Machine Learning Engineer Nanodegree

Capstone Project

Kamal Kaushik

(kmlkshk@gmail.com)

March 08th, 2018

Quora Question Pairs (Kaggle)

I. Definition

Project Overview

Quora is place where people can ask any kind of questions and they are answered or edited by its community members who have knowledge in those specific areas. Since millions of people use Quora to get answers of their queries, so you will find many people asking similar questions. Multiple answers to one question can lead in spending a lot of time in finding the best answer to the question raised by a member. Here, our aim is to find an algorithm and to apply that so as to gather all duplicate questions together and list them out when a member is looking for an answer to that question. Also, it will help members to answer the questions all at once who wish to do so.

Quora has provided train data which has 404290 question pairs with a label column indicating if they are duplicate or not. When I analyze the data I found 255027 negative (non-duplicate) and 149263 positive (duplicate) instances.

It is a NLP problem where Natural Language Sentence Matching (NLSM) is the task of comparing two sentences and identifying the relationship between them. The task is critical for many applications in natural language processing (NLP), such as information retrieval , question answering and paraphrase identification .Natural language sentences have sequential and hierarchical structures, that are essential for understanding them. Sentence similarity ranges between 0% (no relationship at all) and 100% (sentences are semanticaly identical).

Problem Statement

This is a binary classification problem where I have used the Quora dataset from Kaggle competition and this is related to the problem of identifying duplicate sentences. Inputs are two sentences of texts and the goal is to predict if these two sentences are of the same meaning or not.

To solve this problem I have the same motivation as Quora had when releasing their dataset: "there should be a single question page for each logically distinct question". This type of problem is challenging because we usually can't solve it by looking at individual words. No single word is going to tell you whether two questions are duplicate. I have looked at both the items together.

The duplicate detection problem can be defined as follows: given a pair of questions q1 and q2, train a model that learns the function:

f(q1, q2) ► 0 or 1

where 1 represents that q1 and q2 have the same intent and 0 otherwise.

Metrics

I propose to use the evaluation metric defined by Kaggle on the Quora competition, where the model will be evaluated on the log loss (logistic loss or cross-entropy loss) between the predicted values and the ground truth. The ground truth is the set of labels that have been supplied by human experts. The ground truth labels are inherently subjective, as the true meaning of sentences can never be known with certainty. Human labeling is also a 'noisy' process, and reasonable people will disagree. As a result, the ground truth labels on this dataset should be taken to be 'informed' but not 100% accurate, and may include incorrect labeling.For each ID in the test set, there must have a prediction on the probability that the questions are duplicates (a number between 0 and 1).

The log loss looks at the actual probabilities as opposed to the order of predictions. The metric is negative the log likelihood of the model that says each test observation is chosen independently from a distribution that places the submitted probability mass on the corresponding class, for each observation.

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

II. Analysis

Data Exploration

The Quora dataset is a set of questions , with annotations which indicates whether the questions seeks the same information or not. The training data and test data from this problem are provided on Kaggle website: https://www.kaggle.com/c/quora-question-pairs/data. This data set is large, real, and relevant — a rare combination. Every line has a unique ID for each question in the set and a binary value which is 0 or 1 indicates whether the line has a duplicate pair or not.

We will consider that the dataset is divided into two files, training and test, below is a quick analysis-

Datasets

1. **Training**:
    - Question pairs: 404290
    - Questions: 537933
    - Duplicate pairs: 36.92%
2. **Test:**
    - Question pairs: 2345796
    - Questions: 4363832
    - Question pairs (Training) / Question pairs (Test): 17.0%

Input Data fields

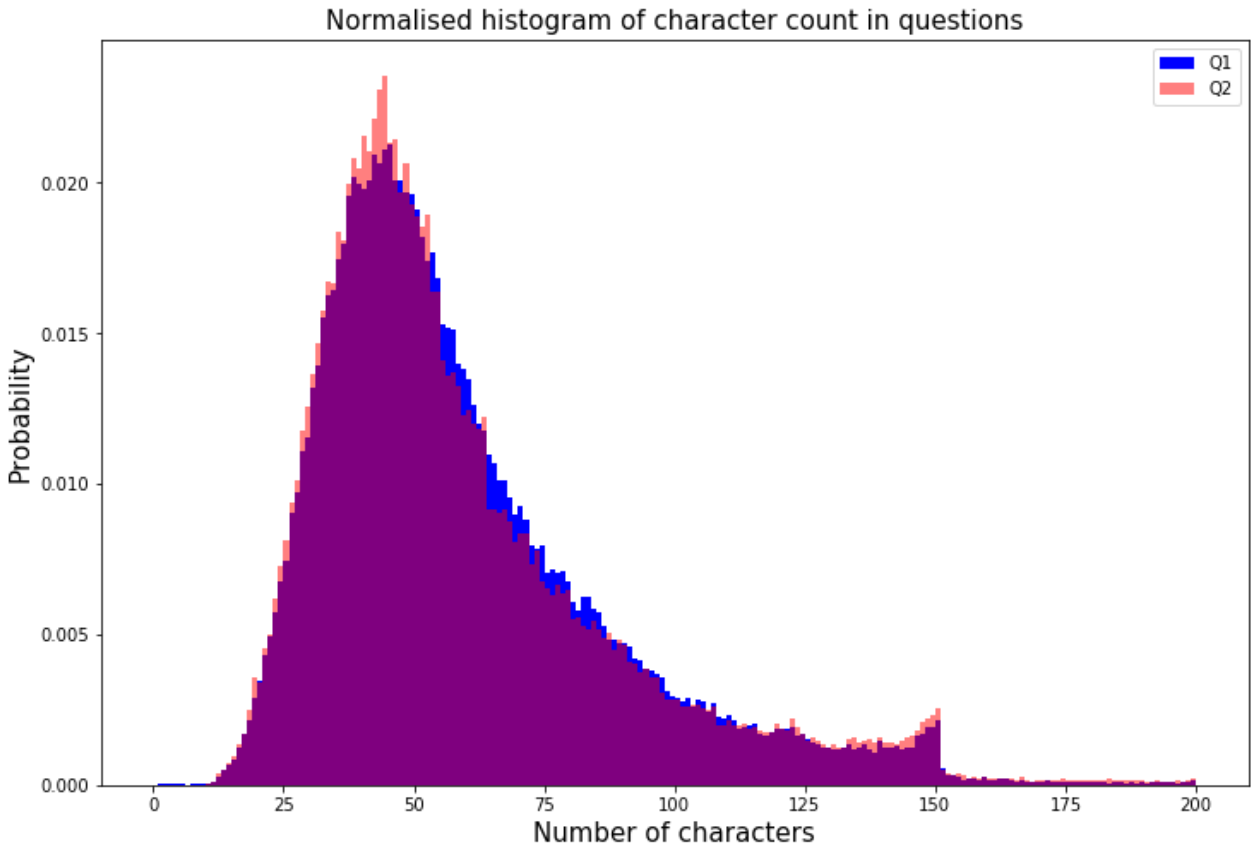
- id - the id of a training set question pair
- qid1, qid2 - unique ids of each question (only available in train.csv)
- question1, question2 - the full text of each question
- is_duplicate - the target variable, set to 1 if question1 and question2 have essentially the same meaning, and 0 otherwise.

Let's look at the first 5 lines of training data:

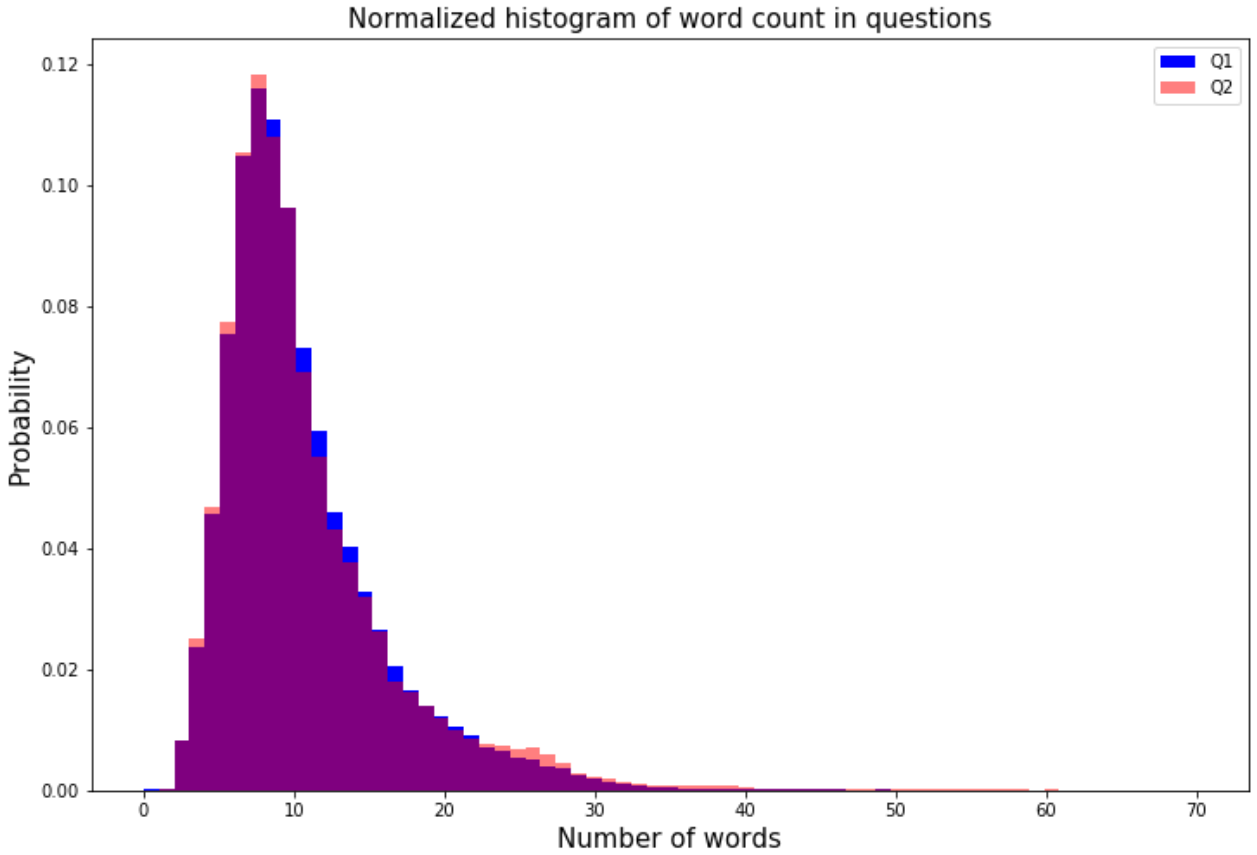
id	qid1	qid2	question1	question2	is_duplica
0	1	2	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
1	3	4	What is the story of Kohinoor (Koh-i-Noor) Diamond?	What would happen if the Indian government stole the Kohinoor (Koh-i-Noor) diamond back?	0
2	5	6	How can I increase the speed of my internet connection while using a VPN?	How can Internet speed be increased by hacking through DNS?	0
3	7	8	Why am I mentally very lonely? How can I solve it?	Find the remainder when [math]23^{24}[/math] is divided by 24,23?	0
4	9	10	Which one dissolve in water quikly sugar, salt, methane and carbon di oxide?	Which fish would survive in salt water?	0

Exploratory Visualization

We can see in the graph a normalized histogram of character count in both question1 and question2. Question1 is represented by blue colour whereas question2 by red colour. It is clearly visible in the graph that character counts distributions are almost the same for question1 and question2. It is highest at a peak of around 45 characters and have an average of 55 characters. Interestingly, at 150 characters there is a slight increase in numbers and then a sudden drop.

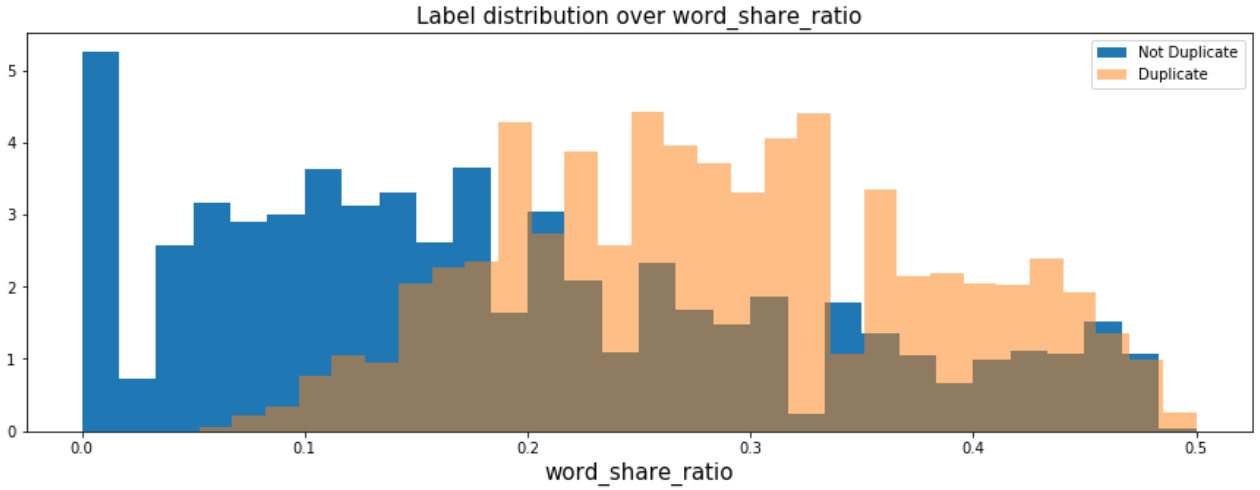


In second graph we can see the normalized histogram of number of words from question1 and question2.

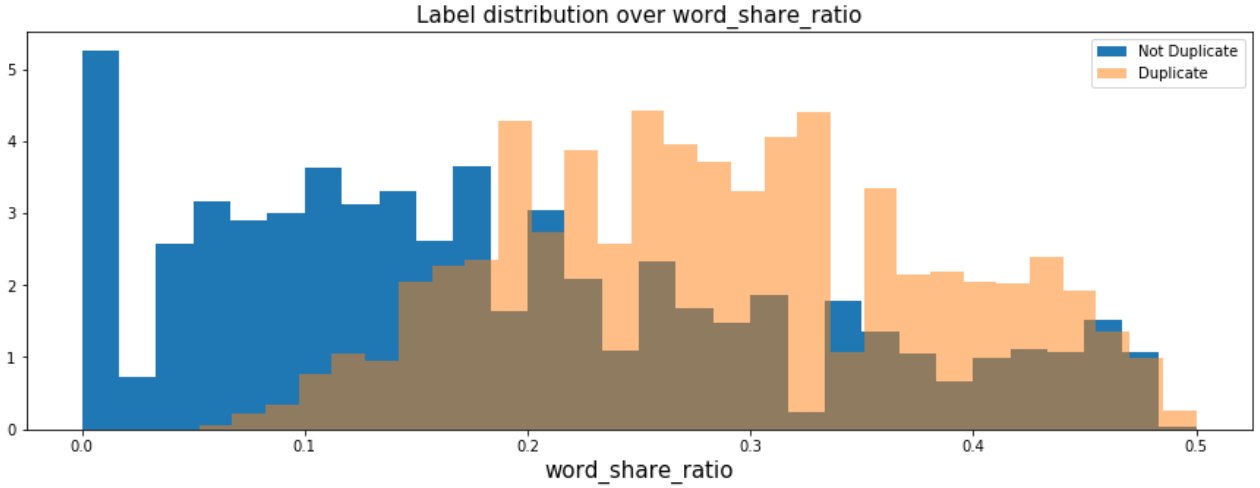


Here, the number of words in questions peak at 10 words. Most people ask questions with roughly 10 words, which is expected.

Below graph shows the bar plot of word_share_ratio. Word_share_ratio is an extracted feature from the training data. The larger is the ratio the more overlapping will be seen in questions. Here, beige color represents duplicate question pairs and blue color non-duplicate question pairs. It is seen that duplicate question pairs generally have more words shared between them.



Below graph is the TFIDF word share ratio which is again a feature extracted from the training data. This also shows that the larger the ratio the more similar the two questions are.



Algorithms and Techniques

Since this is a binary classification problem, I have used the multiple learning algorithms like Logistic regression, SGD classifier and Random forest to get the best possible output.

Below are the steps which I followed-

1. Cleaning up of the strings by removing leading and trailing white space and converting to lower case. Also, if there are any NaN values I have replaced them in earning process.
2. Extracting the features from the attribute given on the data. These features include: word count, character count, word share, TF-IDF share.
3. Then after I have used the StandardScalar to normalize the features.

Following are the algorithms which I have used in the implementation to find the log loss-

1. Logistic regression is a regression model where the outcome is binary, and the model predicts the probability (0 to 1) of the outcome.
2. SGD classifier(stochastic gradient descent) is a learning model where the gradient of the loss is estimated at each sample at a time and the model is updated along the way with a decreasing strength schedule.
3. Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.
4. Naive Bayes classifier is a family of simple probabilistic classifiers based on applying Bayes' theorem with naive independence assumptions between the features.
5. Support vector machine is an algorithm which outputs an optimal hyperplane to separate labeled training data.

Finally, I have imported 2 forms of XGBoost i.e xgb , XGBClassifier where xgb is the direct xgboost library in which I have used a specific function "cv" from this library and XGBClassifier is an sklearn wrapper for XGBoost. This allows us to use sklearn's Grid Search with parallel processing in the same way we did for GBM. XGBClassifier is gradient boosting model is similar to random forests, with the difference being how we train them. For gradient boosting, I have used the StandardScaler transform data to fit the XGBClassifier and predict the score for logloss.

Benchmark

At present to identify duplicate questions, Quora uses Random Forest Model. Also, in this project for the benchmark model it would be logical to use random forest model only.
The random forest model was measured using the Log Loss metric described above, as this is what was used to evaluate results from the other classification models.

III. Methodology

Data Preprocessing

In the first step after analyzing the data I have cleaned up the strings by removing leading and trailing white space and then converted to lower case. Also, if there are any NaN values I have replaced them in the cleaning process.

There are multiple features like Question1 number of words, Question1 character count, Question2 number of words, Question2 character count, word share ratio, TFIDF vector share ratio. The first four are extracted using basic python string functions.

Word_share is calculated as the number of words appearing in both question1 and question2 divided by the total number of words in question1 and question2. The result is a normalized value measuring how much word-overlap is between question1 and question2.

For TFIDF factors, instead of using the sklearn TFIDF library I have written a similar function implementing the TFIDF idea so that it won't look like that I am using a blackbox function, and it provides more opportunity to do the TFIDF vector share calculation.

Firstly, I put together all questions into one big corpus, and then calculated weight for each word by the inverse of word count plus epsilon (=5000) such that the less common words get higher weights. I simply, ignored the words that appeared once in the whole corpus for the reason that it may be typos.

Then I calculated the tfidf_word_share_norm using share_weight divided by total_weight .where-
Share_weight is the total weight of the words appearing in both question1 and question2.
Total_weight is the total weight of every words in either question1 or question2.
The result is a normalized TFIDF_weight_share.

Implementation

First of all I cleaned up the strings by removing markup texts, leading and trailing white space and then converted them to lower case using beauty soap library. Then I normalized the features using TfidfVectorizer and applied the pipeline to run the process to generate the log loss by using different classifiers. But the result showed high log loss which is actually not good. Then I decided to opt for another approach in which I didn't used beautysoap library for cleaning and rather using TfidfVectorizer for normalizing the features I used StandardScalar. I tried multiple learning models excluding the benchmark model. Except the Naïve Bayes model, I normalized all feature values before doing model fit.

The following are implementation of each model:

Logistic regression: I used GridSearch cross validation to find best parameters ('C', 'penalty')

SGD classifier Setting for parameters are loss='log', penalty='l2',alpha=1e-3, n_iter=5, random_state=42.

4. K-Nearest Neighbors: I also did GridSearch cross validation to find best parameters ('n_neighbors', 'weights')

5. Naive Bayes: Naive Bayes features work better if their values are integers. Since Naive Bayes is based on probability, there is no need for feature value normalization.

6. Gradient Boosting (XGBoost): Setting for parameters are (learning_rate =0.01, n_estimators=5000, max_depth=4, min_child_weight=6, gamma=0, subsample=0.8, colsample_bytree=0.8, reg_alpha=0.005, objective= 'binary:logistic', nthread=4, scale_pos_weight=1, seed=27) .I did 200 rounds of boosting.

During the implementation phase, I faced many challenges like – In my first approach I used the

pipeline with TfidfVectorizer and different classifier in which the result for logloss was too high. Even with every classifier used the result was between .70 – . 90 and since I was short of time I couldn't work on this approve to improve the results. So, I tried another approach where I used the word_share, tf-idfword_share and StandardScaler to normalize data but in this scenario the result was very close to the value which I defined in my benchmark . So, I am planning to apply different normalization with existing scaled data to see the better result.

Refinement

In the initial step I have used the pipeline to transform data with a final estimator. The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters. I have used the TfidfVectorizer to normalize the feature and apply the different classifier in which the result for logloss are very high. Score received using the random forests benchmark model was a log loss score of 0.655. In order to improve this initial benchmark score, I explored alternative features and implemented one by making adjustments to the data preprocessing steps. In this I have constructed features for training data i.e. character length, number of words, normalized word share which gives me the better result. After making desired changes the Logloss score of benchmark model came down from 0. 655 to 0.487, this shows improved result. To complete with benchmark model I used XGBoost with same Scaled data and the result (0.452503) was not that impressive but it showed little improvement.

IV. Results

Model Evaluation and Validation

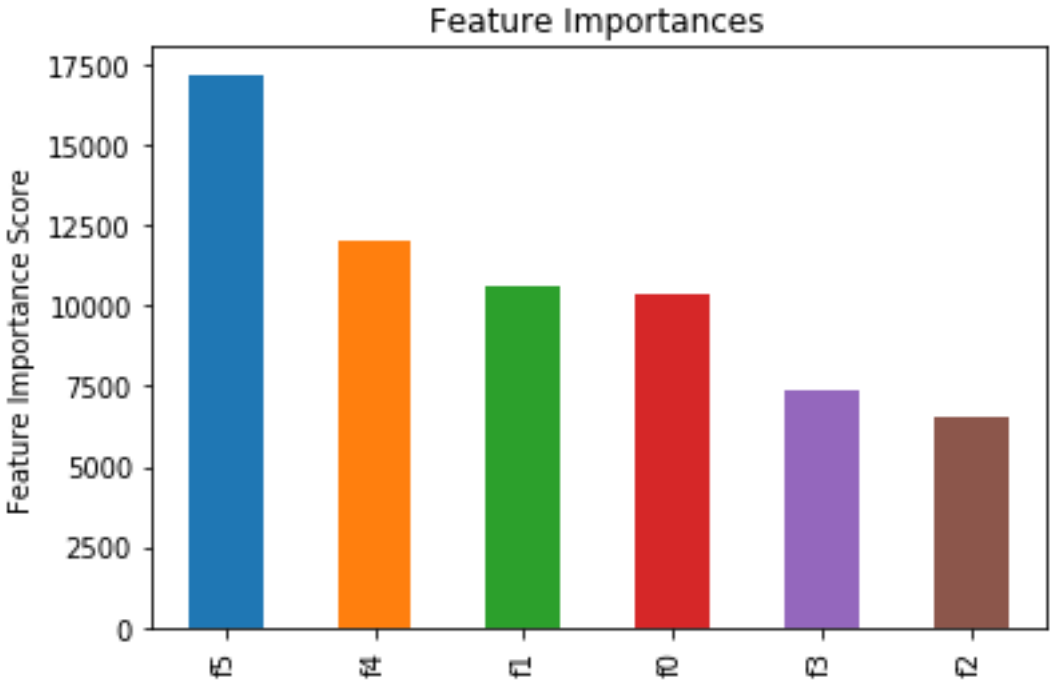
In this problem I have used multiple models to train and evaluate the log loss .The final model I have used is a combination of XGB and XGB Classifier which is similar to random forest, but with a better performance. In XGBoost I fine-tuned the max_depth, gamma, min_child_weight for the better result. Also XGBoost has a very useful function called as "cv" which performs cross-validation at each boosting iteration and thus returns the optimum number of trees required. In the end I tuned regularization parameters (lambda, alpha) for xgboost which helps to reduce model complexity and enhance performance. I have used scaled dataset to make sure there is no overfitting and to find the best parameters within reasonable training time. And to validate the model I have used grid search on multiple modules to tune the parameters and to get the best outcome.

Justification

Yes the final result I have got has improved a little then the benchmark model. Although the idea behind random forest and XGboost is the same, the training process is different. I think XGboost is a more robust model and hence should give better result. In my project, benchmark model(random forest) has log-loss 0.487 while XGboost model has log-loss 0.452503. I believe the final result is better although it is not 100% accurate.

V. Conclusion

Free-Form Visualization



For free form visualization I have used plot() function to visualize the feature importances and the scores in the XGBoost training model.
It can be seen that two features (f5 and f4) have the highest score dominating the final decisions. I assume that f5 and f4 to be word_share and TFIDF_word_share.

Reflection

Initially the problem seemed to be a simple linear regression classification problem. Then I found out the number of words and character length, and then the amount of the same words shared between two questions and it seemed that it is a NLP related problem so, I thought of using TFIDF to process the sentences. But, I didn't get very good results So, finally I decided to implement StandardScalar for feature normalization and applied different algorithms on scalaed data.

Gradient boosting gave me the better result as compared to benchmark model. After that I fine-tuned the parameter and got 0.452503 log-loss . I also tried to add more features extracted from TFIDF vectors, but did not get any significant accuracy gain.

My reflection on this is: adding features is not always better as after adding more TFIDF vectors later results didn’t improved much.

Improvement

Taking into consideration the future perspectives to improve the project I can focus more on sentiment analysis and Natural Language Processing Techniques. I can also try extracting punctuation, upper/lower letter, special characters, foreign languages, math symbols, emojis etc. for more detailed analysis. Can also try learning neural network and convolutional neural network thoroughly and implement them to see outputs.