Machine Learning Engineer Nanodegree

Capstone Project

Sumit Chatterjee, June 23rd, 2019

1. Definition

1.1 Project Overview

- It is a site where people share their views with each other regarding a particular question and give a correct answer regarding a question.
- It's a multi-faceted kind of site, on one hand there is the academic side where people generally answer questions which are complex and also, academic where answers where they may have multiple correct answers. Apart from this we also have the Meme side which is self-explanatory. People post memes for fun.
- Questions about other subjects which doesn't include education, academia, religion etc. This could mean a wide range of question topics like celebrity gossips, sport related, conspiracy theories, health benefits of certain foods, exercises etc. Quora also supports new writers.
- The application currently is using a Random Forest model to identify duplicate questions. With this NLP problem by applying advanced techniques to classify whether question pairs are duplicates or not will make it easier to find accurate answers to questions resulting in an improved experience for Quora writers and readers as, effectively detecting duplicate questions not only saves time for seekers to find the best answer to their questions, but also reduces the effort of writers in terms of answering multiple versions of the same question.

1.2 Problem Statement

The goal is to create an algorithm that will help us to know which of the provided pairs of questions contain two questions with the same meaning.

1. Download and preprocess the data to get a two set of application.
2. One using NLP token format tfiidf and another with tfidf-Word2vec.
3. Train a classifier that can determine if two question have the same intent or not.
4. Check the corresponding algorithms a proper idea on which algorithm to run on which scenario.
5. Identify questions that are already asked on Quora which are duplicates.
6. This could be useful to instantly provide answers to questions that have already been answered.
7. The main intention is to predicting whether a pair of questions are duplicates or not.

2. Data Exploration

2.1 Data Overview

The dataset for this project is available on Kaggle.com. Main Source of Data and other information: https://www.kaggle.com/c/quora-question-pairs (https://www.kaggle.com/c/quora-question-pairs)

We have 4 data fields such as:

- 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.

2.2 A Deeper Dive

The training data set contains 404290 question pairs, whereas the test data set has 2345790 question pairs.

Each data entry consists of the ID of the question pair, the unique IDs and full text of each question, as well as the target variable mentioning whether the two questions are duplicates or not.

The question pairs consists with wide range of topics which includes technology, entertainment, politics, culture, reigion, philosophy etc. Few questions also include special characters such as mathematics symbols and foreign language characters. Also, to note that the duplicate labels in the training set are provided by human experts and are inherently subjective, since the true meaning of sentences can never be known with certainty. As a result, the labels on the training data set represent a reasonable consensus and are considered ground truth, but are not 100% accurate, i.e. the dataset is noisy.

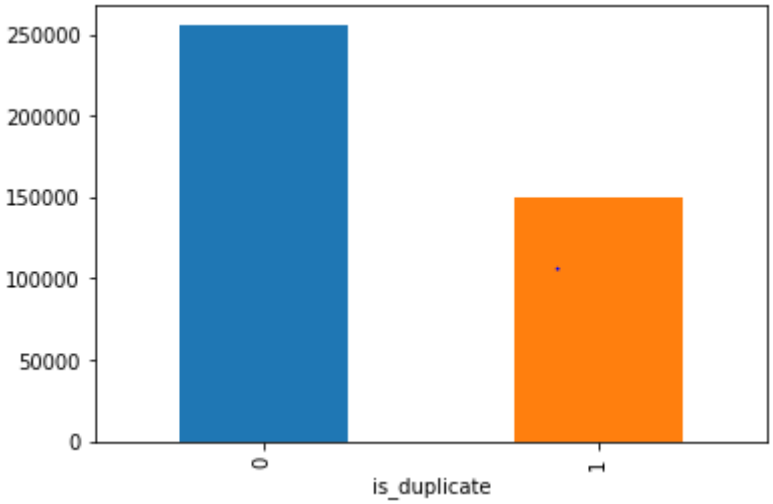
A quick look at the datset overall:

Data columns (total 6 columns):

- id 404290 non-null int64
- qid1 404290 non-null int64
- qid2 404290 non-null int64
- question1 404289 non-null object
- question2 404288 non-null object
- is_duplicate 404290 non-null int64
- memory usage: 18.5+ MB

2.3 Exploratory Visualization

- Number of duplicate and non-duplicate questions in a bar plot !

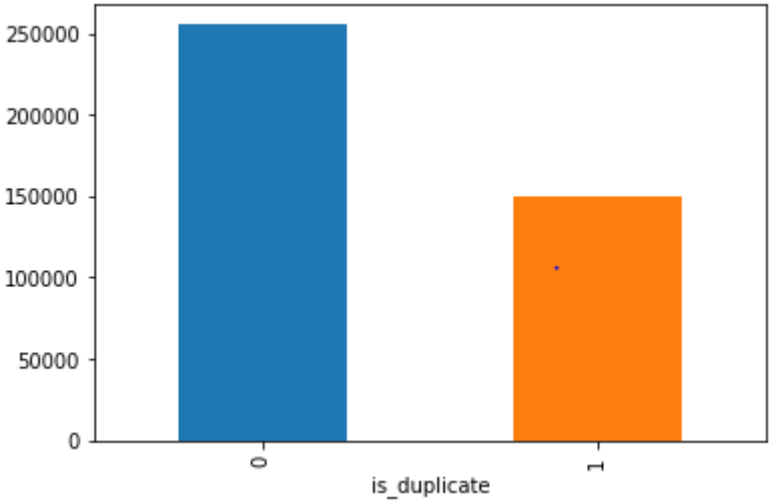


- We have around 63.08% duplicate question pairs and 36.92% non - duplicate question pairs.

With some basic statistical calculations we can easily find out the followings:

- Number of unique questions: 537933
- Number of unique questions that appear more than one time is 111780
- Maximum number of times a single question repeated is 157 (Below graph can prove it)

- Number of occurance of questions



- The one point that shows the maximum number of times a single question repeated is 157

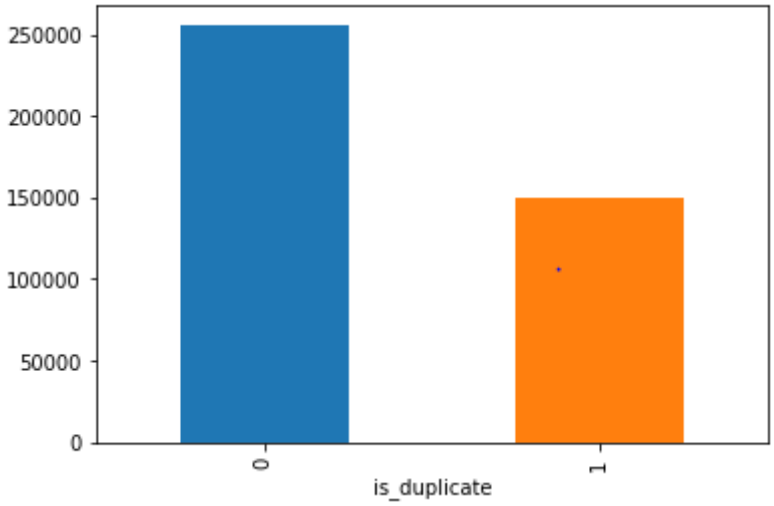
3. Metrics

3.1 Accuracy is a common metric for classifiers; it takes into account both true positives and true negatives with equal weight. Accuracy = (dataset size true positives + true negatives)/ dataset size

3.2 Confusion Matrix also, helps us for a visualization of the performance of an algorithm.

We're also using confusion matrix to determine the performance of the algorithms.To understand this we need to understand few basic terminologies:

```
* true positives (TP): These are cases in which we predicted yes (they have the
 disease), and it's actually an yes.
* true negatives (TN): We predicted no, and it's actually an no.
* false positives (FP): We predicted yes, but it's actually a no. (Also known as
a "Type I error.")
* false negatives (FN): We predicted no, but it's actually an yes. (Also known a
s a "Type II error.")
```



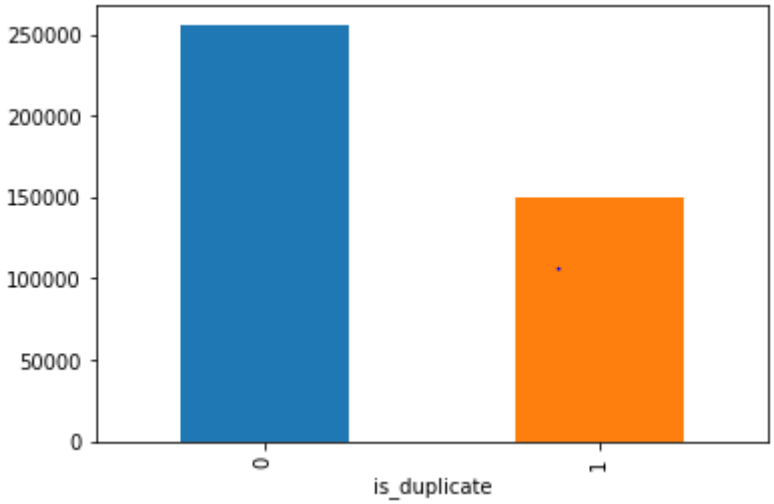
From the definition above it is clear how we are actually getting the ratio. Sometimes we think that precision and recall both indicate accuracy of the model. While that is somewhat true there is a deeper there is distinct meaning of each of these terms.

Precision means the percentage of your results which are relevant. Recall means the percentage of total relevant results correctly classified by your algorithm. Later we've explained how we got our precision and recall of our model.

3.3 Log-Loss i.e. Logarithmic loss measures the performance of a classification model where the prediction input is a probability value between 0 and 1. The goal of our machine learning models is to minimize this value. A perfect model would have a log loss of 0. Log loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high log loss. logloss is especially relevant in cases with imbalanced data or in case of unequally distributed error cost.

4.Analysis

The model that have been followed is depicted in the below image:



4.1 Feature Extraction

In machine learning feature extraction starts from an initial set of measured data and builds derived values intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations.

In this we've done feature extraction from all the column values provided. We've created few functions which does the followings:

4.1.1 Step - I

- 1. normalized_word_Common: It returns the number of common unique words in Question 1 and Question 2.
- 1. normalized_word_Total: It returns the sum of total num of words in Question 1 and Total num of words in Question 2.
- 1. normalized_word_share: This is the A ratio between first two functions.

  From here, we quickly created few feature vectors word_share, word_common, freq_q1+q2, freq_q1-q2 to prepare our first set of records. This we have stored it in a 'csv' file while removin any blank values.

4.1.2 Step - II

* 1.In this step we used standard NLTK libraries to remove the stopwords from the dataset. This commonly contains replacing words like "i'm" to "i am", "she's" to "she is" etc.

* 2.We've also implemented some structural measure changes that complies with token.
    When we split a sentence based space, that is called "token"
    Word: Any token which isn't a stopword

    • Exact match: This measure calculates the number of keywords (i.e. words other than stop words) of one question that exactly match with keywords in the second question.
    • Corrected match: This measure returns the number of keywords in question one that exactly match with the spell corrected keywords in the second question.
    • Synonym match: This measure returns the number of keywords in question one that match with synonyms in question two.

* 3. Fuzzy string matching features: Some fuzzy string matching measures such as Q Ratio (Quick ratio comparison between two strings), W Ratio (measure of the sequences' similarity between 0 and 100 using different fuzzy matching algorithms), partial token sort ratio, partial token set ratio, token set ratio, token sort ratio.

* 4. We've also created new features that will hold last word of both question is same or features that will hold the first word of both question.

- 1. We've also created a function "extract_features" and using this function we've created few features that has been described as below:

  "cwc_min" = Gives the ratio of common_word_count to min lenghth of word count of Q1 and Q2
  "cwc_max" = Gives the ratio of common_word_count to max lenghth of word count of Q1 and Q2
  "csc_min" = Gives the ratio of common_stop_count to min length of stop count of Q1 and Q2
  "csc_max" = Gives the ratio of common_stop_count to max length of stop count of Q1 and Q2
  "ctc_min" = Gives the ratio of common_token_count to min lenghth of token count of Q1 and Q2
  "ctc_max" = Gives the ratio of common_token_count to max lenghth of token count of Q1 and Q2
  "last_word_eq" = Checks if the last word of both questions are equal or not
  "first_word_eq" = Checks if the first word of both questions are equal or not
  "abs_len_diff" = Checks the absolute length differences for Q1 and Q2
  "mean_len" = Checks the average token length of both Questions Q1 and Q2

- 1. Post this we have applied fuzz functions as described them before.
- 1. Post this like before kept them in a file where now we have 21 data columns which consists of the previous feature vectors along with these new feature vectors as defined above.

4.2 Using of Word-to Vector(W2v)

W2v is based on Word Embedding. Word embedding is nothing fancy but methods to represent words in a numerical way. More specifically, methods to map vocabularies to vectors. One of the method of it is W2v.

We can call it as given any “context word” based on which we want to train a model such that the model can predict a “target word”, one of the words appeared within a predefined window size from the context word.

We've applied W2v on the Questions 1 & 2 and merged them and used them to create our final model feature dataset. We've also moved the data to a sqlite db for further convenince of te data and as we don't have to run the whole code sheet to run the model again.

- Before applying any machine learning algorithm we need to convert any strings to numerics.
- Once conversion is done we've split the data into training and testing datasets.
- As discussed above we'll also create a function to define our metric confusion metrics to understand the difference between our predicted and already predicted data. This function we'll apply to all our algorithm once done with the prediction.

4.3 Model Structures and performance

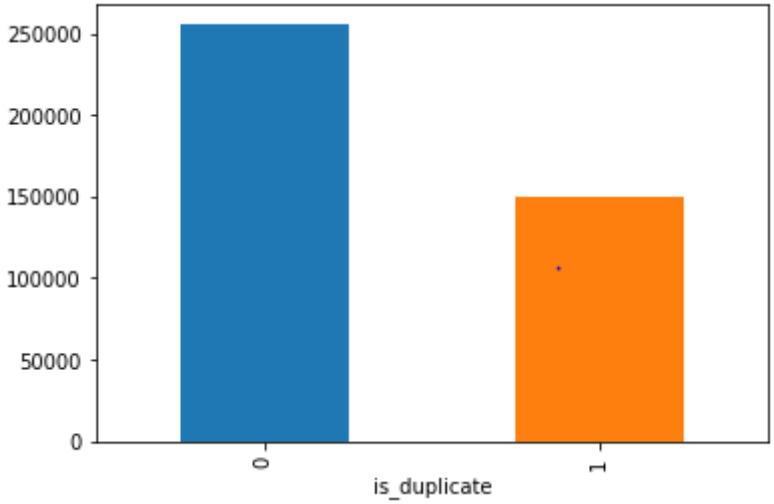
We've created 3 model structures, based on our computational power and time.

- A random dumb model;
- Logistic Regression;
- XGBoost.

5. Methodology

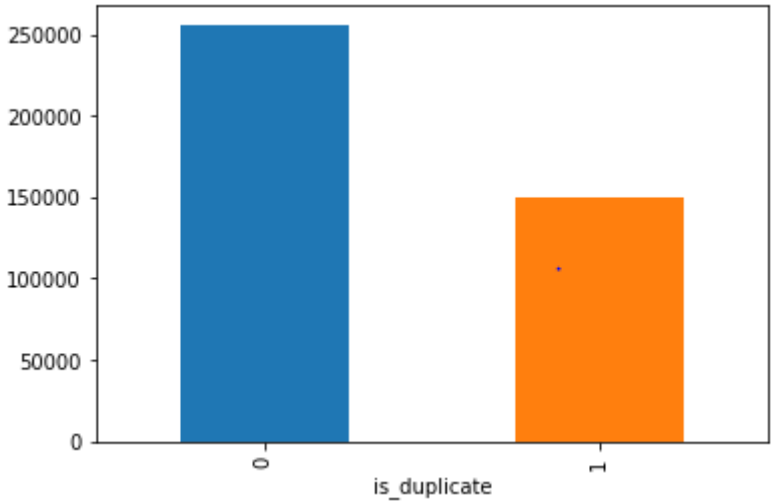
We've split the entire process as proposed int he proposal into several structures. First step, we've done data preprocessing as described above. We've taken the feature extraction process and covered through our dataset and made adjustements to apply them to our machine learning models.

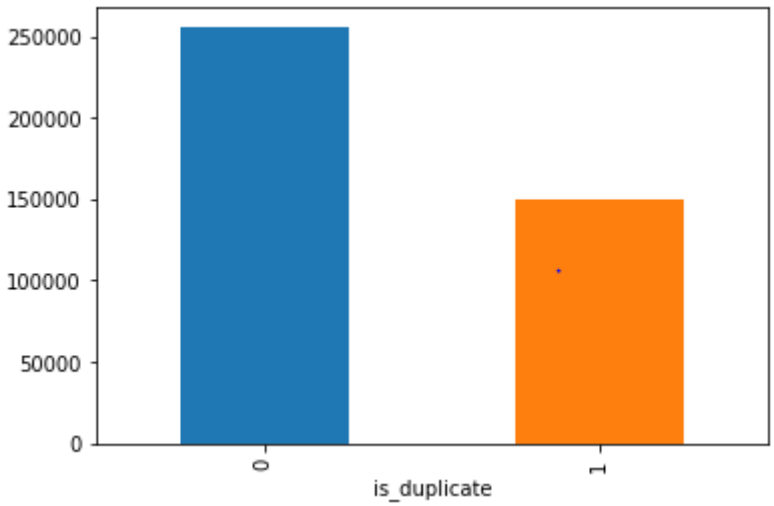
5.1 In the random model we're not taking any parameter into the consideriation and tried plotting the confusion_matrix and got an log loss of 74%. we've also got our precision and recall model plot for this.



5.2 In the Logistic Regression model

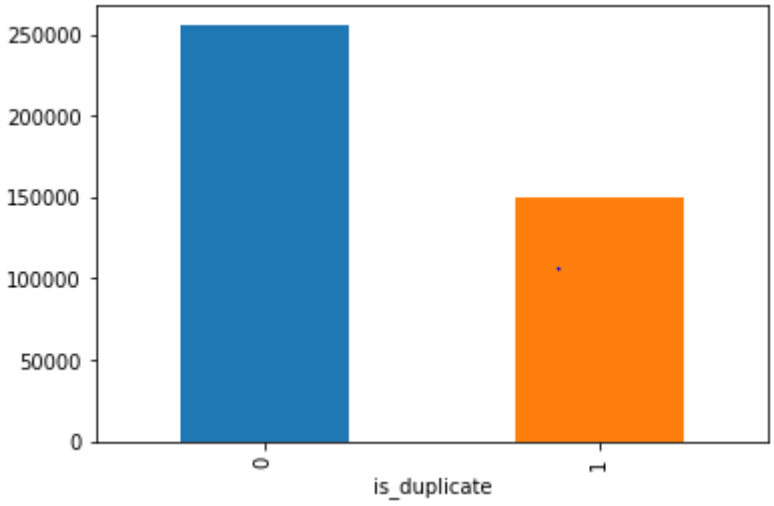
we're taking an SGD classifier and provided hyperparameters to fit the training and testing data and got our log loss for the model. If we notice the the rate of log loss based on the alphas. Based on this model we've got an log-loss of 0.45





5.3 In the XGBoost model

If we notice the the rate of log loss based on the alphas. Based on this model we've got an log-loss of 0.35. With hyperparameter tunings we'd get an log loss of 0.35. The hyperparameters that we've applied here we got it from defining a hyperparameter tuning function where we're govong a set of hyperparameter option and applied K fold cross validation to use them to get a better hyperparameter. The code set is intensive on my local machine and I have used my office desktop to run it and it took almost 4 hours to complete that.



6. Conclusion

In [3]:

```
from prettytable import PrettyTable
ptable = PrettyTable()
ptable.title = "Model Comparision"
ptable.field_names = ['Model Name', 'Tokenizer','Hyperparameter Tunning', 'Test Log Los
s']
ptable.add_row(["Random","TFIDF Weighted W2V","NA","0.89"])
ptable.add_row(["Logistic Regression","TFIDF Weighted W2V","Done","0.50"])
ptable.add_row(["XGBoost","TFIDF Weighted W2V","NA","0.35"])
ptable.add_row(["XGBoost","TFIDF Weighted W2V","Done","0.34"])
ptable.add_row(["\n","\n","\n","\n"])
ptable.add_row(["Random","TFIDF","NA","0.74"])
ptable.add_row(["Logistic Regression","TFIDF","Done","0.45"])
print(ptable)
```

```
+---------------------+--------------------+------------------------+-----
----------+
|      Model Name     |     Tokenizer      | Hyperparameter Tunning | Test
Log Loss |
+---------------------+--------------------+------------------------+-----
----------+
|        Random       | TFIDF Weighted W2V |           NA           |
0.89     |
| Logistic Regression | TFIDF Weighted W2V |          Done          |
0.50     |
|       XGBoost       | TFIDF Weighted W2V |           NA           |
0.35     |
|       XGBoost       | TFIDF Weighted W2V |          Done          |
0.34     |
|                     |                    |                        |
|
|                     |                    |                        |
|
|        Random       |       TFIDF        |           NA           |
0.74     |
| Logistic Regression |       TFIDF        |          Done          |
0.45     |
+---------------------+--------------------+------------------------+-----
----------+
```

6.1 Scope & Chances

However, that being said if given time I'd like to perform this entire project on the basis of Deep learning, as we know most of these NLP technologies are powered by Deep Learning. If we get larger amounts of training data,faster machines and multicore CPU/GPUs. With this our algorithms can be developed with advanced capabilities and improved performance.

I'd like to use other Word@vec models such as Skip-Gram and Continuous Bag of Words, apart from this I think we can also be greatly benefit by doing this project Long short-term memory(LSTM) networks. A great source and starting point would be this article: http://xiaojizhang.com/files/quora-question-pairs.pdf (http://xiaojizhang.com/files/quora-question-pairs.pdf)

In this we've tried lowercasing the data, used stemming, stopword removal and also used porterstemmer.

Apart from this we can also implement 'Lemmatization' which is very similar to stemming, where the goal is to remove inflections and map a word to its root form. The only difference is that lemmatization tries to do it a proper way i.e. it doesn't chop words off, it actually transforms words to the actual root. Example the word "excellent" may map to "very good".

Apart from this we can also do 'Normalization'. It's the process of transforming a text into a standard form. For example, the word "gooood" and "gud" can be transformed to "good". Another example is mapping of near identical words such as "stopwords", "stop-words" and "stop words" will just go to "stopwords".

As far other ML technique goes a linear SVM would be another good model to select to do check the performance of the algorithm.

6.2 Briefly

Tokenizer: TFIDF Weighted W2V

```
First we have applied simple Random Model which is a dumb model in sense, which
 gives the log loss of 0.89 which means, other models has to produce less than
 0.89.
After that we have applied Logistic Regression on less than 100K dataset with hy
perparameter tuning, which producs the log loss of 0.52, which is significantly
 lower than Random Model.
We then applied XGBoost Model on less than 100k dataset with no hyperparameter t
uning, which produces the log loss of 0.35, which is significantly low.
Lastly, we applied XGBoost Model on same 100k dataset with hyperparameter tunin
g, which produces the log loss of 0.34, which is slightly lower than XGBoost Mod
el with no hyperparameter tuning. We know that on high dimension dataset 'XGBoos
t' does not perform well, but it did farely well in the above dataset because of
low dimension of 794. Whereas 'Logistic Regression' and performs moderately on l
ow dimension data.
```

To check on this we performed the same task on around 400k dataset, and we should get better results as compared to above models.

Tokenizer: TFIDF

```
First we have applied simple Random Model which is a dumb model in sense, which
 gives the log loss of 0.74 which means, other models has to produce less than
 0.74.
After that we have applied Logistic Regression on 400K dataset with hyperparamet
er tuning, which producs the log loss of 0.45, which is significantly lower than
Random Model. Also, lower than previous LR model.
```

Therefore we can say that on low dimesion data,we will use hyperparameter tuned 'XGBoost' model and for high dimension data we will use 'Logistic Regression'.

7. References

https://en.wikipedia.org/wiki/Quora (https://en.wikipedia.org/wiki/Quora)
https://www.kaggle.com/c/quora-question-pairs (https://www.kaggle.com/c/quora-question-pairs)
https://en.wikipedia.org/wiki/Textual_entailment (https://en.wikipedia.org/wiki/Textual_entailment)
https://www.linkedin.com/pulse/duplicate-quora-question-abhishek-thakur (https://www.linkedin.com/pulse/duplicate-quora-question-abhishek-thakur)
https://www.kaggle.com/c/quora-question-pairs/overview (https://www.kaggle.com/c/quora-question-pairs/overview)
https://www.kaggle.com/c/quoraquestion-pairs/overview (https://www.kaggle.com/c/quoraquestion-pairs/overview)
http://xiaojizhang.com/files/quora-question-pairs.pdf (http://xiaojizhang.com/files/quora-question-pairs.pdf)
http://xiaojizhang.com/files/quoraquestion-pairs.pdf (http://xiaojizhang.com/files/quoraquestion-pairs.pdf)
https://en.wikipedia.org/wiki/Confusion_matrix (https://en.wikipedia.org/wiki/Confusion_matrix)
https://en.wikipedia.org/wiki/Confusion_matrix (https://en.wikipedia.org/wiki/Confusion_matrix)
https://en.wikipedia.org/wiki/Type_I_and_type_II_errors#False_positive_and_false_negative_rat (https://en.wikipedia.org/wiki/Type_I_and_type_II_errors#False_positive_and_false_negative_rat)
https://en.wikipedia.org/wiki/Type_I_and_type_II_errors#False_positive_and_false_negative_ra (https://en.wikipedia.org/wiki/Type_I_and_type_II_errors#False_positive_and_false_negative_ra)
https://en.wikipedia.org/wiki/Feature_scaling (https://en.wikipedia.org/wiki/Feature_scaling)
https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/ (https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/)