curve will be the plot.

Via the ROC Curve, I aim to check the Area Under the Curve (AUC) value which represents how well the model can distinguish between the closed and HQ questions.

AUC Score definition:

- 1 = model is perfectly able to distinguish between closed and HQ questions
- 0.5 = model is unable to distinguish between closed and HQ questions
- 0 = the model is predicting all closed questions as HQ and vice versa



Based on the AUC score, there is a ~0.88 probability that the classifier will correctly classify a randomly chosen HQ question. This is a good result, indicating that the model will perform well for the HQ question recognizer.

Final Model & Next Steps:

Based on all the advanced modelling conducted in this notebook and deeplearning supplemantry notebook, the XGBoost model with the manual hyperparameters was the most successful at classifying the closed and HQ Questions.

Best Model Parameters:

Model	Default Model Parameters	Remainder Accuracy (%)	Test Accuracy (%)
XGBoost	learning_rate = 0.5, max_depth = 6, n_estimators = 40, random_state = 17	84.7%	79%

Next Steps:

On any project, there will always be areas for improvement or further work. This is especially true when there are time and computational limitations (such as i the case with this project). Some areas this work could be furthered or improved include:

- Search for less biased data
- Engineer some features from the dataset
- Conduct a WordEmbedding analysis
- Run an ML Pipeline with GridSearch on a wider array or models and hyperparameters (possibly via AWS due to computational limits with local machine)
- Develop an API to predict the quality rating of the questions