Machine Learning Engineer Nanodegree

Capstone Project

I. Definition

Project Overview

Data quality, deduplication, and data management have long been critical functions for organizations and enterprises to use information technology resources efficiently. With an increasing amount of data created over time, it's ever more important for companies to focus on proper management of their information, in all aspects. In the domain of master data management, a key component is identifying unique or master records so that a common reference point is available for sharing and use by enterprise applications and end users. This allows the organization and it's users to minimize confusion that duplicate records cause, and to focus on data quality for use downstream.

A second way organizations and businesses can make sense of their data is through Natural Language Processing (NLP). In the domain of NLP, the goal is identify patterns, insights, and/or meaning through the processing of natural language (speech, signing, writing). This is a vast domain, and it serves quite a few purposes. Example tasks for solving NLP problems include sentiment analysis, topic segmentation, part-of-speech tagging, and even speech recognition (among many others).

This Capstone Project implements a solution to the Quora Question Pairs Competition from Kaggle. The Competition asks: "Can you identify question pairs that have the same intent?". The goal of this project was just that, to identify question pairs as being duplicate or not based on their intent. Kaggle and Quora have made available approximately 3 million question pairs as part of their training and testing datasets allowing the Kaggle community to create algorithms to predict whether or not a pair of questions was duplicate or not.

Problem Statement

Per Quora’s description on the Kaggle Competition page, the problem is defined as follows:

> ***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. Quora values canonical questions because they provide a better experience to active seekers and writers, and offer more value to both of these groups in the long term.***
>
> ***The goal of this competition is to predict which of the provided pairs of questions contain two questions with the same meaning.***

One solution, that Quora employs to a certain degree, is the use manual human tagging. This is time-consuming, inefficient and costly. A second solution, as shared by Kaggle, is that Quora currently uses a Random Forest model to identify duplicate questions.

The approach that was taken to address this problem as part of this project was to develop a classification algorithm to predict whether or not a pair of questions is duplicate or not. This involved a number of steps, including:

- **Data Preprocessing & Cleaning:**Includes removing punctuation, stop words, and word stemming
- **Generating New Features:**Deriving new features from the dataset as inputs to the classification algorithm
- **Training and Testing Classification Algorithms to make Predictions:** Leveraging the training data and newly generated features, classification algorithms such as Random Forests, Logistic Regression [1]and XGBoost [2]were used to generate predictions.

In terms of measuring and quantifying the success of my approach, the log loss function was used to quantify the accuracy of predictions on the training and test datasets. These predictions were then be compared against a benchmark algorithm using the log loss scoring metric.

Metrics

In this particular Kaggle competition, submissions are evaluated on the logarithmic loss (or log loss) between the predicted values and the ground truth. This is a good evaluation metric as it quantifies the accuracy of a classifier by penalizing false classifications. Predictions that are correct or incorrect are rewarded or punished proportionally to the confidence of the prediction.

According to fast.ai [3], Log Loss is defined as follows:

> ***Logarithmic loss (related to cross-entropy) 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.***

Here's an example of the mathematical representation of the log loss function:

$$V(f(\vec{x}), y) = \frac{1}{\ln 2} \ln(1 + e^{-yf(\vec{x})})$$

The log loss metric was chosen over several other scoring metrics:

- **Accuracy:** Accuracy (an evaluation metric familiar to all) is the number of correct predictions made as a ratio of all predictions. In comparing to accuracy, in order to calculate Log Loss the classifier must assign a probability to each class rather than simply yielding the most likely class. This provides an added benefit by providing us a 'level of confidence' in relation to a given prediction.
- **F1 Score:**The F1 score is defined as "the harmonic mean (a kind of average) of precision and recall. Precision is the percentage of all positive predictions that are in fact actual positives. Recall is the fraction of all actual positives that are predicted positive" [5]. Based on the F1 definition, Recall is undefined when the number of actual positives is 0, and Precision is undefined when the actual negatives are 0. While this likely wouldn't pose a problem for binary classification problems, when working with multi-label problem, this can pose an issue. Per the "Failing Optimally – Data Science's Measurement Problem" on KDNuggets.com [5], many implementations actually set this undefined value to 1 to

work around this. Additionally, since F1 is a balance between precision and recall, if there is a class imbalance in the dataset, it is possible to obtain varying results across systems. [6]

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\mathrm{recall}} + \frac{1}{\mathrm{precision}}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

II. Analysis

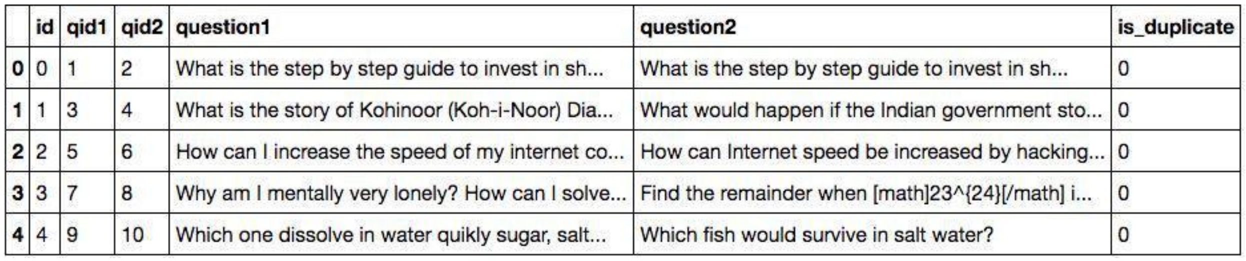
Data Exploration

Quora has provided a training and test datasets containing the full text of question pairs, and whether or not these two questions were considered as duplicate. The train.csv.zip dataset contains following 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.

The test.csv.zip dataset contains three fields: **test_id**, **question1**, and **question2**, but does not contain an **is_duplicate**column. Per the Kaggle competition rules, five (5) submissions can be made per day, upon which the submission results are scored.

Here’s the first five (5) records from the training dataset for review:



	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

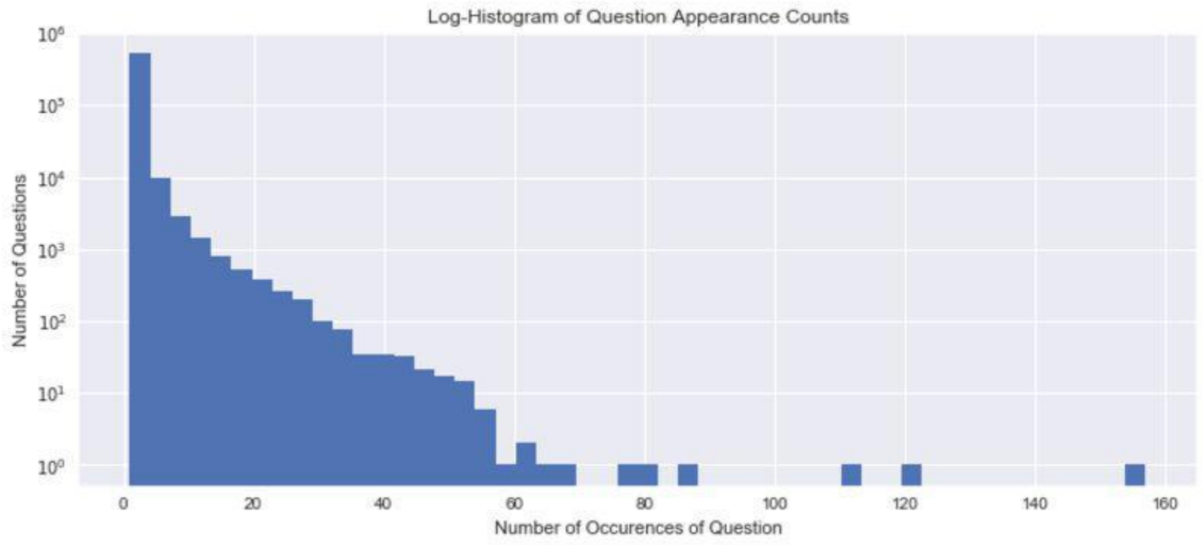
The training dataset contains 404,288 rows and 6 columns, and the test dataset contains 2,345,796 rows, and 3 columns. Within the training dataset, there are 255,027 (63.08%) non duplicates, and 149,263 (36.92%) duplicates. After checking for null values, there are 2 null values present in the training dataset, and 6 null values in the test dataset.

The submission that Kaggle scores should be provided in two columns: **test_id**, and **is_duplicate**. [6]

Exploratory Visualization

Prior to training and testing predictions on the data, new features were generated from the **question1**and **question2**text fields in order to extract patterns from the data. This means that different features need to be generated from the data to determine how much variance is explained in the data.

In reviewing individual questions that appear multiple times across both the testing and training datasets, there is a total of 537,933 unique questions, and 111,780 of these questions appear more than once.



As visible in the histogram above, the vast majority of questions either appear once or a few times. Additionally, there is a very small number of questions that appear up to around ~50 times, and three questions that appear over 100 times.

Algorithms and Techniques

For this classification problem, the algorithms that were used include Logistic Regression and XGBoost.

Logistic Regression: Per the Scikit-Learn documentation, “Logistic Regression is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier”[1]. Logistic regression is similar to linear regression with the primary difference being that the predicted values in logistic regression are the probability that it belongs to a given class (as opposed to only the raw class prediction). By providing this probability, logistic regression provides an additional indication of confidence that a given prediction is accurate.

In summary: Linear regression models a continuous response as a linear combination of the features:

$$y = \beta_0 + \beta_1 x$$

Whereas, logistic regression models the log-odds of a categorical response being “true” as a linear combination of the features [16]:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

The default parameters that were used for Logistic Regression include:

- C=0.1
- Solver = 'sag'

XGBoost: Per MachineLearningMastery.com, “XGBoost is an implementation of gradient boosted decision trees designed for speed and performance” [17].

- A decision tree [18]uses a tree-like graph or model as a predictive model to make observations about an item in order to make conclusions about the item’s target value. Decision trees have been around for decades and is the basis for many of the most powerful models available.
- Gradient boosting is a technique which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in stages (similar to other boosting methods), and makes generalizations by optimizing a differentiable loss function [19].

XGBoost (or “Extreme Gradient Boosting”) has become a very popular tool for data scientists. According to the XGBoost documentation site [7], it’s been used to win many machine learning challenges, it’s used in production by multiple companies and it’s flexible, supports distributed training on the cloud, and is optimized for performance.

The default parameters that were used for XGBoost include:
- objective = 'binary:logistic'
- eval_metric = 'logloss'
- eta = 1
- max_depth = 4

Benchmark

Currently, Quora uses a random forest model to identify duplicate questions. For the benchmark model in this project, it would be logical to use a random forest model as well.

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 (and it’s also what Kaggle scores submissions on).

III. Methodology

Data Preprocessing

After loading the training dataset and test dataset into python, the first step that was taken was to replace any null values in both datasets with some default text for empty questions. Two questions in the training dataset and six questions in the test dataset were replaced with the generic text “empty”.

Next, after performing data exploration, the questions were cleaned by removing punctuation and stop words. The justification for removing punctuation is that this may allow the features that are generated to identify similarities in questions if a word or characteristic exists across different questions.

Stop words are common words like “the” that appear frequently in text which may not provide contextual value to the classifier. The justification for removing stop words as they are often deemed as irrelevant. There are sets of English language stop words available in the Scikit Learn package (includes 318 words) and in the NLTK package (includes 153 words). Another approach is to define a custom set of stop words that is specific to the domain or problem.

The next data preprocessing task that was performed was stemming using the Porter stemmer. The Porter stemmer is “a process for removing the commoner morphological and inflexional endings from words in English” [15].

Next, features were generated to use as inputs to the classification algorithms. Different approaches that were explored include the following:

- **Shared Words:**Review the percentage of words shared by the two questions
- **Levenshtein Distance:**A string metric for measuring the difference between two sequences [89]
- **Term Frequency - Inverse Document Frequency (TF-IDF):**A numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus [10]. Term frequency is essentially quantifying the frequency of a given word or term within a corpus. Inverse Document Frequency is a measure of how much information the word provides (whether common or rare). For this implementation TfidfVectorizer[11]and the Cosine Similarity metric were used to determine the distance between words.

Implementation

After the data was cleaned and features have been generated, the next step is to run the classification algorithm.

As earlier discussed, the mean positive prediction rate in the training set is 36.92%. But what is the mean for positive predictions in the test set? Based on analysis by several Kaggle contributors, it is apparent that the mean positive predictions in the test set is roughly 16.5-17.5%. This was calculated by submitting a test submission to Kaggle using the mean positive prediction rate as the **is_duplicate** score for all **test_id**’s (essentially including a score of 0.369198 for entries). According to another Kaggle Competitor, David Thaler [12], “We can compute [the fraction of positives] directly. Using the log loss formula .. and using the fact that this is a constant prediction, we get:

$$r = \frac{logloss + log(1-p)}{log\left(\frac{1-p}{p}\right)}$$

Based on this, it is possible to compute the mean positive predictions in the test set, which is 17.4%. In order to align the data in the training and test datasets, the training data negative values were oversampled in order to balance the datasets.

Next, the training dataset was split into training and validation sets in order to train the classifiers. Three classifiers were selected for comparison, including the Random Forest Classifier, Logistic Regression, and XGBoost.

It proved difficult to continually benchmark against random forests, as this could only be done using the training data alone (labels were not publicly provided for the test set). This meant that submissions had to be made to Kaggle - with a limit of 5 per day - in order to view testing results. After parameter tuning, Random Forests actually produced the best log loss score against the training set (0.3155), yet when scored against the test set it actually scored much lower (0.5128). [12, 13, 14]

Refinement

Without any tuning, the initial score received using the random forests benchmark model was a log loss score of 0.75660. In order to improve this initial benchmark score, alternative features were explored and selected by making adjustments to the data preprocessing steps. Parameters were also tuned on all three algorithms.

The models were trained by selecting certain features and not others (including both features generated from the cleaned text and original text).

Additionally, parameters were tuned for all three algorithms: GridSearchCV was used for both Random Forest and Logistic Regression, and manual parameter tuning was performed for XGBoost. After tuning the algorithms and exploring approaches to training, it appeared that removing the punctuation and stop words were not beneficial given the feature generation methods used in this approach.

For each of the algorithms, the following parameters were tuned:

- **Random Forests:**
  - **n_estimators:**The number of trees in the forest.
  - **random_state:** The seed used by the random number generator.
- **Logistic Regression:**
  - **solver:** Algorithm to use in the optimization problem.
  - **C:**Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

    - **random_state:** The seed used by the random number generator.
- **XGBoost** [20]**:**
    - **eta:**Analogous to learning rate in Gradient Boosting Machines (GBM). Makes the model more robust by shrinking the weights on each step.
    - **max_depth:**The maximum depth of a tree, same as GBM. Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
    - **min_child_weight:** Defines the minimum sum of weights of all observations required in a child. This refers to min “sum of weights” of observations while GBM has min “number of observations”.
    - **gamma:** A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split. Makes the algorithm conservative. The values can vary depending on the loss function and should be tuned.

After reviewing features to be selected for inclusion in the prediction model, and tuning parameters for the different classifiers, this score was able to be improved to 0.42292 using Logistic Regression, and finally to the model’s best score of 0.39062 using XGBoost.

IV. Results

Model Evaluation and Validation

The model that achieved the best prediction rate, and which was ultimately selected was XGBoost. It consistently performed the best when making predictions against the test dataset.

This final model was derived by generating new features from the original and cldeaned data, testing various combinations of features, and through tuning parameters. This score would also not have been possible without balancing the training dataset through oversampling.

The final model does lose a small amount of log loss score against unseen data (in this case, the test dataset is in fact ‘unseen’, since the labels are unknown). The scores obtained are:

- Training Dataset Log Loss: 0.33179
- Validation Dataset (derived from the original Training dataset) Log Loss: 0.37285

- Test Dataset Log Loss: 0.39062

It does seem understandable that there would be a slight drop in log loss score (as seen in the statistics above), given that the scores are being calculated on new data.

The model is believed to be reasonable and does align with personal expectations, as the evaluation metric score has been considerably improved through multiple iterations of data preprocessing, feature exploration and generation, and model tuning. Additionally, a more subjective metric is comparing results of this model against that of the Kaggle Competition Leaderboard, of which as of this writing, the model sits at the 29th percentile out of roughly 3,000 submissions.

The model can also be trusted to identify duplicates, as the log loss score metric that is provided is based on the probability of each question pair being deemed as duplicate or not. If Quora has a low tolerance for error when identifying question pairs as duplicate, only those duplicate questions with a high probability score (i.e., is_duplicate scores of 1.0 to 0.9) can be leveraged, and others can be manually reviewed. On the other end of the spectrum, if Quora has a high tolerance for identifying duplicate questions, they can be more lenient and take a larger portion of the ‘higher probability’ (i.e., is_duplicate scores of 1.0 to 0.6) questions and consider them as duplicates.

Justification

The Random Forest classifier was used as a benchmark model, and after tuning parameters a final best log loss score on the test set was obtained of 0.51279. In comparison to this model, after parameter tuning the XGBoost score ended up at 0.39062, which is a considerable improvement.

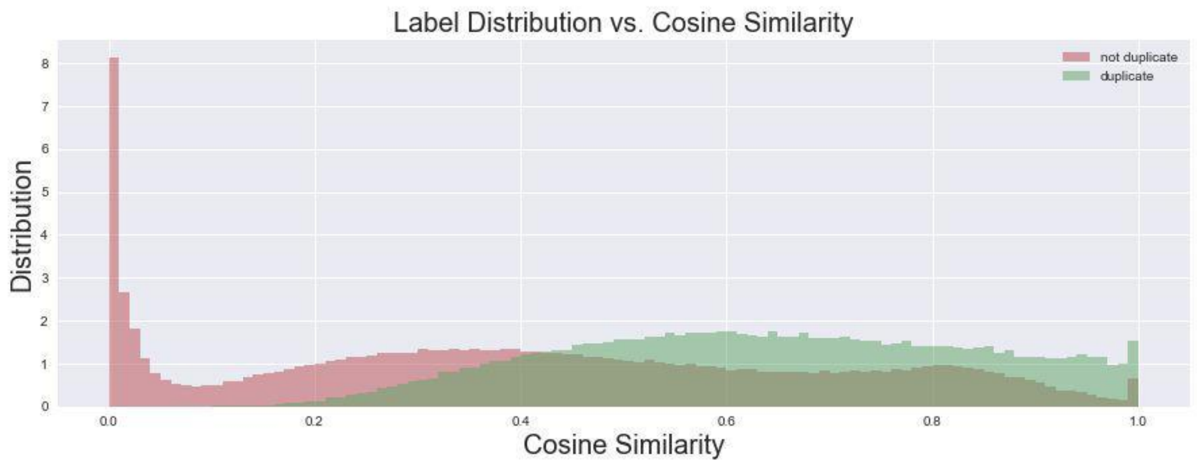
The model does a good job at predicting the probability of whether a given pair of questions is duplicate or not, and to that end it does solve the problem, to a degree. The best ‘real world’ answer to whether this model is robust enough to solve the problem would be whether it 1) is an improvement to the ***current model***used in Production by Quora today, or 2) provides additional insight into an approach that could assist Quora to make improvements to their current model.

V. Conclusion

Free-Form Visualization

A key factor in building a successful prediction model when working with text based documents is the ability to generate new features to extract patterns in the dataset. The field that focuses on this domain is NLP. TfidifVectorizer was used to extract TF-IDF scores based on the words contained in the text of each question, and then the Cosine Similarity was calculated by comparing the TF-IDF scores for the words contained in each pair of questions.

The histogram below depicts the distribution of ‘Duplicate’ and ‘Not Duplicate’ questions compared to the Cosine Similarity score for each question pair. This illustrates the predictive power of the TfidfVectorizer/Cosine Similarity feature that was generated using this NLP approach.



Reflection

In summary, in order to create a prediction model to answer the question: “Quora Question Pairs: Can you identify question pairs that have the same intent?”, a number of steps were performed:

1. Imported the Training and Test datasets to python
2. Performed Data Exploration to understand characteristics and nature of the dataset
3. Preprocessed Dataset Questions to remove stop words, punctuation, and perform stemming
4. Checkpoint: Saved the Preprocessed Data to disk for faster retrieval
5. Generated new features using the text from the question pairs; this included calculating the share of matched words, Levenshtein ratio, and cosine similarity using TfidfVectorizer

6. Rebalanced the training dataset to have an equal set of positive labels in both training and test datasets
7. Executed benchmark classification predictions using Random Forests, while optimizing using GridSearchCV
8. Executed classification tests using Logistic Regression and XGBoost, while optimizing both manually and with GridSearchCV
9. Optimized for the Evaluation Metric: Log Loss Score
10. Saved results and submitted to Kaggle to retrieve Log Loss Scores on the test dataset
11. Iteratively improved the model by tuning classifiers, generating and testing new features, adjusting preprocessing techniques, and identifying and addressing overall issues and inconsistencies

The most interesting aspects of this project were exploring natural language processing techniques, exploring other data scientist models and approaches, exploring new classifiers like XGBoost, and building an end-to-end prediction model as a solution for a real-world problem.

The most challenging area of this project was learning different natural language processing techniques, and determining how these approaches can be applied within the context of the Quora Question Pairs problem. This was also the most interesting aspect of the project, and a domain that will be explored in greater depth in the future.

The model does fit my expectations for solving the problem, and I believe that similar techniques could be used for identifying duplicate or similar records consisting of numbers or text. However, for larger unstructured text documents, more advanced natural language processing approaches would be necessary in order to obtain useful results.

Improvement

There are several improvements that could be made on both the algorithms and techniques that were used throughout this project. This was the first time that I used XGBoost, and so a more rigorous or scientific approach could be applied to algorithm tuning. NLP could be used more effectively to generate new features from the data.

If I had more experience with XGBoost and NLP, I would absolutely explore applying these approaches. Additionally, with more experience I would explore using neural networks to solve this problem using LSTM, Keras, and Tensorflow.

I am certain that better solutions exist in comparison to this project's final solution because a number of teams have developed models that have performed better (as seen on the Kaggle Competition Leaderboard).

References:

1) http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
2) https://github.com/dmlc/xgboost
3) http://wiki.fast.ai/index.php/Log_Loss
4) http://www.kdnuggets.com/2015/03/data-science-measurement-problem-accurac y-auroc-f1.html
5) http://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithm s-python/
6) https://www.kaggle.com/c/quora-question-pairs/data
7) https://xgboost.readthedocs.io/en/latest/
8) https://en.wikipedia.org/wiki/Levenshtein_distance
9) http://www.coli.uni-saarland.de/courses/LT1/2011/slides/Python-Levenshtein.htm l
10) https://en.wikipedia.org/wiki/Tf%E2%80%93idf
11) http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer
12) https://www.kaggle.com/davidthaler/how-many-1-s-are-in-the-public-lb
13) https://www.kaggle.com/ashhafez/temporal-pattern-in-train-response-rates
14) https://www.kaggle.com/anokas/data-analysis-xgboost-starter-0-35460-lb
15) https://tartarus.org/martin/PorterStemmer/
16) http://nbviewer.jupyter.org/github/justmarkham/DAT8/blob/master/notebooks/12_logistic_regression.ipynb
17) http://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

18)https://en.wikipedia.org/wiki/Decision_tree_learning

19)https://en.wikipedia.org/wiki/Gradient_boosting

20)https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

21)

22)