

# Quora Insincere Questions Classification

**Author:** Jeevan Anand Anne

**Date:** December 14th, 2018

## I. Definition

### Project Overview

This project aims to identify the insincerity in the Quora questions i.e. to classify whether the Quora questions are insincere or not. Quora is a platform that enables people to learn from each other, people can ask different questions and connect with others who contribute unique insightful answers. A key challenge is to recognize and flag insincere questions. This is very important for very good user experience. [2]

This is a case of classification which is part of Supervised Learning. For a given question, "target" is labeled as "insincere", which has a value of 1, otherwise 0 (sincere), while qid and question\_text are other variables.

### Datasets and Inputs

Dataset Source: <https://www.kaggle.com/c/quora-insincere-questions-classification/data> [2]

### Problem Statement

Classify if a question asked on Quora is sincere or not. It is a binary classification problem.

### Metrics

As this is a supervised classification problem, F1 Score is used for evaluation. Here in this problem both precision and recall are important to consider when we have high class imbalance, so F1 score is used to get a tradeoff between precision and recall. It can be mathematically defined as:

$$F1 = (2 * P * R) / (P + R)$$

where,

P = Precision. It's a proportion of actual positives of all predicted positives

R = Recall. It's a measure of proportion of positives of all actual positives

F1 score is a combined metric of precision and recall as mentioned.

## II. Analysis

### Data Exploration & Visualization

We have train and test datasets. Train dataset has 2 features (qid, question\_text) and 1 target variable. Test dataset has 2 features (qid, question\_text).

```
#import data
train_df = pd.read_csv("train.csv")
test_df = pd.read_csv("test.csv")
print("Train shape :", train_df.shape)
print("Test shape :", test_df.shape)
```

```
Train shape : (1306122, 3)
Test shape : (56370, 2)
```

```
#display top 5 rows
train_df.head()
```

	qid	question_text	target
0	00002165364db923c7e6	How did Quebec nationalists see their province...	0
1	000032939017120e6e44	Do you have an adopted dog, how would you enco...	0
2	0000412ca6e4628ce2cf	Why does velocity affect time? Does velocity a...	0
3	000042bf85aa498cd78e	How did Otto von Guericke used the Magdeburg h...	0
4	0000455dfa3e01eae3af	Can I convert montra helicon D to a mountain b...	0

Train dataset has approximately 1.3M observations and test dataset has 56K observations. We have only unstructured text data from which we will have to extract relevant features to feed into the machine learning algorithms. Preprocessing has to be done on the text.

```
#structure of the dataframe
train_df.info()
```

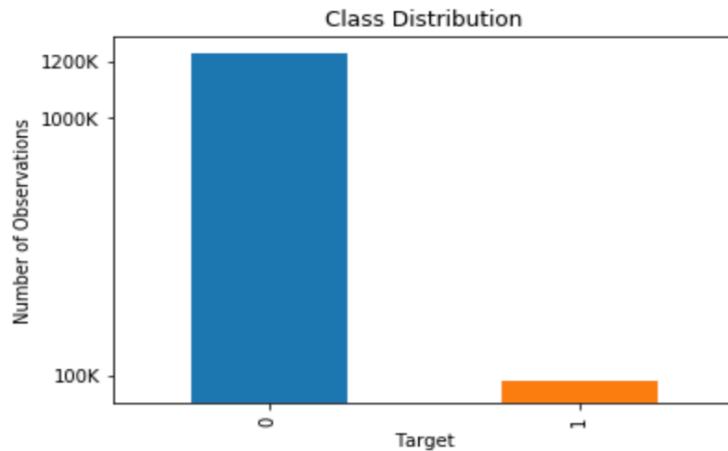
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1306122 entries, 0 to 1306121
Data columns (total 3 columns):
qid          1306122 non-null object
question_text 1306122 non-null object
target       1306122 non-null int64
dtypes: int64(1), object(2)
memory usage: 29.9+ MB
```

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56370 entries, 0 to 56369
Data columns (total 2 columns):
qid          56370 non-null object
question_text 56370 non-null object
dtypes: object(2)
memory usage: 880.9+ KB
```

There are no missing values in train and test datasets

## What is the class (target) distribution?



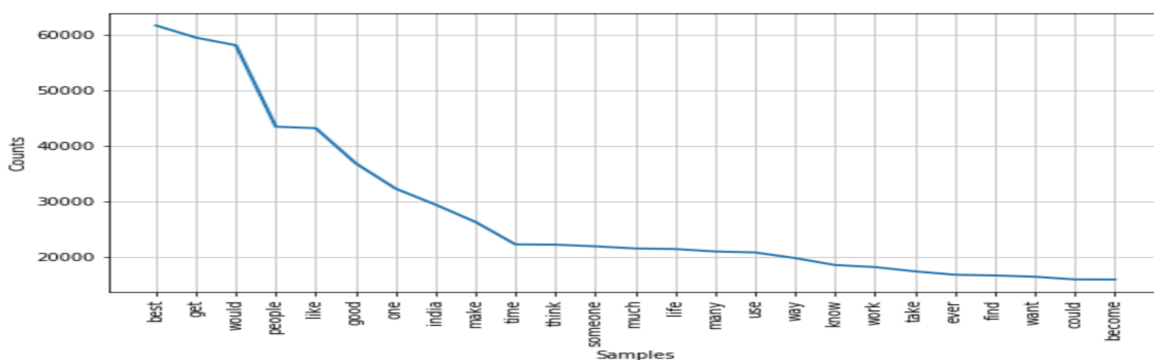
```
train_df['target'].value_counts(normalize=True)
0      0.93813
1      0.06187
Name: target, dtype: float64
```

From the plot, we can say that there are almost 93.8% of Quora questions are sincere (target=0) and 6.2% of Quora questions are insincere (target=1), which is heavily imbalanced.

## What are the most frequent words in both of the classes?

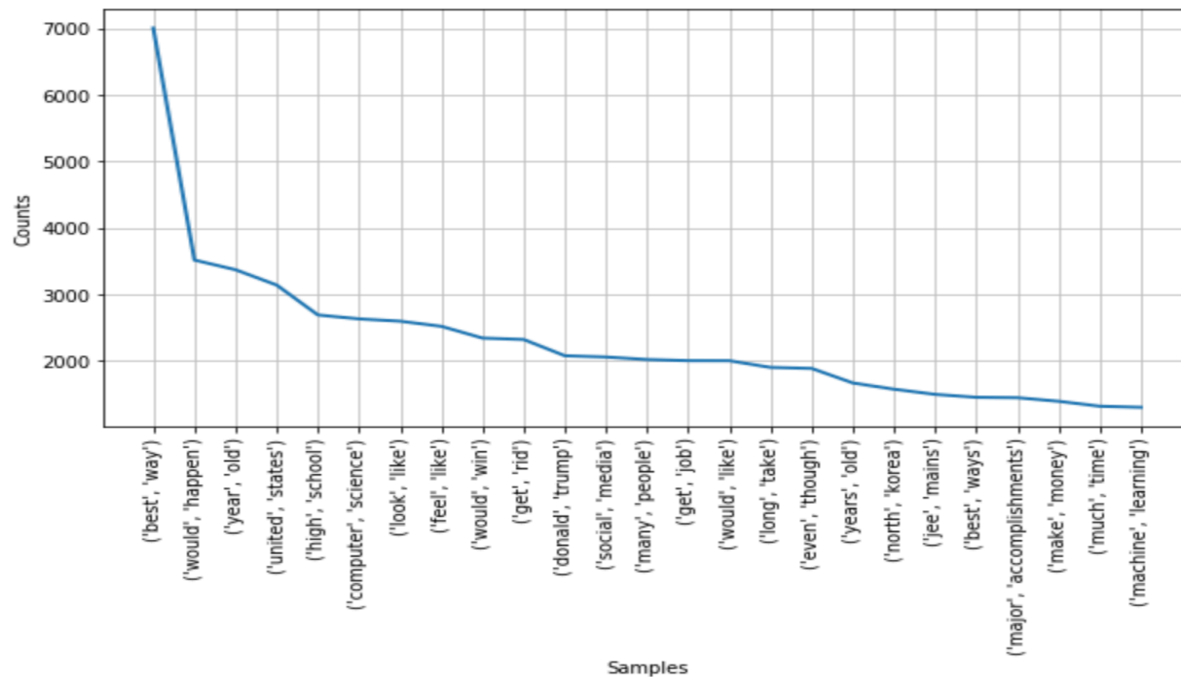
### For Sincere Questions:

#### Unigram:



It makes sense based on the words in the sincere questions we see in the plot.

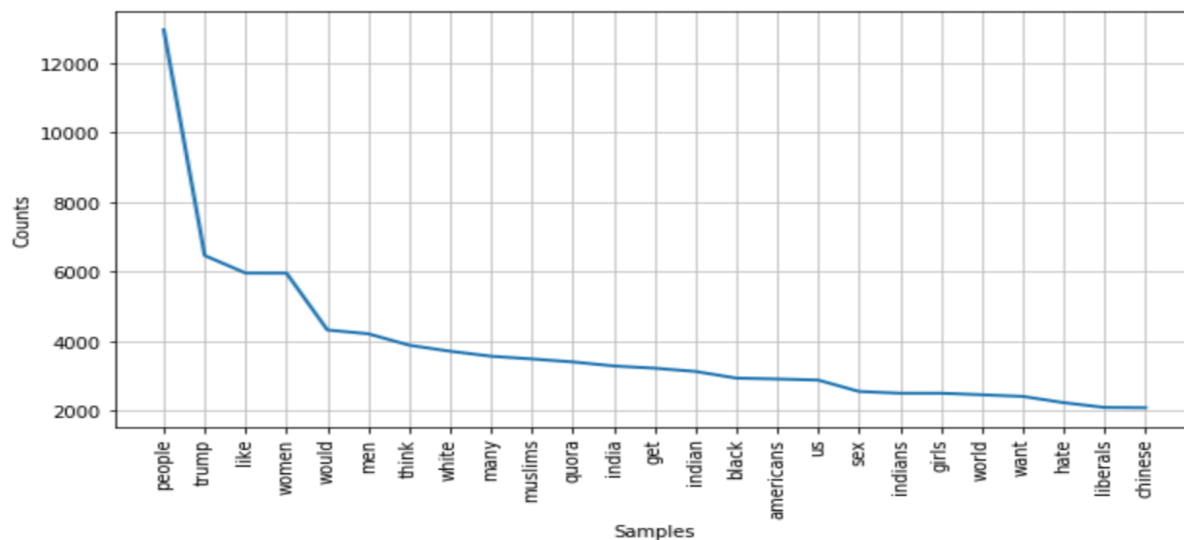
## Bigram:



All the words in the sincere questions we see in the plot are all in positive and they make sense.

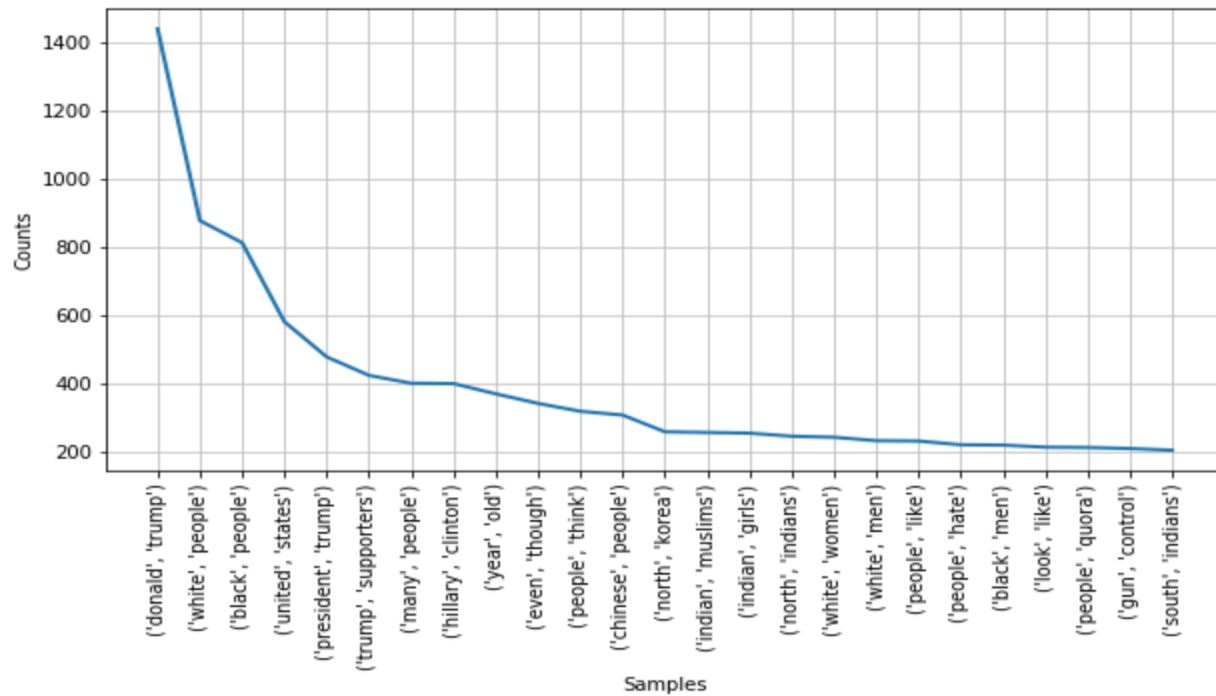
## For Insincere Questions:

### Unigram:



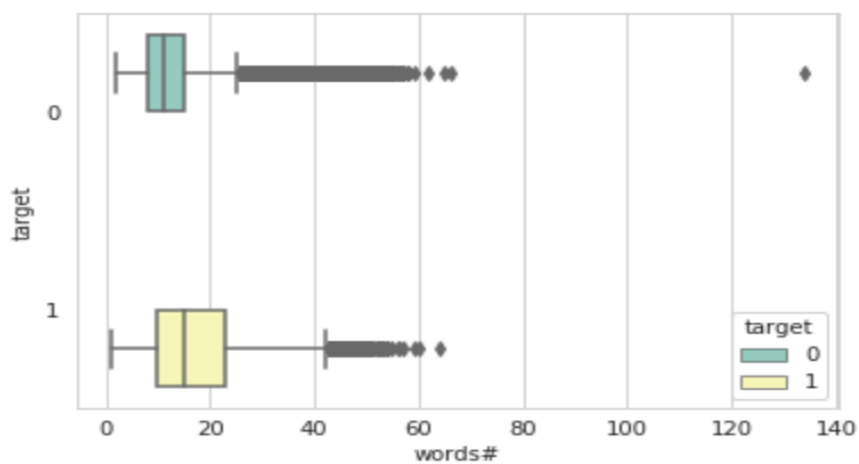
We see most of the frequent words in insincere questions are about politics, race etc.

## Bigram:



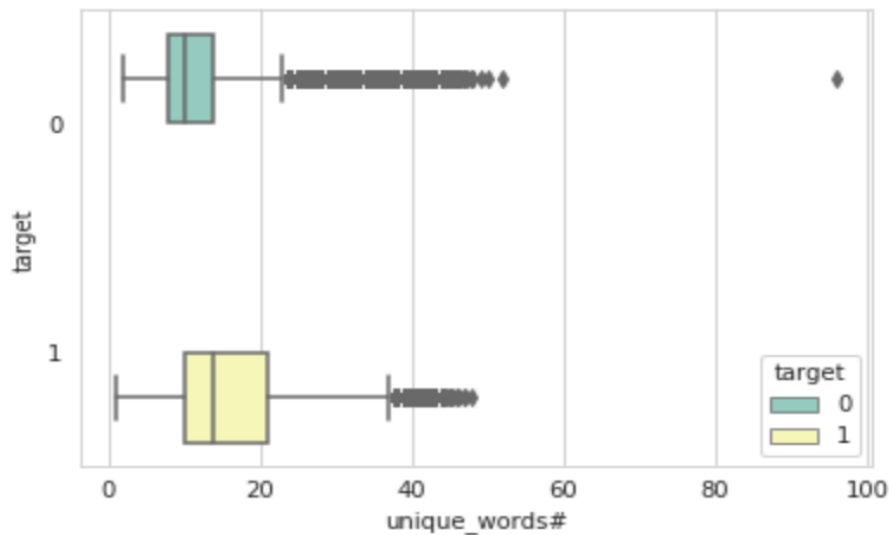
Now it makes more sense about the words used in insincere questions.

**How the number of words in training dataset are distributed?**



There are a greater number of words for insincere questions than in sincere questions when comparing the median values. This might be a good feature to use. There are outliers too.

**How the number of unique words in training dataset are distributed?**



There are a greater number of unique words for insincere questions than in sincere questions when comparing the median values and might be a good feature to use.

## Algorithms and Techniques

There are different types of machine learning models for classification which can be used to train a model. We will try to implement following classifiers for solving the problem :

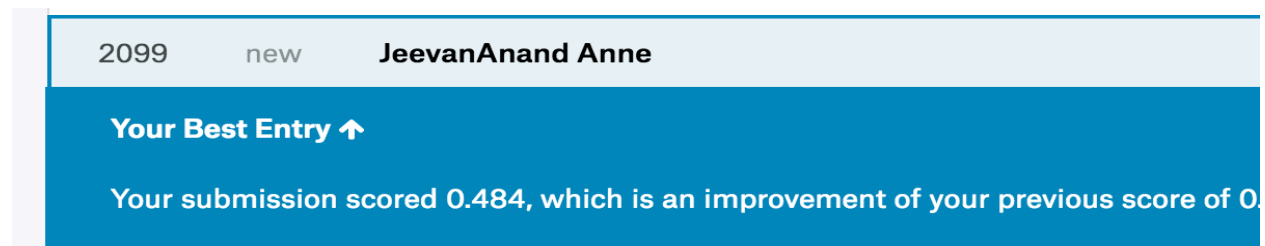
- Logistic Regression
  - Naïve Bayes
  - Bagging (Random Forest)
- a) Implementing a Linear Classifier (Logistic Regression). It measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic/sigmoid function. They are easy to implement, fast and accurate.
- b) Naive Bayes classifiers are linear classifiers which are simple and efficient probabilistic models. This algorithm works well mostly on the high dimensional text data. Despite of the assumption that features are independent, Naive Bayes will still tend to perform well. They are easy to implement, fast and accurate.

- c) A simple Decision Tree will likely overfit. I will be using ensemble methods to overcome this issue.

**Bagging:** Multiple classifiers are built independently on samples of the training data and the output is combined through voting (either mean, median or mode). For example: Random Forest and Extra Trees Classifiers. Random Forest works well on high dimensional data, so it is a good fit for this dataset as it has a large number of features. Extra trees will make random splits on features.

## Benchmark

The benchmark is the F1 score of the logistic regression classification technique used, which obtained a F1 score of  $\sim 0.484$  using sklearn's pipeline which includes TF-IDF vectorization without any cross validation. We will try to beat this leaderboard score on Kaggle and try not to overfit.



A screenshot of a Kaggle leaderboard entry. The header row shows the year '2099', the status 'new', and the user name 'JeevanAnand Anne'. Below this, a blue banner contains the text 'Your Best Entry ↑'. Underneath the banner, a message states: 'Your submission scored 0.484, which is an improvement of your previous score of 0'.

2099	new	JeevanAnand Anne
Your Best Entry ↑		
Your submission scored 0.484, which is an improvement of your previous score of 0		

## Methodology

### Data Preprocessing

#### Stopword Removal

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) which has been programmed to ignore in many of the text related techniques.

We would not want these words (features) taking up space or taking up valuable processing time. We can remove the stop words from the actual text data before vectorizing. NLTK (Natural Language Toolkit) in python has a list of stop words stored.

## N-grams

The idea is to make (statistical) predictions about what is happening in a sentence. Things that happen could be that a particular word shows up next, or that an element belonging to a particular word class shows up next. [5]

Here is our sentence "I am doing capstone project for MLND"

The machine wants to get the meaning of the sentence by separating it into small pieces. How should it do that?

- Unigram can be regarded as words one by one i.e tokens.  
"I", "am", "doing", "capstone", "project", "for", "MLND"
- It can regard words two at a time. This is bigram; each two adjacent words create a bigram.  
"I am", "am doing", "doing capstone", "capstone project", "project for", "for MLND"
- It can regard words three at a time. This is trigram; each three adjacent words create a trigram.  
"I am doing", "am doing capstone", "doing capstone project", "capstone project for", "project for MLND"

## TF-IDF

tf-idf, short for term frequency–inverse document frequency, is a numeric measure that is used to score the importance of a word in a document based on how often it appeared in that document and a given collection of documents. In simple words, if a word appears less often in a document, then it should be important, also relevant, and we should give that word a high score. But if a word appears more often, it's does not give much information, therefore we should assign a lower score to that word. The math formula for this measure: [6]

$$\text{tfidf}(t,d,D)=\text{tf}(t,d)\times\text{idf}(t,D)$$

Where t denotes the terms; d denotes each document; D denotes the collection of documents.

The first part of the formula  $\text{tf}(t,d)$  is simply to calculate the number of times each word appeared in each document.

Let's first write down the complete math formula for IDF.



$$idf(t, D) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}$$

- The numerator:  $|D|$  is inferring to our document space. It can also be seen as  $D = d_1, d_2, \dots, d_n$  where  $n$  is the total number of documents. [6]
- The denominator:  $|\{d \in D : t \in d\}|$  implies the total number of times in which term  $t$  appeared in all of your document  $d$  (the  $d \in D$  restricts the document to be in your current document space). Note that this implies it doesn't matter if a term appeared 1 time or 100 times in a document, it will still be counted as 1, since it simply did appear in the document. As for the plus 1, it is there to avoid zero division. [6]

Creating training and validation datasets from train data, in order to build a model on train data and evaluate on validation dataset.

## Standardization

We will be using the numerical features like Number of words, Number of unique words etc., and hence need to scale, which will then be used by the machine learning algorithms.

## Implementation

### Logistic Regression

#### Approach1:

I am using scikit-learns Logistic Regression classifier without any cross validation, with TF-IDF word representations and with default parameters.

Public Leaderboard score of 0.484 on Kaggle and there will be no private leaderboard until the competition is over.



#### Approach2:



I used sklearn's Logistic Regression classifier without any cross validation, with the same parameters mentioned above and also ngrams (unigram and bigram).

This model was used to predict the test dataset and uploaded to Kaggle. It got the following scores:

Validation Score: 0.5917 on validation dataset

Public Leaderboard Score: 0.577 on text dataset

1971	new	JeevanAnand Anne		0.577	9	5m
<b>Your Best Entry</b> ↑						
Your submission scored 0.577, which is an improvement of your previous score of 0.484. Great job!						
					Tweet this!	

2170	new	JeevanAnand Anne		0.484	8	2d
<b>Your Best Entry</b> ↑						
You advanced 199 places on the leaderboard!						
Your submission scored 0.577, which is an improvement of your previous score of 0.484. Great job!						
					Tweet this!	


## Naïve Bayes:

I used sklearn's Naïve Bayes MultinomialNB classifier without any cross validation, bag of words and with the default parameters.

This model was used to predict the test dataset and uploaded to Kaggle. It got the following scores:

Validation Score: 0.572

Public Leaderboard Score: 0.557

1978	new	JeevanAnand Anne		0.577	13	12m
<b>Your Best Entry</b> ↑						
Your submission scored 0.557, which is not an improvement of your best score. Keep trying!						

## Random Forest:



I used sklearn's Random Forest classifier without any cross validation, with the tf-idf vectorization, removing stop words and ngrams (unigram, bigram). It took more time to execute and not able to submit the predictions in Kaggle as it exceeds more than 120 minutes.

This model was used to predict the test dataset and uploaded to Kaggle. It got the following scores:



Validation Score: 0.499

## Refinement

Using the n-grams (Unigram, Bigram) features along with tf-idf vectorization creation, stop word removal, new features (word length, unique word length) and Logistic Regression Classifier, it improved the score from 0.484 (baseline) to 0.593, approximately 11% improvement compared to our baseline Logistic Regression classifier

2028	new	JeevanAnand Anne		0.577	14	2d
<b>Your Best Entry ↑</b>						
You advanced 63 places on the leaderboard!						
Your submission scored 0.593, which is an improvement of your previous score of 0.577. Great job!						
 Tweet this!						

1968	new	JeevanAnand Anne		0.593	15	2h
<b>Your Best Entry ↑</b>						
Your submission scored 0.593, which is an improvement of your previous score of 0.577. Great job!						
 Tweet this!						

## Results

### Model Evaluation and Validation

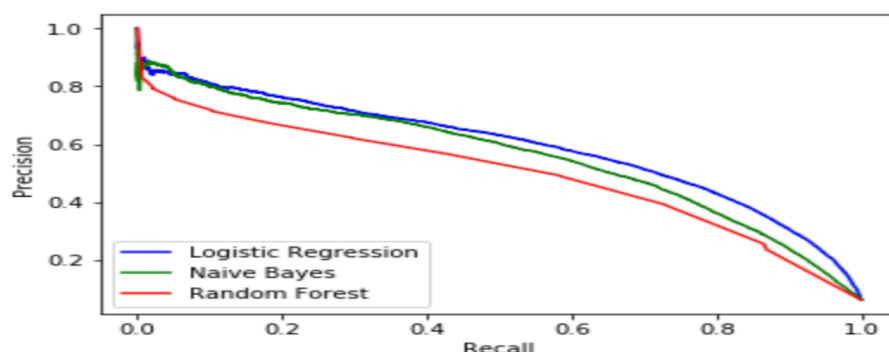
Precision and Recall curves

a) Comparison of precision recall curves for the below algorithms:

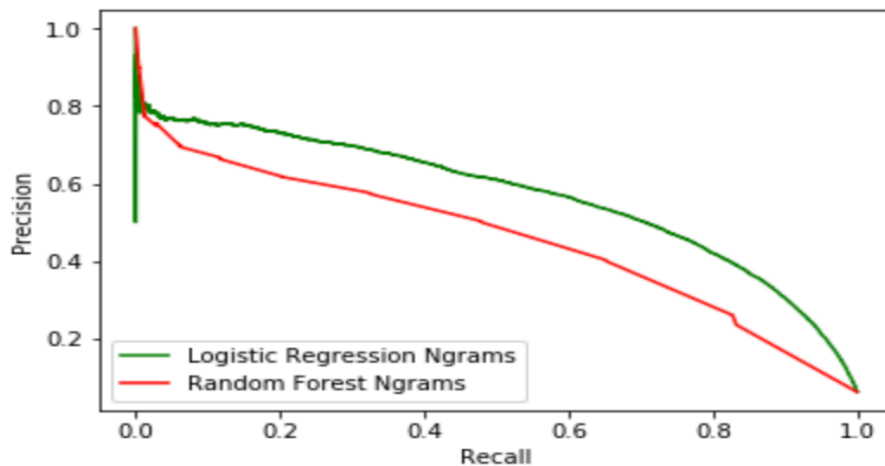
Logistic Regression with tf-idf vectorization features and stop word removal

Naïve Bayes with bag of words features

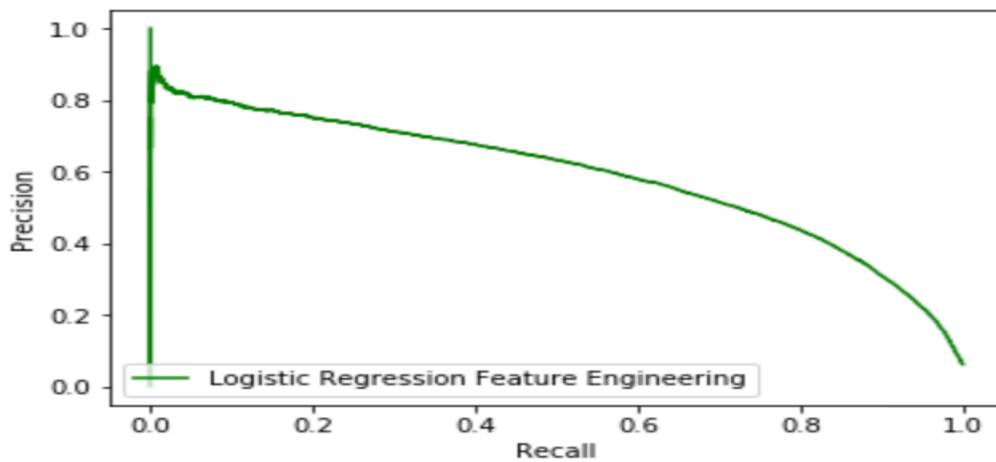
Random Forest with tf-idf vectorization features and stop word removal



b) Logistic Regression and Random Forest with ngrams (unigram, bigram)



c) Below is the precision recall curve for the Logistic Regression using new features (Number of words, Number of Unique Words) along with tf-idf vectorization and stop word removal. Here we could see an improvement.



The curve closer to 1 of precision and recall will be the best model. Precision and recall curves are better to look at than ROC curves when we have imbalance datasets.

## Justification

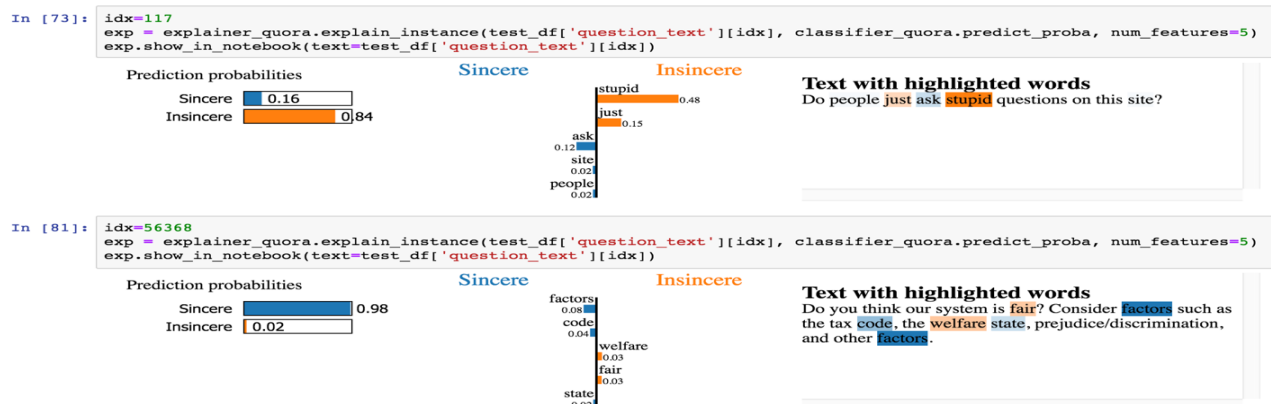
Difference in the benchmark and my solution:

Models	Preprocessing	Val - F1 Score	Test - F1 Score - Leaderboard
Logistic Regression - Benchmark	TF-IDF, Stop Words Removal	0.5949	0.484
Logistic Regression	TF-IDF, Stop Words Removal, Ngrams (unigram, bigram)	0.588	0.577
Logistic Regression	TF-IDF, Stop Words Removal, Ngrams (unigram, bigram), Number of words and Number of Unique word features	0.5964	0.593
Naïve Bayes	Bag of words	0.5695	0.557
Random Forest	TF-IDF, Stop Words Removal	0.535	Not able to submit, as the kernel execution exceeded 120 minutes
Random Forest	TF-IDF, Stop Words Removal, Ngrams (unigram, bigram)	0.497	Not able to submit, as the kernel execution exceeded 120 minutes

Given that the benchmark I used and created a better model than benchmark, the improvement of about 11% is significant enough to consider my solution a satisfactory one. Sometimes simple model gives better results, as in the above situation.

## Conclusion

### Free-Form Visualization



With our model built, let's interpret the reasons of the predicted class using LIME (Local Interpretable Model-Agnostic Explanations) [7].

Our model:

- Correctly predicted “Insincere” for the text “Do people just ask stupid questions on this site?” with high probability of the word stupid in the text.
- Correctly predicted “Sincere” for the text “Do you think our system is fair? Consider factors such as the tax code, the welfare state, prejudice/discrimination, and other factors.”

Both the results make sense based on the words appear in the text

## **Reflection**

My whole project experience as follows:

- Wanted to explore more NLP preprocessing techniques, and advanced techniques like word2vec and thought this Kaggle competition will help me look at mentioned techniques
- Found the dataset on the Kaggle competitions
- Visualized the data, did preprocessing by learning from other contestants in kernels from Kaggle.
- Preprocessing was an integral part of this project, which helped to get better results.
- Applying selected algorithms and tuning the best performing model.
- Comparing my tuned model against the benchmark result.
- Interesting part was learning new preprocessing techniques and visualization techniques for creating features.
- Difficult was figuring out to run the bagging models as it is taking so much of time.

## **Improvement**

1. Create more better features – feature engineering.
2. Dimensionality reduction of the features
3. Preprocessing like contractions, correcting the words which are mistyped etc.
4. Use advanced bagging and boosting models while having better resources GPU for example
5. Implement word2vec with the pretrained embeddings available and improve the coverage of the vocabulary with necessary preprocessing at each stage

6. Implement sequence models like LSTM, GRUs which might help us in improving our score.

## References

- 1) <https://www.kaggle.com/c/quora-insincere-questions-classification/kernels>
- 2) Quora Inc, <https://www.kaggle.com/c/quora-insincere-questions-classification/data>
- 3) <https://www.cnblogs.com>
- 4) <https://www.geeksforgeeks.org> - NLTK
- 5) <https://en.wikipedia.org/wiki/N-gram>
- 6) <https://en.wikipedia.org/wiki/Tf%E2%80%93idf>
- 7) <https://homes.cs.washington.edu/~marcotcr/blog/lime/>