QUORA QUESTION PAIRS

RAVIKANT SINGH
HTTPS://WWW.LINKEDIN.COM/IN/RAVIKANT-SINGH-7981AB115/
A PROJECT FOR EDWISOR.COM

Table of Contents

Contents 1

1. Problem Statement 2
2. Business Understanding 2
3. Data Understanding 2
4. Data Preprocessing (Text Mining) 3
    a. Case Folding 3
    b. Removing Numbers 3
    c. Removing Stopwords 3
    d. Removing Punctuations 3
    e. Stemming 3
    f. Stripping Whitespace 3
    g. Document Term Matrix 3
5. Feature Engineering 4
    a. Distance based score 4
    b. Similarity based score 4
    c. Calculating Sentiments 4
    d. Calculating tf-idf 4
6. Visualization 5
    a. Visualizing proportion distribution 5
    b. Density plot of target variable 5
    c. Wordcloud of most frequent words 6
7. Predictive Modelling 7
    a. Binomial Logistic Regression 7
    b. Random Forest 7
    c. Deep Learning 7
    d. Gradient Boosting 7
    e. Error Metric 7
    f. Conclusion 8

QUORA QUESTION PAIRS

1. Problem Statement

Build a binary **classification model** to predict which of the provided pairs of questions contain two questions with the same meaning.

2. Business Understanding

Over 100 million people visit Quora every month, so it's no surprise that many people ask similarly worded questions. Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question, and make writers feel they need to answer multiple versions of the same question.

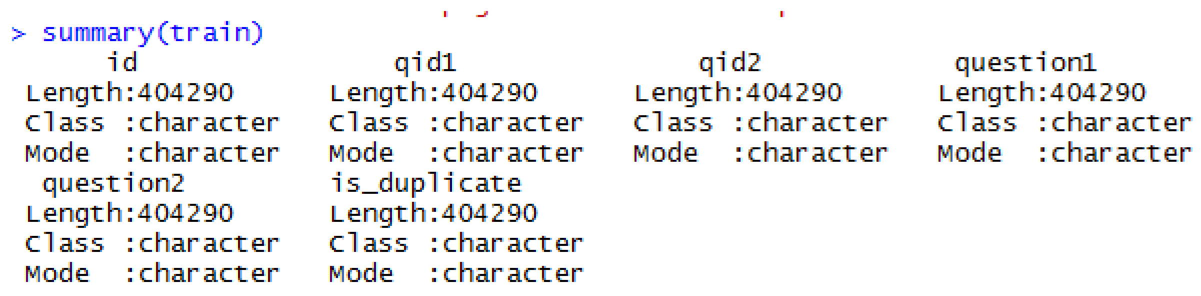
3. Data Understanding

We have been provided with 2 datasets i.e. Training and Test.
The training dataset contains, **404290** obs. of **6** variables-

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

The test dataset contains, **2345796** obs. of **3** variables

a. **test_id -** the id of a test set question pair
b. **question1, question2 -** the full text of each question



```
> summary(train)
      id                 qid1               qid2             question1        
 Length:404290      Length:404290      Length:404290      Length:404290     
 Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character  
  question2         is_duplicate      
 Length:404290      Length:404290     
 Class :character   Class :character  
 Mode  :character   Mode  :character  
```

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.

4. Data Preprocessing (Text Mining)

As we can see our data have two columns with text data, this means we need to perform 2 levels of data cleaning i.e. 1 for Question 1 and 2 for Question 2. We have cleaned our data by passing the train and test data through 2 functions- **match.calc()** and **weight.calc()**.
match.calc() takes each observation of Question 1 and 2 as argument and converts the data into corpus, then we convert the textual data into a corpus and then the corpus is cleaned using tm package (see below for more info on text mining/cleaning using tm package) and lastly we convert the data into a Document Term Matrix and perform feature engineering for model building.Weight.calc() performs a function similar to match.calc() but instead of working on individual rows , it takes the all observations. This was a brief summary of our data preprocessing lets dive a little deeper…

a. Case Folding

In the case folding stage we provide our corpus as input to tm_map() function with tolower() as the function argument. We convert all the text data to lowercase to avoid data inconsistency and improve linear similarity between similar words.

b. Removing Numbers

In this stage we remove all the numbers from the corpus using removeNumbers() function in tm_map().

c. Removing Stopwords

Stopwords are the commonly used words like “the”, “is” etc, these words often does not help in the model building but sometimes these could be helpful based on the problem statement and approach.
We have removed all the English stopwords from our corpus after evaluating that removing stopwords is helping our model to predict the output better.

d. Removing Punctuations

Punctuation marks like comma, fullstop are removed from the corpus as it does not help in prediction.

e. Stemming

Word can have different forms like opportunity , opportunities , these are reduced to its base form which helps in calculating score based on similar words. We use stemDocument() function of tm package for stemming.

f. Stripping Whitespace

At our last data cleaning step we use stripWhitespace() function of tm package to remove all the whitespace from our corpus that were created after we performed the above text cleaning functions on our corpus.

g. Document Term Matrix

We create a document term matrix using the DocumentTermMatrix() function, the DTM creates a matrix like form all the documents as rows and terms as columns(or vice versa using TermDocumentMatrix()).
We have created DTM in both functions match.calc() and weight.calc() for creating feature engineered variables(Next Topic).

5. Feature Engineering

The predictive power of a model can be increased significantly if supplied with the right feature engineered variable inputs. In Quora question pairs project, we are supplied with textual data as inputs which in itself cannot help much in prediction. We have created different feature engineered variables to improve the predictive power of our model. Each of these variables are mentioned in detail below.

a. Distance based score

We have used the stringdist package for calculating the approximate string matching and string distance. We have calculated Levenshtein distance, Jaccard distance and soundex distance(default) for each observation (distance between question 1 and 2) using **stringsim()** function.

b. Similarity based score

We have created a score based on the similar words between both questions. First we extracted the similar words from both questions and then we created a formula for calculation score i.e.
(2 * count of similar words) / count of total words in question 1 and 2.

c. Calculating Sentiments

We have calculated the total positive and negative sentiments of question 1 and question 2 and provided as input to our model, the importance of these variables are not high but they are still contributing a small amount to our prediction model, hence keeping these variables.

d. Calculating tf-idf

We have calculated the term frequency – inverse document frequency of both corpus (question 1 and 2) using weighting = weightTf parameter in DocumentTermMatrix.
The total tf-idf are then provided as an input to our predictive model.

NOTE :- a,b,c were calculated under match.calc() function, while d was separated calculated under weight.calc() function.

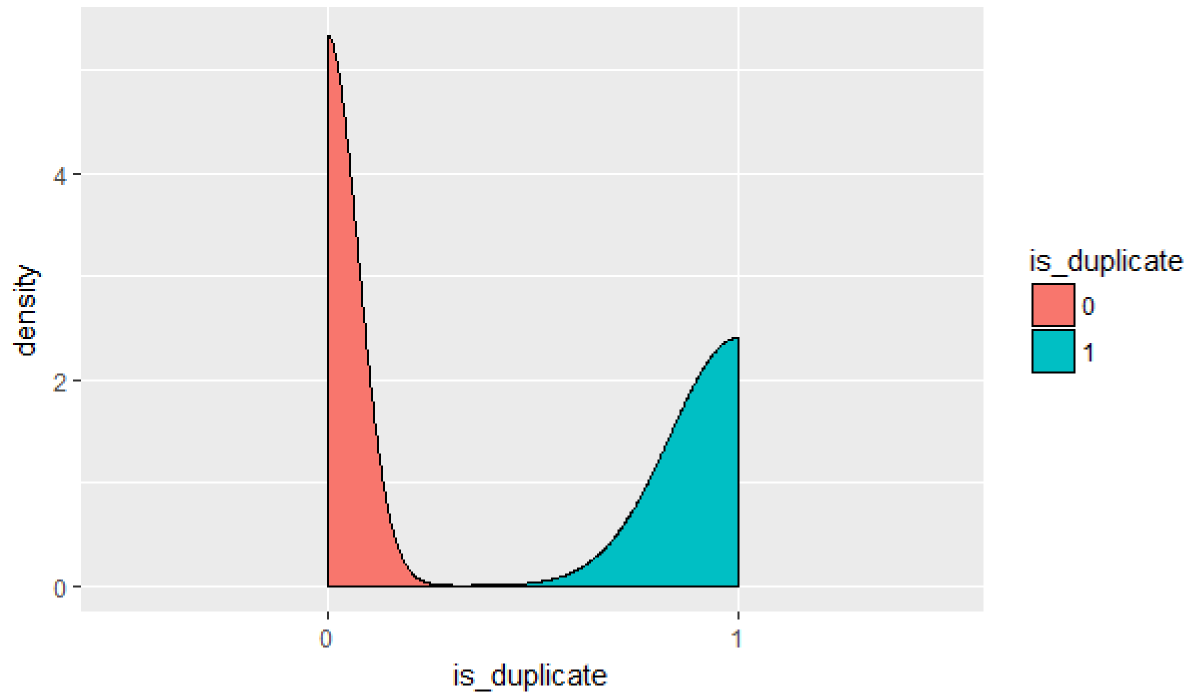
6. Visualization

a. Visualizing proportion distribution

```
        0         1
0.6308021 0.3691979
```

Data is divided into 63 to 36 percent (0 to 1)

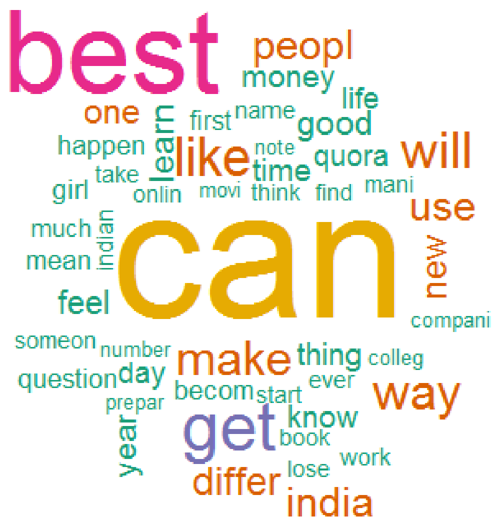
b. Density plot of target variable



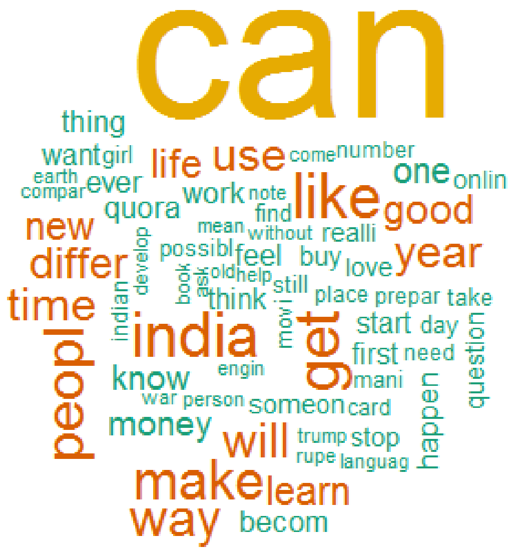
As we can see the density of not similar question is higher than similar questions.

c. Wordcloud of most frequent words

Question 1



Question 2



7. Predictive Modelling

For our modelling we have initially considered Binomial Logistic Regression, Random forest, Deep Learning and Gradient Boosting models using h2o cluster for fast model processing. From our initial model execution, we found out all of our models are working better with numerical data as compared to factor variables.
Our final two models were Random Forest and Gradient Boosting model.
NOTE- We have used different seed values for our random sampling to analyze the most suitable model and the average accuracy we are receiving on all the models.

We have finalized our Gradient Boosting Model to be the best suitable model for our current problem statement.

a. Binomial Logistic Regression

All our feature engineered variables are in numerical format, and our regression model was able to predict the target variable with accuracy between (65 – 70) percent, mean accuracy was 68 percent.

b. Random Forest

Being a tree based algorithm, we were expecting random forest to perform better than our regression model, and it did. Random forest was able to predict the outcome with accuracy between (69 – 73) percent, mean accuracy was 70 percent.

c. Deep Learning

Here we have tried to channel the power of the neural networks using h2o's native deeplearning function. We have provided hidden layer as 50 , 50 , which basically places 2 layers of 50 nodes each.The performance of deep learning model was mostly fluctuating between the accuracy (68 – 74) percent, although we can say the mean accuracy was 70 percent.

d. Gradient Boosting(GBM)

Best performance noted was from our gradient boosting model, as the gradient boosting is a tree based algorithm which focuses on making the weak predictors strong. The accuracy we received were between (70 – 74) percent, with a mean accuracy of 72 percent.

e. Error Metric

We have used confusionmatrix for evaluating our model's accuracy and selecting the most suitable model for our problem statement.

Insights on evaluation –
We have noticed that while predicting some values although having high probability of being 0 still was selected as 1 and vice versa, so we set a up a threshold of 0.5 below which value will be 0 and above which value will be 1. This has increased the accuracy of our model by (2-4) percent.

f. Conclusion

To predict whether two questions are duplicate or not , we have worked on our data to create some interesting feature engineered variables like calculating distance, calculating a score based on word similarity, the total frequency of tf-idf and creating positive and negative sentiments.

We have evaluated our different models using different seed for random sampling and the analysis of accuracy and misclassification error concludes that **Gradient Boosting Algorithm** is the most **suitable model** for our problem statement.

Gradient boosting is known for making weaker variables **strong** by combining their prediction powers (variance explained by each weak variable). For our problem statement GBM is predicting the outcome with mean accuracy of 72 percent.

Thank You.