Quora Question Pair Report

September 23, 2021

1 Introduction

There are over 100 million people visiting Quora every month, it is quite possible that people ask similarly worded questions. Quora uses the random forest model to classify duplicate questions currently. The goal of this challenge is applying advanced techniques to evaluate whether the provided pairs of questions deliver the same meaning. Once successfully identify duplicate questions, users can easily find high quality answers without spending more time looking for best answer among multiple similar questions, and also less time for writers to answer the same questions multiple times.

Quora is a question answering website where users ask questions and other users respond. The best answers are up-voted and these answers are a valuable learning resource for many topics. Duplicate questions on this site are not uncommon, particularly as the number of questions asked grows. This poses an issue because, if treated independently, duplicate questions may prevent a user from seeing a high quality response that already exists and res-ponders are unlikely to answer the same question twice. Identifying duplicate questions addresses these issues. It reduces the answering burden for responders and makes it possible to direct users to the best responses, improving the overall user

2 Data

The data is in a csv file named "quora.csv". quora.csv contains 5 columns : qid1, qid2, question1, question2, is_duplicate Number of rows in quora.csv = 100000 'qid1' and 'qid2' are the ids of the respective questions, 'question1' and 'question2' are the question bodies themselves and 'is_duplicate' is the target label which is 0 for non similar questions and 1 for similar questions.

is_duplicate **0** means for non similar question we have **62879** is_duplicate **1** means for similar question we have **37121**

3 Data Pre-processing

Before Extracting Basic and Advanced Features we need to perform preprocessing - Steps to be followed-

- Tokenization
- Lower casing
- Removing Special Characters
- Stop words removal
- Stemming/Lemmatization

3.1 Tokenization

Tokenization is a way of separating a piece of text into smaller units called tokens. Here, tokens can be either words, characters, or sub-words. Splitting the sentence into words. Tokenization is commonly used to protect sensitive information

3.2 Lower casing

Converting a word to lower case (NLP -¿ nlp). Words like Book and book mean the same but when not converted to the lower case those two are represented as two different words in the vector space model (resulting in more dimensions)

3.3 Stop words removal

Stop words are very commonly used words (a, an, the, etc.) in the documents. These words do not really signify any importance as they do not help in distinguishing two documents. For removing stop words, We made use of NLTK, a popular NLP library for Python. They have a comprehensive set of stop words for many languages.

3.4 Stemming/Lemmatization

It is a process of transforming a word to its root form. Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is important in natural language Lemmatization: Unlike stemming, lemmatization reduces the words to a word existing in the language. Either Stemming or Lemmatization can be used. Libraries such as nltk have stemmers and lemmatizers implemented. These are built based on a rule-based approach. Stemmer is easy to build than a lemmatizer as the latter requires deep linguistics knowledge in constructing dictionaries to look up the lemma of the word. For lemmatization to resolve a word to its lemma, part of speech of the word is required. This helps in

transforming the word into a proper root form. However, for doing so, it requires extra computational linguistics power such as a part of speech tagger.

4 Featurization

Featurization is process where we extract features from our data. These features are what we will be feeding into our machine learning algorithms Featurization is quite possibly the most important aspect of machine learning. It is the so called "art" part of data science. We have Extracted Basic and Advantages Features.

Basic Features are-

- Q1_len: Number of characters in question 1.
- Q2_len: Number of characters in question 2.
- Q1_num_of_words: Number of words in question 1.
- Q2_num_of_words: Number of words in question 2.
- Num_of_same_words : Number of words which occur in question 1 and two, repeated occurrences are not counted.
- clean_question1_stem- Performed Stemming for Question 1
- question1_length_stem- Length of clean_question1_stem
- clean_question2_stem - Performed Stemming for Question 2
- question2_length_stem-Length of clean_question2_stem
- clean_question1_lema- Performed lemmatization for Question 1
- question1_length_lema-Length of clean_question1_lema
- clean_question2_lema- Performed lemmatization for Question 2
- question2_length_lema– Length of clean_question2_lema

Advanced Features are-

FuzzyWuzzy is a Python library which has some methods to compare string equivalence. I highly recommend taking 15 minutes to go through this link, which explains the various functionalities of FuzzyWuzzy. I used different string comparison techniques from FuzzyWuzzy to extract fuzzy features. The fuzzy features are:

- Simple Ratio : Measurement of edit distance (Minimum number of edits required to convert one sentence to other)

- Partial Ratio : How much accurately a part of sentence match to other sentence ("Chennai Super Kings", "Super Kings")
- Token Sort Ratio : Tokenizing the string in question, sorting the tokens alphabetically, and then joining them back into a string
- Token Set Ratio : Tokenize both strings,split the tokens into two groups of intersection and remainder. We use those sets to build up a comparison string.
- Last_Word : Checks if last word is same in both Q1 and Q2
- Firs_Word : Checks if First word is same in both Q1 and Q2
- Length_diff : Finds the length diffrence between Q1 and Q2
- StopWord_Ratio : Number of stopwords in both Questions
- Token_Ratio : Number of tokens in both Questions
- Longest_Substr_ratio : Ratio of the Longest Substring that is found in between Q1 and Q2

5 Vectorizing textual features

Computers have a hard time understanding strings and textual characters. Humans are very good at it.So we need to perform techniques like BOW,TFIDF,Word 2 vector Bag of Words (BOW) is a method to extract features from text documents. These features can be used for training machine learning algorithms. It creates a vocabulary of all the unique words occurring in all the documents in the training set. Machine Learning algorithms cannot be used directly on any textual data as they can only process numerical data in the form of an array. This is why we need to convert text, images, audio or any type of data into numerical data first and then only we can use machine learning algorithms.

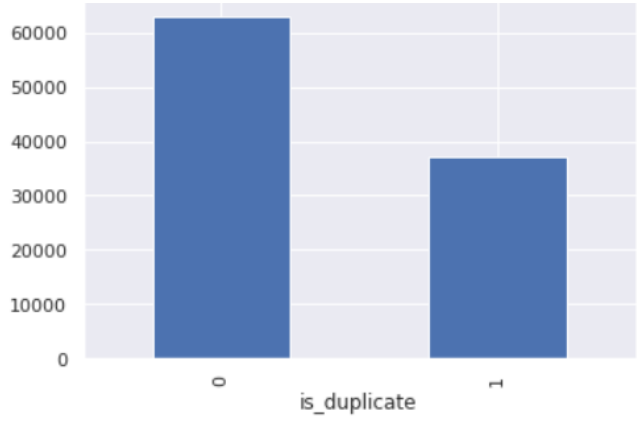
BOW: Which stands for Bag of words is a Natural Language Processing technique of text modelling. In technical terms, we can say that it is a method of feature extraction with text data. ... It is called a "bag" of words because any information about the order or structure of words in the document is discarded.

Disadvantages of BOW: If the new sentences contain new words, then our vocabulary size would increase and thereby, the length of the vectors would increase too.Additionally, the vectors would also contain many 0s, thereby resulting in a sparse matrix (which is what we would like to avoid) We are retaining no information on the grammar of the sentences nor on the ordering of the words in the text.

6 Exploratory Data Analysis(EDA)

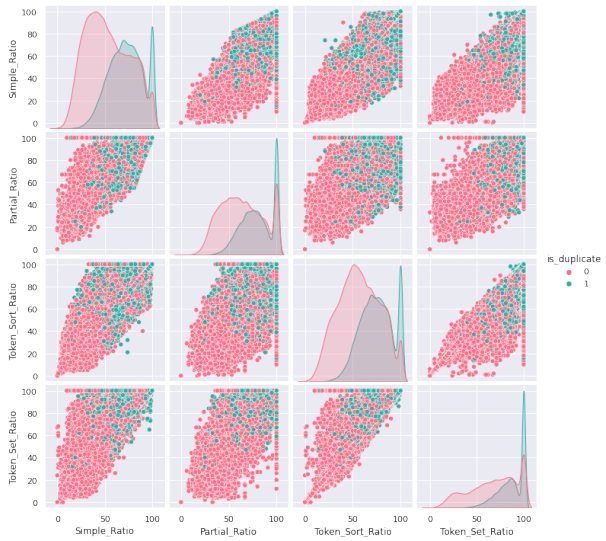
Exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

Distribution of data points among output classes

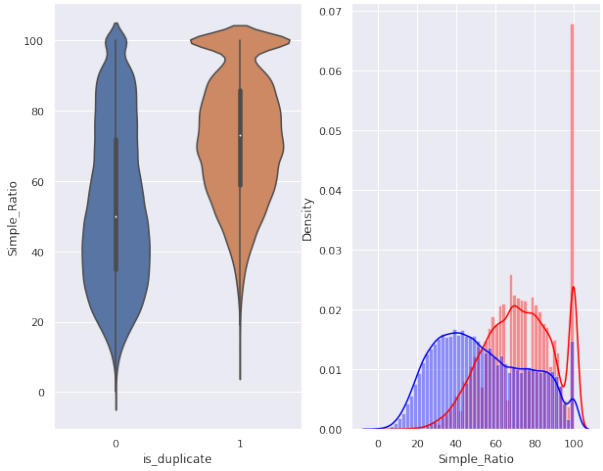


We have 62.88 % of non duplicate pairs and 37.12 % duplicate pairs.

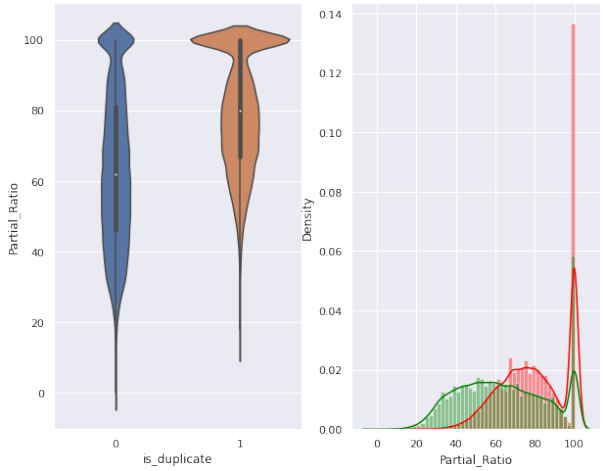
we have plotted word cloud for question1 text data.Word cloud is an image composed of words used in a particular text or subject, in which the size of each word indicates its frequency or importance.



These are the pair plots of few of the advanced features. One this we can observe is that almost all the plots have partial overlapping. Hence we can conclude that these features can provide partial separability. They all provide some predictive power



The Violin plot describes about probability density of data at differnet values for features is_duplicate and Simple_ratio The Distplot describes about the distributions for normalized Simple_Ratio have some overlap on the far right-hand side, i.e., there are quite a lot of questions with high Ratio similarity

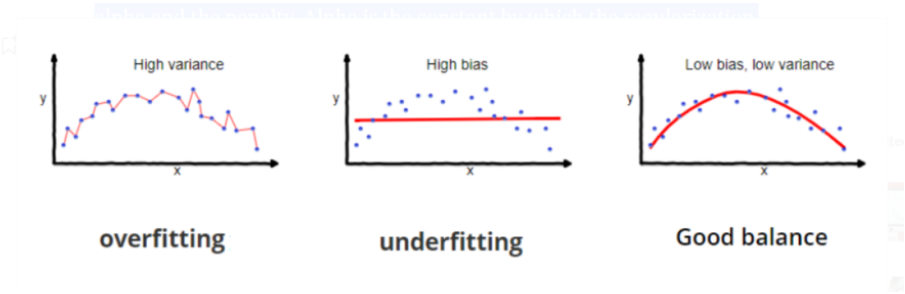


The Violin plot describes about probability density of data at differnet values for features is_duplicate and Partial_ratio The Distplot describes about the distributions for normalized Partial_Ratio have some overlap on the far right-hand side, i.e., there are quite a lot of questions with high Ratio similarity

7 Applying machine learning for classification

So now we have our data ready, we have cleaned and preprocessed it, extracted features and vectorized our text. We are now ready to go ahead and make some classifications. We have chosen to go with Logistic Regression, Linear SVM and Random Forest.

For logistic regression and linear SVM, the hyper parameters tuned were alpha and the penalty. Alpha is the constant by which the regularization term is multiplied. Higher alpha indicates a more powerful regularization. A stronger regularization helps in generalization but an overly powerful regularization will induce bias (under fit) in the model while a very weak regularization will inflate the variance (over fit) the model. Penalty is the type of regularization applied (L1, L2 or ElasticNet).



7.1 Building a random model (Finding worst-case log-loss)

Our key performance metrics 'log-loss' Function. Hence we need a random model to get an upper bound for the metric . A random model is one which when given x_i will randomly produce either 1 or 0 where both labels are equiprobable.

```
Log loss on Test Data using Random Model 0.88634182830674
```

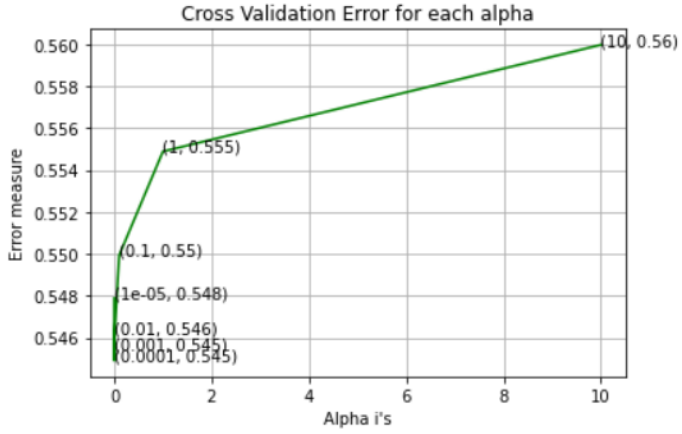
0.88 turns out to be the value of log loss for our random model. A 'decent' model for our problem will have a value of log loss which isn't close to 0.88.Note that have more data points for class 0 than for class 1.

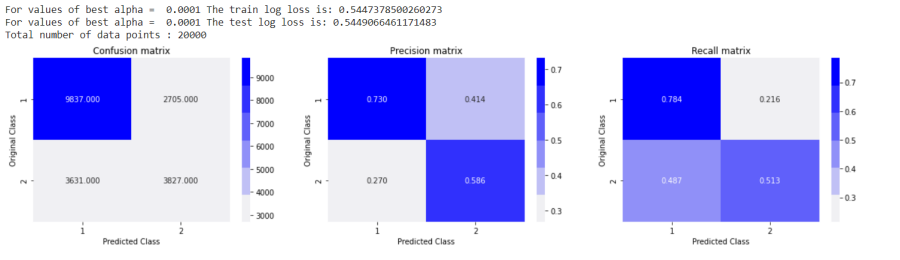
The metric that we are trying to optimize is the log-loss. This is the metric upon which the competition was judged and thus it is the one we are optimizing.

7.2 Logistic Regression with hyperparameter tuning

Since our data is neither high dimensional (eg 1000) nor low dimensional (eg 30), it lies somewhat in the middle with 221 dimensions . Hence we will be first trying Logistic regression model with hyper parameter tuning. We will be using grid search.

```
For values of alpha =  1e-05 The log loss is: 0.5478860713815852
For values of alpha =  0.0001 The log loss is: 0.5449066461171483
For values of alpha =  0.001 The log loss is: 0.5454714997569333
For values of alpha =  0.01 The log loss is: 0.5462431893665
For values of alpha =  0.1 The log loss is: 0.5499279372822296
For values of alpha =  1 The log loss is: 0.554884377689931
For values of alpha =  10 The log loss is: 0.5599692749598869
```



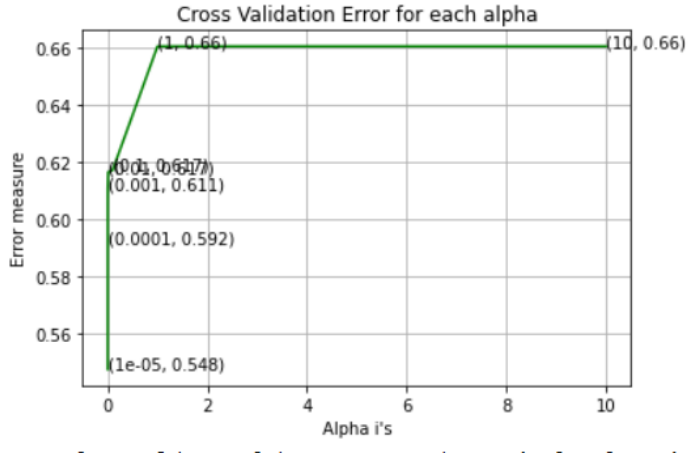


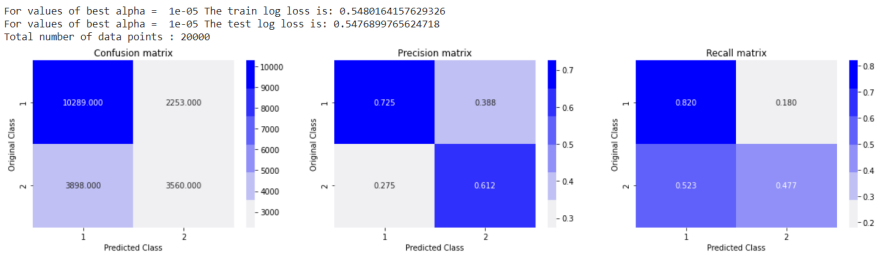
We got our best alpha(hyper-parameter) to be 0.0001 with a log loss of about 0.5447 on the test data which is slightly better than the random model. Few things we observed about this model are-

- The model is not suffering from over fitting since it's log loss on train and test data are quite close. It may be suffering from high bias or under fitting
- Model is able to predict class 0 decently but under performs in case of class 1.
- Precision for both classes is around 0.73 which is not very high.
- Recall for class zero is high , but for class 1 it is quite low.

7.3 Linear SVM with hyperparameter tuning

```
For values of alpha =  1e-05 The log loss is: 0.5476899765624718
For values of alpha =  0.0001 The log loss is: 0.5917890571175313
For values of alpha =  0.001 The log loss is: 0.6106920704947059
For values of alpha =  0.01 The log loss is: 0.6165439018918017
For values of alpha =  0.1 The log loss is: 0.6172424997693171
For values of alpha =  1 The log loss is: 0.6604906458187804
For values of alpha =  10 The log loss is: 0.6604906458187847
```



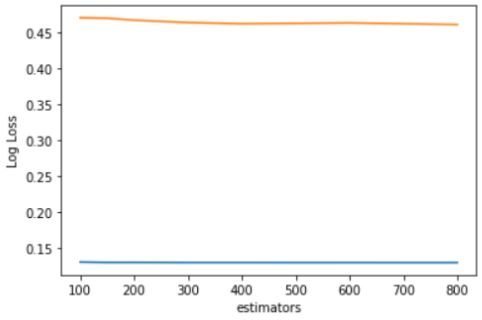


- Similar to logistic regression model ,linear SVM model is not suffering from over fitting since it's log loss on train and test data are quite close. It may be suffering from high bias or under fitting.
- Similar problem of precision and recall is with linear SVM.
- We don't observe any significant improvement in the model since log loss on the test data remain quite similar.

7.4 Random Forest With Hyperparamter Tuning

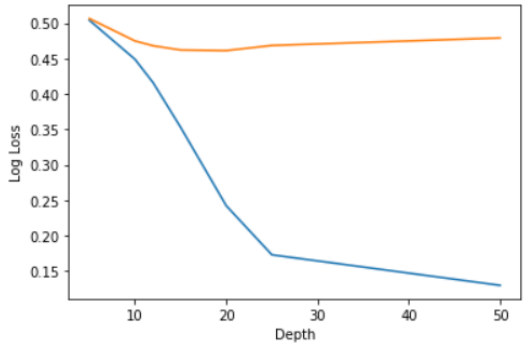
We have performed Random Forest With Hyperparamter Tuning for different values of estimators and depth Random Forest With Hyperparamter Tuning for different values of estimators we can say that it is suffering from over fitting since it's log loss on train and test data are not quite close

```
estimators =  100 Train Log Loss  0.1300801444855785 Test Log Loss  0.47113608752779035
estimators =  150 Train Log Loss  0.12951880187620887 Test Log Loss  0.47056750489402227
estimators =  200 Train Log Loss  0.12957082104641535 Test Log Loss  0.46782082368536587
estimators =  300 Train Log Loss  0.12935241514838905 Test Log Loss  0.4644738188027834
estimators =  400 Train Log Loss  0.12935631818737148 Test Log Loss  0.462843826971321
estimators =  600 Train Log Loss  0.12929783775523138 Test Log Loss  0.4639616379816283
estimators =  800 Train Log Loss  0.12927274553273052 Test Log Loss  0.461725756436557
Text(0, 0.5, 'Log Loss')
```



Random Forest With Hyperparamter Tuning for different values of depth we can say that it is not suffering from over fitting since it's log loss on train and test data are quite close . It may be suffering from high bias or under fitting.

```
Depth =  5 Train Log Loss  0.5041376951541853 Test Log Loss  0.5064421233256937
Depth =  10 Train Log Loss  0.4489897730670751 Test Log Loss  0.4748286636121834
Depth =  12 Train Log Loss  0.4157395260243374 Test Log Loss  0.4680201998480072
Depth =  15 Train Log Loss  0.3529913830662518 Test Log Loss  0.4620916472057321
Depth =  20 Train Log Loss  0.24229690560763426 Test Log Loss  0.46123790461814496
Depth =  25 Train Log Loss  0.1730297694846438 Test Log Loss  0.46849722969035384
Depth =  50 Train Log Loss  0.1299531054530457 Test Log Loss  0.4790842107894389
Text(0, 0.5, 'Log Loss')
```



8 Conclusion

Algorithms	HyperParameter	Train Log-Loss	Test Log-Loss
Logestic Regression	Alpha = 0.0001	0.5447	0.5449
Support Vector Machine	Alpha = 1.0	0.5518	0.5526
Random Forest	Estimator = 100	0.1300	0.4711
Random Forest	Depth = 5	0.5041	0.5064

In Conclusion we can say that our Dataset is performing good with Logestic Regression and Random Forest With Depth as hyperparameter Because Train loss log and Test Log loss are nearly close to each other.