Machine Learning Capstone Report

Quora Insincere Questions Classification Project

Fabian Reyes

December 2018

I. Definition

Project Overview

The digital transformation has completely changed our way of communicating between ourselves. Today most of our social relationships, work, and reading is done online. Although, online communication has been a great place for personal expression and insightful discussions, it has also been a place for toxic and divisive content. The threat of abuse and online harassment, which derives from this toxic content, has been a growing concern this past few years [1]. People who suffer from this divisive content usually stop expressing themselves and give up on seeking different opinions. On the long run, this can become an existential problem in freedom of speech.

Many well-known social and news media platforms have tackled this issue with online moderators, however, the immense amount of content that every day is generated makes this task extremely hard and most of the time futile. Hence, many platforms have turn to machine learning to identify, evaluate and support online moderators on this critical task. This has become a reality for big media platforms such as The Economist [2], Wikipedia [3], New York Times [4], and The Guardian [5]. All of them have partnered with Google and Jigsaw to experiment with their new Perspective API that helps them host better online conversations [6].

This task is part of a subfield called Natural Language Processing (NLP), which is concerned with the interactions between computers and human languages, in particular how to program computers to process and analyze large amounts of natural language data [7]. This field has greatly grown these past few years and I hope that with this proyect I get introduce myself to this new subject, learn and in the long run contribute.

Problem Statement

Quora is a platform that empowers people to learn from each other, thus it's been a frequent target for abusive content. Quora wants to keep their platform as a place where users can feel safe sharing their knowledge with the world, therefore it wants to face this problem by hosting a Kaggle competition aimed at **identifying** and **flagging** insincere questions [8]. By definition an insincere question is a question which is founded upon false premises, or that intend to make a statement rather than look for helpful answers. Quora wants kagglers to develop **scalable methods and models** that will help achieve this goal.

Metrics

As we will se in the exploratory section of this report, our data set is very unbalanced. Therefore, and as recommended in the kaggle competition, I will use de F1-score for model performance. This metrics considers both precision and recall, thus is very helpful in this case because we are dealing with an unbalanced dataset. The formula used to obtain this metric is:

$$\frac{2 * precision * recall}{precision + recall}$$

In [2]:
```
# Loading Data
tqdm.pandas()
train_set = pd.read_csv('Data/train.csv', encoding = 'latin1')
print('Sample of sincere questions')
train_set[train_set.target == 0].sample(3, random_state = 3)
```

```
Sample of sincere questions
```

Out[2]:

	qid	question_text	target
609611	775fb805432cf184d6f9	What do you think of Jallikattu?	0
251920	314cfd7b0a403233d4c0	How can a civil engineer earn well in structur...	0
750481	930783154f445fac3fb9	How many licks would it take to dissolve a nic...	0

In [3]:
```
print('Sample of insincere questions')
train_set[train_set.target == 1].sample(3, random_state = 20)
```

```
Sample of insincere questions
```

Out[3]:

	qid	question_text	target
736959	90548d5140d3a0b4d36e	Do most of the Chinese regularly eat cockroach...	1
307200	3c2badc823240a49f725	Hitler, Pol Pot, and Mao Zedong forced away th...	1
949791	ba1c7c2b0d952944f44f	Do Indians hate Chinese?	1

Figure 1: Sample of sincere (0) and insincere questions (1)

For univariate feature evaluation I will look into **chi-squared statistic** between each non-negative feature and class. Chi-square test measures dependence between stochastic variables, so using this function it filters out features that are the most likely to be independent of class and therefore irrelevant for classification [9]. In case negative features emerge (metafeature extraction por example), I will use a information gain feature evaluation like **Mutual information** implementation of scikit learn. Mutual information (MI) between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency [10].

II. Analysis

Exploratory and Visual Data Analysis

In this section I will go in deep in understanding the data, showing basic statistics and visualizations. First, lets see a sample of the data:

The dataset considered for this project can be found and downloaded from the competition site [11]. **The data set has a total of 1048575 rows and 3 columns**. The description of each column is as folliwing:

- qid: unique question identifier
- question_text: Quora question text
- target: a question labeled “insincere” has a value of 1, otherwise 0

The criteria’s to define an insincere question was based on the following characteristics:

- Has a non-neutral tone: exaggerated tone or rhetorical tone meant to make a statement about a group of people.
- Is disparaging or inflammatory: making discriminatory and/or harsh comments, or based on outlandish premise about a group of people.
- Isn’t grounded in reality: based of false information or absurd assumptions.
- Uses sexual content for shock value, and not to seek genuine answers

As expected from this type of labeling, the ground-truth are not guaranteed to be perfect but suficient for the problem.

```
In [5]: # Target variable count
ax = sns.countplot(y="target", data=train_set, palette="Set1") # strong class imbalance
cnts = train_set.target.value_counts()
print("Strong class imbalance. "
      "There are {} of insincere question that represent"
      "the {}% of the data".format(cnts[1], round(cnts[1]/sum(cnts)*100, 1)))
```

```
Strong class imbalance. There are 64774 of insincere question that representthe 6.2% of the data
```

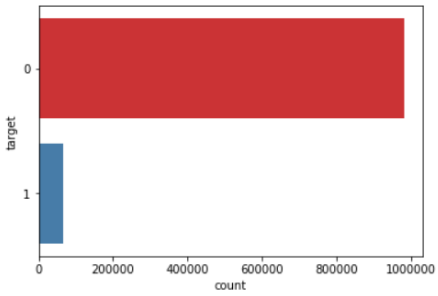


Figure 2: Evidence of class unbalance

As plotted above the data set is very unbalance. Therefore, I will take measures to reduce this problem during modelling (many models include a `class_weight = 'balanced` parameter to assign more weight to the lower class), and splitting data (`stratify` parameter).

Word Level Exploration

A first approach to understand the problem at hand is to visualize the questions and calculate basic frequency statistics of words.

As you can se from the plots and tables above, its very dificult to extract a pattern between target classes. The main reason is because each document is clouded by **noise and common words that we call stopwords**. So one important step will be eliminating this noise words and standarise each word to the same forma

Document level exploration

I will now create 3 basic meta document features to find macro structures and **outliers**. These will be the `char_count_full`, `word_count_full`, and `word_density_full`. These features can give us an idea of the overall structure of the questions.

On average a question consist of **12 words, around 70 character**. The amount of characters per word (word density) is around 5. However, its ver easy to see that there are extremes. On the lower bound, we have questions that have 1 word, or 1 character or in the upper bound questions of 134 words or a word density of 43 characters per word, which is extremely rare. They are non-sense question, for example:

Upper bound extremes are easier to see in a histogram:

The histogram is righ skewed showing that there are big outliers in terms of document lenght. Above 145 characters, documents are considered outliers of this data. Lets look at the same statistic by target:

It seems that insincere questions tend to be more **elaborate than sincere**. On average there are **17 words** per insincere questions instead of **13 words** per sincere question. Now lets see if there are different distribution of the target in the outlier questions:

After testing multiple cut-off point for outliers, the `word_count` < 3, `char_count` > 200, and `word_density` >10 maximizes the insincere class. This criteria will be use to create an outlier feature and help the model with

```
In [13]: h.comparison_plot(wf_sincere[:20],wf_insincere[:20],'word','wordcount', .35)
```

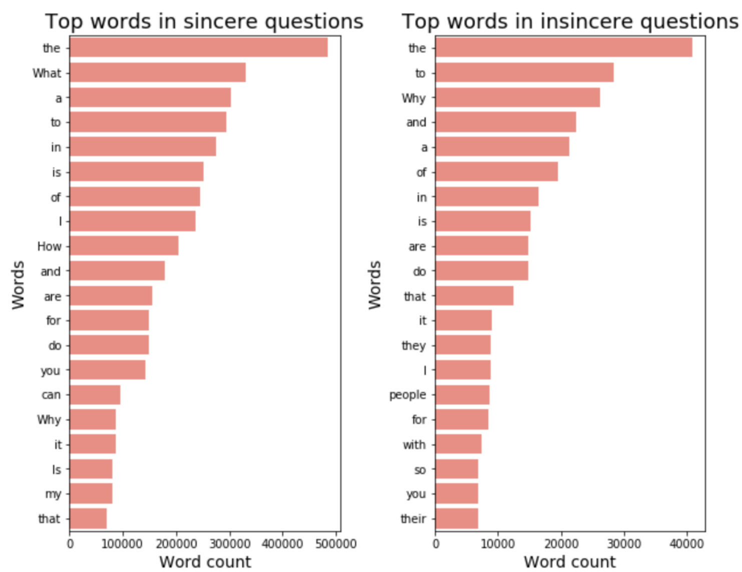


Figure 3: Top unigrams by target

```
In [15]: col = ['char_count_full', 'word_count_full', 'word_density_full']
         pd.DataFrame(train_set.word_count_full.value_counts(sort=False).head(10))

         # Summary statistics
         train_set[col].describe().applymap(np.int64)
```

Out[15]:

	char_count_full	word_count_full	word_density_full
count	1048575	1048575	1048575
mean	70	12	5
std	38	7	0
min	1	1	0
25%	45	8	4
50%	60	11	5
75%	85	15	5
max	1017	134	43

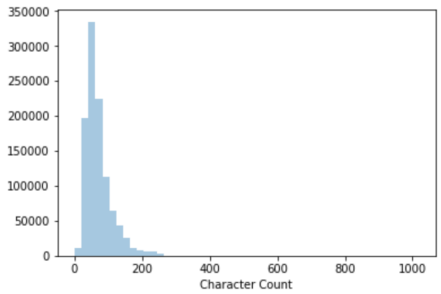
Figure 4: Summary statistics of 3 document level features

```
In [16]: train_set[train_set.word_density_full >= 43].question_text

Out[16]: 455194    ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ...
         Name: question_text, dtype: object

In [17]: train_set.question_text[455194].split()

Out[17]: ['ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZzzzzzz
         zzzzzzzZZZZZZZZZZZZZZZZZZZZZZZZzzzzzzzzzzzzzzzzzzzzzzzzzzz',
          'hi',
          'earth?']
```

Figure 5: Outlier question example

In [18]:
```
sns.distplot(train_set.char_count_full, kde=False, axlabel = 'Character Count')
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a251bd7b8>
```



In [19]:
```
iqr = np.subtract(*np.percentile(train_set.char_count_full, [75, 25])) #interquartile range
upper_bound_outlier = np.percentile(train_set.char_count_full, 75) + 1.5*iqr
print('Above {} characters documents are considered outliers'.format(upper_bound_outlier))
```

```
Above 145.0 characters documents are considered outliers
```

In [20]:
```
iqr = np.subtract(*np.percentile(train_set.word_density_full, [75, 25])) #interquartile range
upper_bound_outlier = np.percentile(train_set.word_density_full, 75) + 1.5*iqr
print('Above {} word density documents are considered outliers'.format(upper_bound_outlier))
```

```
Above 6.907969639468693 word density documents are considered outliers
```

Figure 6: Histogram of character counts

In [21]:
```
# Target comparison
train_set[['word_count_full', 'target']].groupby('target').aggregate([min, np.mean, np.median, max])
```

Out[21]:

	word_count_full			
	min	mean	median	max
target				
0	1	12.510228	11	134
1	1	17.238645	15	64

Figure 7: Histogram of character counts

In [22]:
```
criteria = (train_set.word_count_full < 3) | (train_set.char_count_full > 200) | (train_set.word_density_full > 10)
outliers = train_set[criteria]
proportion = 4458/(4458+12337)*100
print('Beign an outlier makes the insincere class proportion significantly higher ({}%)'.format(round(proportion)))
outliers['target'].value_counts()
```

```
Beign an outlier makes the insincere class proportion significantly higher (27%)
```

Out[22]:
```
0    12337
1     4458
Name: target, dtype: int64
```

Figure 8: Histogram of character counts

```
In [45]: # baseline score: Logistic Regression
         lrn_basic = LogisticRegression(class_weight = 'balanced',
                                        C = 1e10, # Large C for no regularization
                                        random_state = 33,
                                        penalty = 'l1')
         pipe_lrn = create_pipeline(lrn_basic)
         results_mod = cv_evaluation(pipe_lrn, train_x.question_text, train_y)
         print('The mean train {} and test CV {}.'.format(round(results_mod['train_score'], 2),round(results_mod['test_score'],

         The mean train 0.71 and test CV 0.47.
```

Figure 9: Benchmarck LogisticRegression

these types of questions. This will specially useful when looking into these documents after processing, its most likely that many will end up in zero words because of stopwords removal.

Algorithms and Techniques

The main algorithms are separated into two main groups:

Feature extraction algorithms

```
- **Bag of Words Model**: all text feature extraction will be done using this model. is represented as
- **TF-IDF statistic**: TF-IDF score represents the relative importance of a term in the document and t
```

Classification algorithms and techniques

This is a binary classification task. Hence, the models that will be tested with pros and cons are:

```
- **Linear Models**: The main algorithm will be the logistic classifier implemented as `LogisticRegressi
- **Naive Bayes**: Is a family of simple "probabilistic classifiers" based on applying Bayes' theorem wi
- **LightGBM**: Is one of the most recent and most powerful gradient boosting machines. Microsoft develo
its way to top kaggle competitions. Gradient boosting in general suffers from eficiency and scalabity wh
- ** Ensemble**: I will also test an ensemble scheme. The goal of ensemble methods is to combine the pre
```

Since our evaluation metric will be F1-score, our prediction must be discrete, 0 or 1. It becomes necessary to search an adequate **cut-off probability threshold** to decide to which of the classes the prediction belongs and to maximize F1-Score.

Benchmark

I will construct two benchmark model:

- Naive model: This benchmark is my naive reference, it has no intelligence behind it. I will predict for all test cases the predominant label. All my expected results must be always above this score. As we saw in the exploratory section above, the mayority class is 0 (sincere question), thus if we predict that all the questions are sincere we result in a **F1-Score of 0.06**. This is my lower bound limit, if I have a result very similar or even lower it means that something is wrong with my approach.
- Basic linear model: This second benchmark is my simple yet intelligent model, which I will try to beat with more complex models models. This logistic classifier will be trained on the crude words of the questions, without preprocessing. This will allow me to evaluate also the impact of my preprocessing step.

```
In [45]: # baseline score: Logistic Regression
lrn_basic = LogisticRegression(class_weight = 'balanced',
                               C = 1e10, # Large C for no regularization
                               random_state = 33,
                               penalty = 'l1')
pipe_lrn = create_pipeline(lrn_basic)
results_mod = cv_evaluation(pipe_lrn, train_x.question_text, train_y)
print('The mean train {} and test CV {}.'.format(round(results_mod['train_score'], 2),round(results_mod['test_score'],

The mean train 0.71 and test CV 0.47.
```

Figure 10: Learning curve of LogisticRegression benchmarck

```
In [15]: col = ['char_count_full', 'word_count_full', 'word_density_full']
pd.DataFrame(train_set.word_count_full.value_counts(sort=False).head(10))

# Summary statistics
train_set[col].describe().applymap(np.int64)
```

Out[15]:

	char_count_full	word_count_full	word_density_full
count	1048575	1048575	1048575
mean	70	12	5
std	38	7	0
min	1	1	0
25%	45	8	4
50%	60	11	5
75%	85	15	5
max	1017	134	43

Figure 11: Classification Strategy

I used high `C`parameter in `LogisticRegression` because natively this algorithm implements regularization. Regularization is a method that constrains or shrinks the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting. A high `C` parameter discourages regularization.

The CV score for train and test are significantly different from each other, which implies strong overfitting. This is more evident when we observe the learning curve:

There is also a leaderboard that orders each of the competition participants by their public test set performance [15]. This will also be consulted for comparison.

III. Methodology

In the figure below I detail the strategy that I used to tackle this classification problem. The idea behind the strategy is to feed the last prediction with a multitude of information. On one hand, you have the text features obtain through the bag of words model. This generates a vast document-term matrix, which can be directly use for classification. Nevertheless, its extremely sparse and models have a hard time finding patterns. After applying TFIDF statistic, there is a need to reduce the dimensionality of this matrix for performance and time efficiency. Two ways of achieving this, and at the same time improving the data, is by feature selection, and by using a classifier's prediction as a feature for another model. This is very similar to stacking. The idea behind it is to provide the next model with an already process view of the data. On the meta feature side, I will extract document and sentiment level features as describe before. I hope that with this strategy the final model will be able to grasp more patterns when combining the process text features with the meta features of the questions.

Data Preprocessing

The data was intensively processed in our task. I divided this phase into two parts:

1. **Text cleaning process**: The main objective of this part is to apply a series of transformations to the text data know as text preprocessing techniques. These techniques have been shown to improve, standarise and facilitate modelling. The techniques are de following:
    - Lower casing: Standarise text data. Python, for example, finds that `Hello` and `hello` are to different strings.
    - Punctuation removal: Standarise text data. Punctuation marks is a human readible code that helps us give rhtyms and intonation to our reading. For our task and strategy, most of it is noise.
    - Special character removal: Standarise text data. Mainly accents and non latin characters.
    - Lemmatization: Standarise text data. A word may be express in multiple tense depending on the context. So lemmatization is the process of grouping together the inflected forms of a word so they can be analysed as a single item.
    - Stopwords removal. Noise reduction. Many words serve as connectors in a sentence. They are the most common and frequent words in any language, thus they are usually not informative and are filtered out the analysis.

Most of them are standard transformations and have become canonical in most text classificaction tasks. This proyect implemented a function called `clean_text` that applies all the transformations above and generated two columns: `qt_clean` a clean standarise text vector with stopwords and `qt_clean_stop` a clean standarise text vector without stopwords.

2. **Feature Extraction**: This is the most intensive part of this project. The main objective is to extract all the posible and informative features from the text for later modelling. A poor feature pool yields a poor classification model, thus my strategy will be:
    - Meta document features: These are all features related to the document statistics. They by itself are not strong predictors but in combination they can help the model gain more insight of the text. These are number of characters, number of words, word density, number of stopwords, number of punctuation marks, number of upper case words, number of nouns, verb, adjectives, pronouns, and adverbs.
    - Sentiment Features: Words in all language have an inherent sentiment behind it. Most of the time it depends on the context of what it is said. However, words like `Happiness` almost all of the time is positive. Many libraries exist today that are able to score a group of words and approximate a sentiment. So I will make use of the powerfull `TextBlob` package to extract polarity, subjectivity and positivity of the questions.
    - Text features: I will decompose each text into a document-term matrix, describes the frequency of terms that occur in a collection of documents. The matrix consist of (m x n) dimensions beign **m** the number of documents or questions and **n** the terms which are present in the documents. Since each documents consist of a small portion of the whole vocabulary of the set, most of the matrix is extremely sparse and noisy. This is even worse when considering not only words, but bigrams or trigrams of words. To reduce noise, I will normalize each term using Term Frequency (TF) Inverse Document Frequency (IDF) statistic which generally improves modelling performance. Moreover, I will filter-out high sparse terms and perform feature selection using chi-squared statistic.
    - Predictive Text features: I will find the model that performs best with only the text features. This prediction will become a feature that will be stack with the metafeatures (meta document features + Sentiment Features).
3. **Modelling**. To extract the most out of the features I will emsemble different models on different set of features and the predictions of this group of models will be pooled to a master classifier that will produce the classification task. I will use a mixture of the following classifiers:
    - Logistic Regression (base learner)
    - Naive Bayes
    - LightGBM (very similar to gradient boosting but faster)
    - Ensemble (from the 3 above)

I will support my modelling with learning curves and cross-validation (5-fold) methodology.

References

[1] Nobata, Chikashi & Tetreault, Joel & Thomas, Achint & Mehdad, Yashar & Chang, Yi. (2016). Abusive Language Detection in Online User Content. 145-153. 10.1145/2872427.2883062.

[2] https://medium.economist.com/help-us-shape-the-future-of-comments-on-economist-com-fa86eeafb0ce. Help us shape the future of comments on economist.com

[3] https://meta.wikimedia.org/wiki/Research:Detox. Research:Detox.

[4] https://www.nytco.com/press/the-times-is-partnering-with-jigsaw-to-expand-comment-capabilities/. The Times is Partnering with Jigsaw to Expand Comment Capabilities

[5] https://www.theguardian.com/technology/series/the-web-we-want. The web we want.

[6] https://www.perspectiveapi.com/#/. Perspective API.

[7] https://en.wikipedia.org/wiki/Natural_language_processing. Natural Language Processing.

[8] https://www.kaggle.com/c/quora-insincere-questions-classification. Quora Insincere Questions Classification.

[9] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html. Scikit Learn Chi-squared statistic.

[10] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html#sklearn.feature_selection.mutual_info_classif. Scikit Learn Mutual Information.

[11] https://www.kaggle.com/c/quora-insincere-questions-classification/data. General Description of Data.

[12] https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-i Comprehensive guide to implement text classification in python.

[13] https://scikit-learn.org/stable/modules/ensemble.html#ensemble. Ensemble methods.

[14] https://lightgbm.readthedocs.io/en/latest/. LightGBM's documentation.

[15] https://www.kaggle.com/c/quora-insincere-questions-classification/leaderboard. Leaderboard.