**Shalin Gosalia**

# Final Report:

# Quora Insincere Data Classification

## **Problem Statement**

Quora is a free form forum which invites contributions from everyone on topics ranging from current affairs to latest technology trends to personal view points in the realm of sports, politics etc. The content is not moderated before it can be published and hence, is available for general view to the public.

An existential problem for any major website today is how to handle toxic and divisive content. Quora wants to tackle this problem head-on to keep their platform a place where users can feel safe sharing their knowledge with the world.

An effective data oriented solution needs to be devised by developing models that identify and flag insincere questions that keeps a check on misinformation that can spread at a viral peace if not kept in check.

This is a chance to combat online trolls at scale in order to help Quora uphold their policy of “Be Nice, Be Respectful” and continue to be a place for sharing and growing the world’s knowledge.

Scope of the solution would be to perform data classification over the dataset available for the scope of this problem, which includes questions pulled from Quora postings in the range of ~432K Quora postings (split into training and test datasets) and weed out insincere questions.

## **Data**

#### [Kaggle Data](https://www.kaggle.com/c/quora-insincere-questions-classification/data)

#### Data fields

* qid - unique question identifier
* question\_text - Quora question text
* target - a question labeled "insincere" has a value of 1, otherwise 0

Test dataset - ~ 376K rows

Training dataset - ~ 1048K rows

## **Data Wrangling**

[Data Wrangling Jupyter notebook](Data%20Wrangling%20Python%20notebook)

**Main steps**

* Data Extraction
* Data Cleansing

The aim of this data wrangling exercise was to perform text processing on ‘question\_text’ column of the Dataframe which is going to be cleansed, pre-processed and broken down into independent categorical features before performing modeling on the data as against the ‘target’ column which would be the dependent feature for this dataset.

Essential steps performed were:

* **Merge training and test data set** 
  + Total question entries **– 1424K**
* **Missing data manipulation**
  + Since there are limited columns in the dataframe, finding reason for why the data was missing and handling it using techniques other than removing those entries didn’t make sense. Hence, null data handling was done by simply removing null entries of 'target' column of the dataframe.
* **Datatype conversion**
  + Especially of ‘target’ column questions marked as 0 and 1 identified as ‘sincere’ and ‘insincere’ questions into numeric variable which is going to be the basis for classification i.e. dependent variable to perform classification on.
* **Converting all letters to lower or upper case**
  + For the sake of consistency of text in the dataframe.
* **Removing punctuations**
  + This step allowed me to remove this set of symbols [!”#$%&’()\*+,-./:;<=>?@[]^\_`{|}~] which essentially would be not very helpful for the classification of data as sincere vs insincere.
* **Removing stop words, sparse terms, and particular words**
  + “Stop words” are the most common words in a language like “the”, “a”, “on”, “is”, “all”. These words do not carry important meaning and are usually removed from texts. It is possible to remove stop words using Natural Language Toolkit (NLTK), a suite of libraries and programs for symbolic and statistical natural language processing.
* **Stemming using NLTK library**
  + Stemming is a process of reducing words to their word stem, base or root form (for example, books — book, looked — look). I applied this step to the dataframe using Porter stemming algorithm that removes common morphological and inflexional endings from words.
    - <https://pythonprogramming.net/stemming-nltk-tutorial/>
* *Other steps included*
  + **Removing white spaces and newline characters**
  + **Removing entries with very long words (likely junk values)**
    - In this project, after applying all the above steps, it was my best judgement to remove entries with words > 15 characters.
    - After above step, it was observed that alphanumeric values (that have mix of numbers of strings) can be filtered out, however, there were certain question\_text entries with http OR www values AND could have a mix of numbers and characters AND still could exceed the threshold word length (as described in the above point) that we are looking to filter, hence they need to be retained.
  + **Removing entries with small text length (wouldn’t aid the analysis of the project)**
    - Judgement was to get rid of entries with text length < 10 characters.
  + **Generating new columns**
    - Created ‘total words’ and ‘question length’ which would be a

**Data Wrangling Output data**

After data extraction and cleansing, the final dataset contains 1024K – 39% of the dataset was trimmed as a part of the Data wrangling exercise. This cleansed dataframe will be further used for EDA, pre-processing and modeling steps

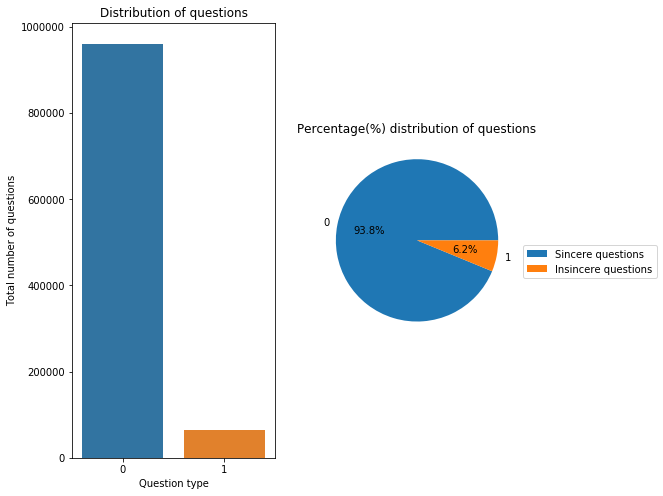
## **Exploratory Data Analysis**

[EDA report](https://github.com/shalin4788/Springboard/blob/master/Capstone%20Two/EDA_Quora%20Insincere%20Capstone%20Data.ipynb)

Conducted EDA on the Quora Insincere cleansed dataset

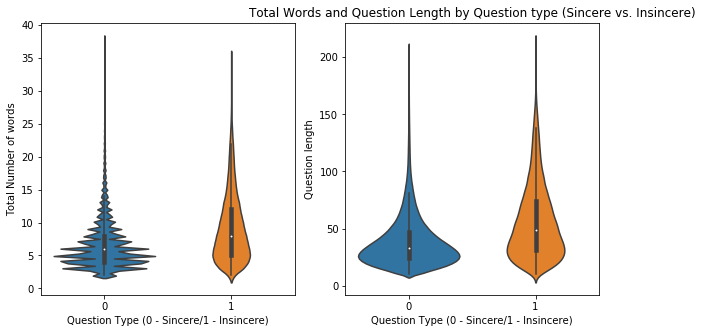
**Inferences**

* Insincere questions contribute to 6.2% of the dataset. It would be essential to classify sincere and insincere questions
  + It is essential that we have fewer false positives and false negatives both. We will revisit this in the modeling step on which model is able to classify this data in the most optimal manner.



**Image 1: Sincere vs Insincere question distribution**

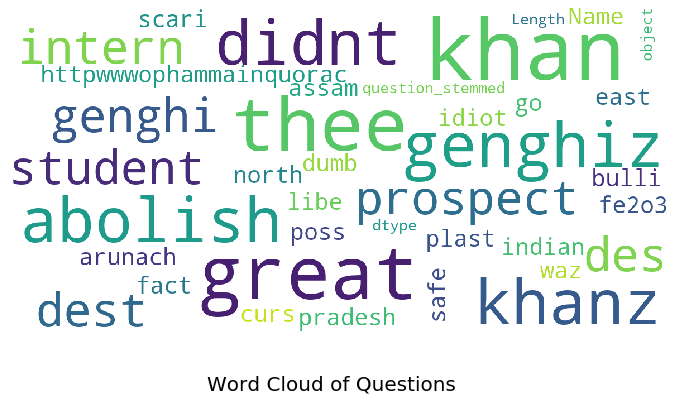
* We can see that the insincere questions have more number of words as well as characters/ question length compared to sincere questions (See Image 2). So, this might be a useful feature in our model.



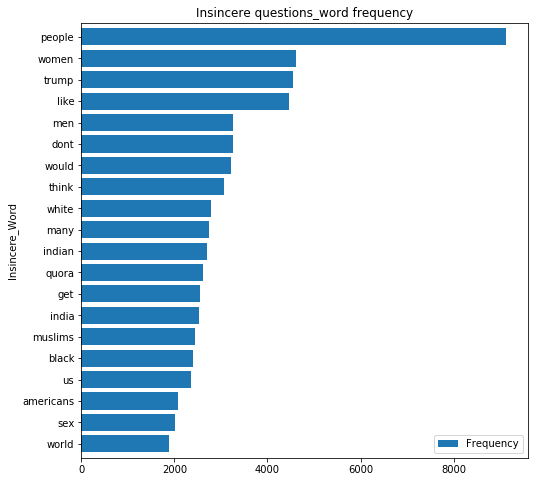
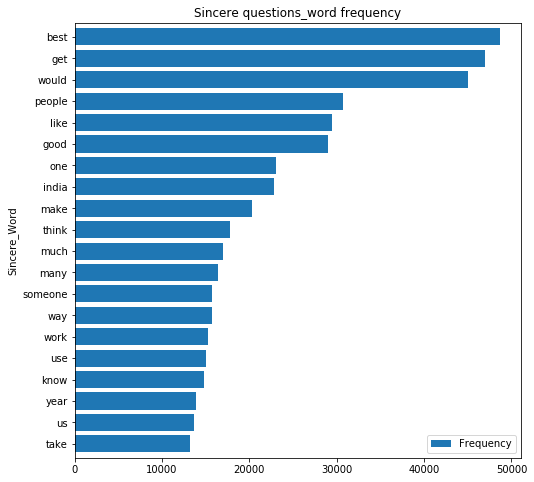
**Image 2: Violin plot of sincere vs insincere questions**

**(Question length and # of Words)**

* **Word cloud of Quora text**

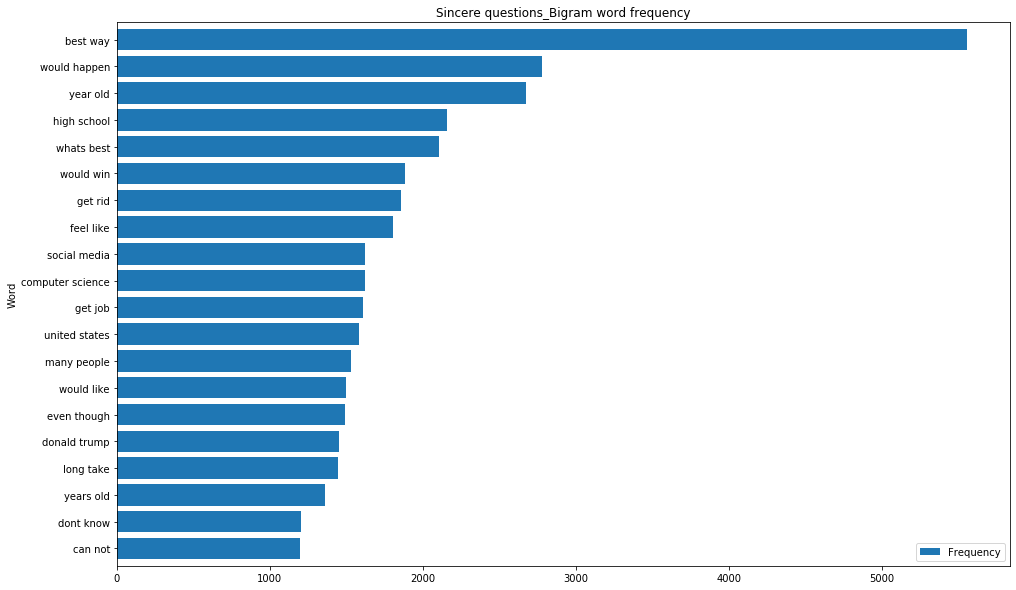


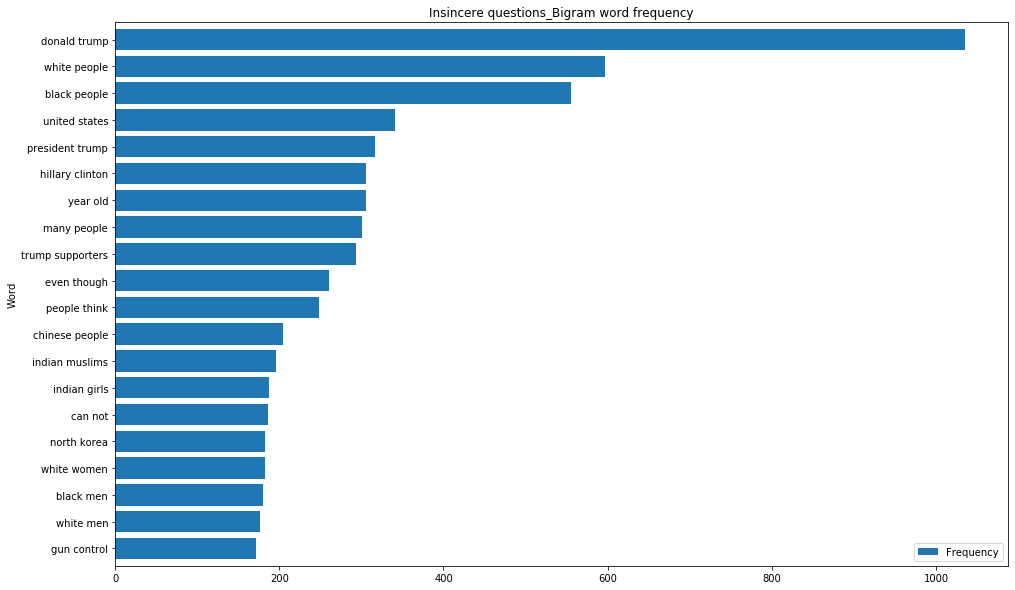
**Image 5: Word cloud of Quora text**

* One step was to plot unigrams and bigrams of words across the data set – for both sincere and insincere questions
  + To do this, I created 2 dataframes split from the cleansed dataframe classified by the ‘target’ variable – 0: sincere vs 1: insincere
  + Then, I determined the most frequent occurrences of unigrams and bigrams by plotting a horizontal bar chart – See image 4 and 5
* **Unigram of word frequency **

**Image 4 – Unigrams of most frequently occurred Sincere and Insincere words**

* **Bigram of word frequency**

****



**Image 5 – Bigrams of most frequently occurred Sincere and Insincere words**

## **Pre-processing and training data development**

[Feature Engineering Notebook](https://github.com/shalin4788/Springboard/blob/master/Capstone%20Two/Preprocessing%20and%20Training%20Data%20Development.ipynb)

**Steps involved**

* Standardizing the magnitude of numeric features using a scaler function
  + Standardization of numeric feature – ‘average\_word\_length’ in the dataset using MaxAbsScaler
* Performing Count vectorization OR TFIDF on categorical feature to fit and transform data
  + Tfidf involves ignoring common words which was applied on the dataset breaking down the ‘question\_final’ (originally question\_text’ column) into 200 categorical features
* Splitting the standardized and vectorized dataset (after performing above 2 steps) into test (30%) and training (70%) datasets
  + 717K entries in training dataset
  + 307K entriest test dataset

## **Model Selection (Algorithms & Machine Learning)**

[ML Notebook](https://github.com/shalin4788/Springboard/blob/master/Capstone%20Two/Quora%20Insincere%20Classification%20-%20Modeling%20Step.ipynb)

This is a classification problem, in supervised learning. Here I applied the following classification models on the training dataset after breaking the training dataset consisting of 717K quora questions into training and validation data:

* Logistic Regression
  + Normal
  + With L2 regularization
* KNearest Neighbors model
* Naive Bayes model
* Decision Tree model
* Random Forest (Ensemble) model
* Gradient Boost (Ensemble) model

### Model Evaluation

* Used metrics like **Precision, Accuracy, Recall, f1 score, ROC AUC score** to evaluate model performance
* Built **confusion matrix** to see false positives and false negatives after running each model against the training dataset and computing predictions against validation dataset
* Since we want to weed out insincere questions from Quora dataset, it is important to have a high f1 score (in this case taken weighted average of positive (insincere questions) and negative (sincere questions) observations) since it is best to have a low false positive rate as well as false negative rate
  + **Reason** - It is important to flag insincere questions and at the same time, not flag sincere questions

#### Final metrics for evaluation

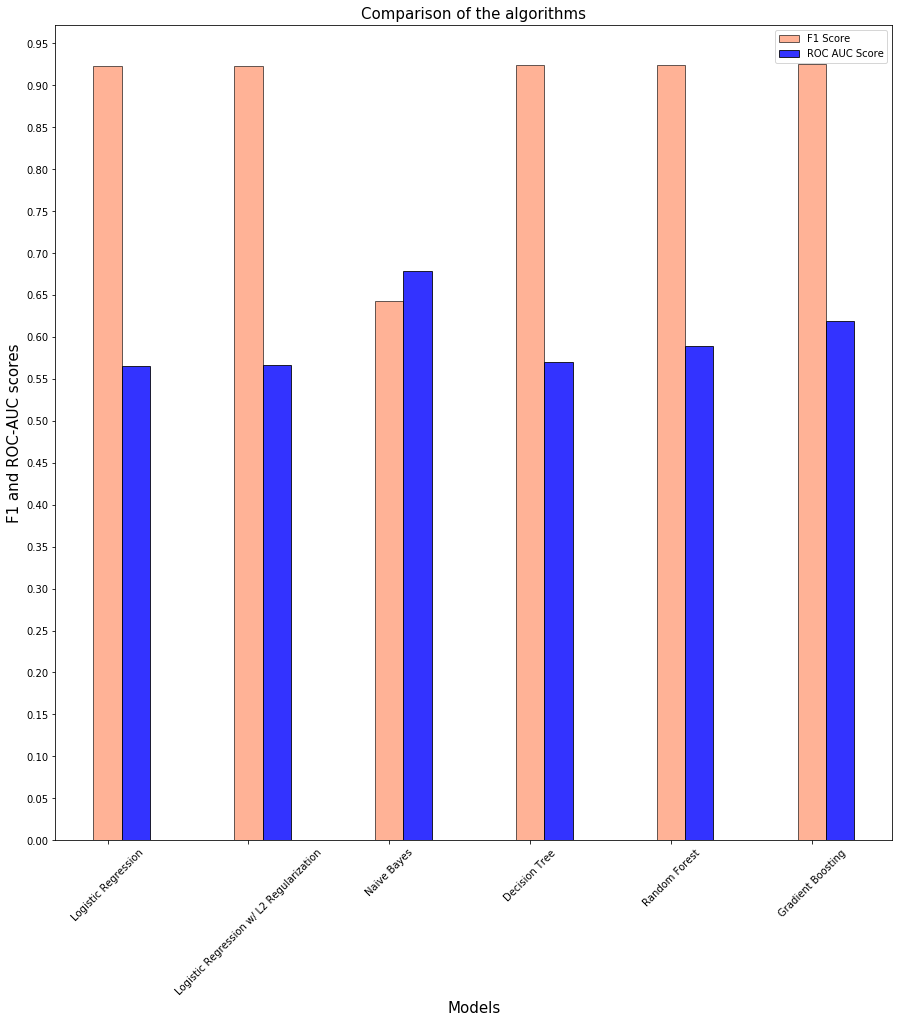
* **f1 weighted score** and **ROC AUC score** were used for shortlisting best performing models
* Also, ROC AUC score is important to see how well classifier performed
  + So, I chose to plot 'f1 score (weighted)' and 'ROC-AUC' scores to compare models

By that logic, the two best performing models are the Random forest and Gradient Boosting classifiers (see reasoning below)

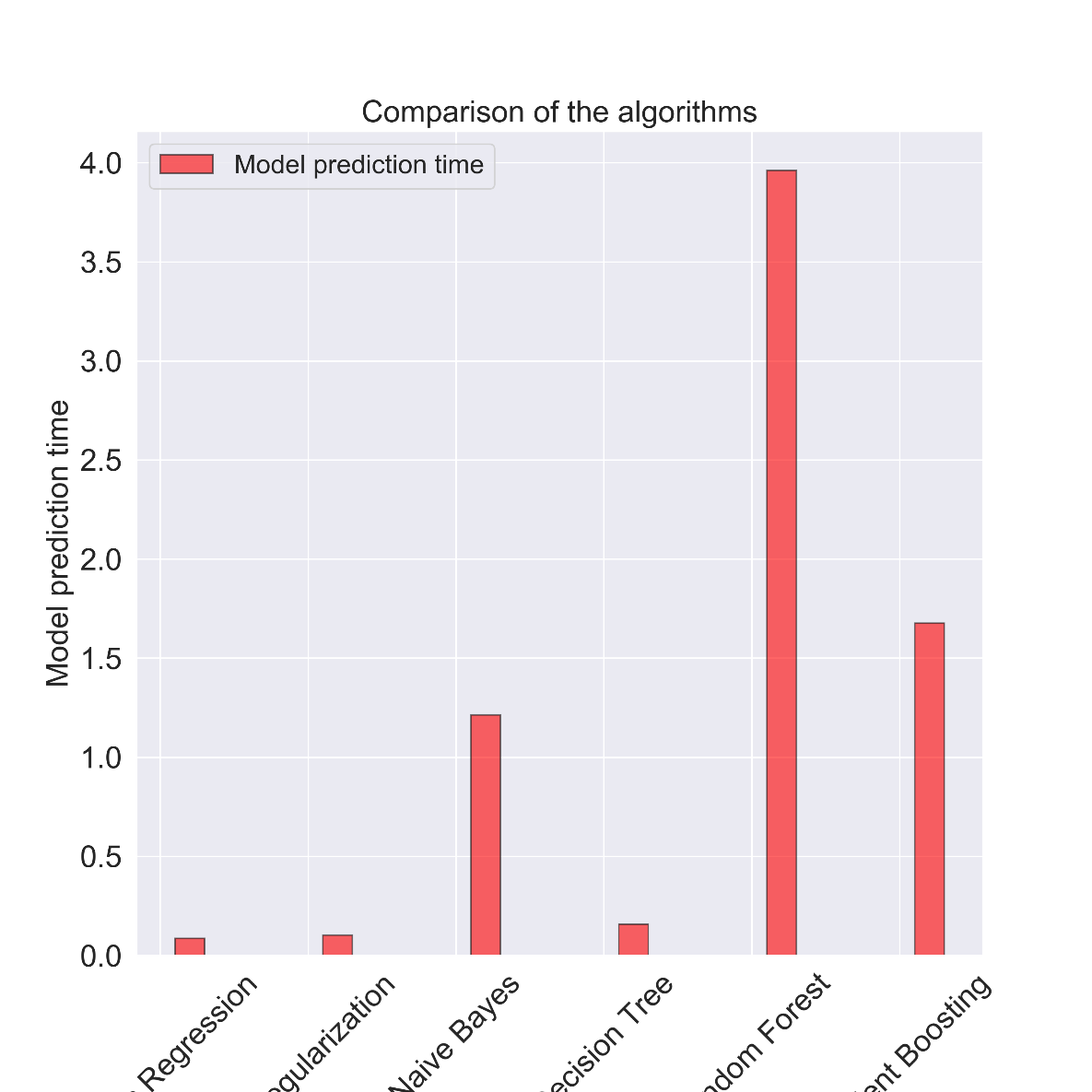
#### Model comparison

* **Logistic Regression and even Logistic Regression w/ L2 regularization** yielded high f1 scores but very low ROC-AUC scores, hence were rejected.
* **Naive Bayes Classifier** gave an low f1 score but a very high true positive rate and ROC AUC score, since it goes with the assumption that features are independent of one another when conditioned upon class labels which is not true in this case. Hence the model can be deemed not very useful for this use case.
* **KNN model** was also attempted, but it was taking forever to run considering the size of the dataset, in this case n is close to 1 mn records, hence the computational time is around O[N log(N)] time - hence the algorithm was stopped midway and rejected.
* After multiple trials, it seems like **Decision tree (w/ gini impurity)** at max\_depth = 10 using gini model, gives a better computational performance and is extremely fast. It has a better ROC AUC score compared to more simpler models like Logistic regression and Naive Bayes classifier, but since decision tree is more susceptible to overfitting and performance erosion as number of trees increase, it is better to not go with this model.
* **Random forest**, another ensemble Decision tree model results in a better ROC AUC score compared to Decision trees. Though computationally slower and little more complicated to comprehend when more hyperparameter tuning is performed with addition of more features and trees, it is likely to enhance model prediction and scores which could be advantageous for this dataset.
* **Gradient Boosting** after performing multiple hyperparameter tuning at just 1 learning rate and low max depth and max\_estimators performed pretty well on F1 score and best in terms of ROC AUC scores.

**Conclusion** - Based on this, RF and Gradient Boosting are chosen and in the next section, I will try to perform GridSearchCV and hyperparameter tuning on atleast one of the 2 models

****

**Image 6: Model comparison (Weighted f1 score and ROC AUC score)**

****

**Image 7: Model comparison (prediction time on validation dataset)**

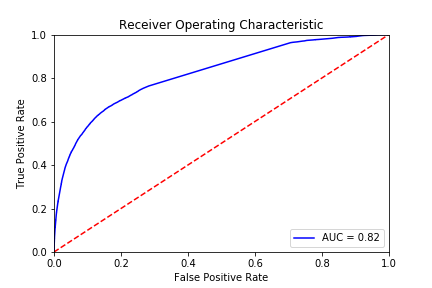
**Conclusion** **1**–

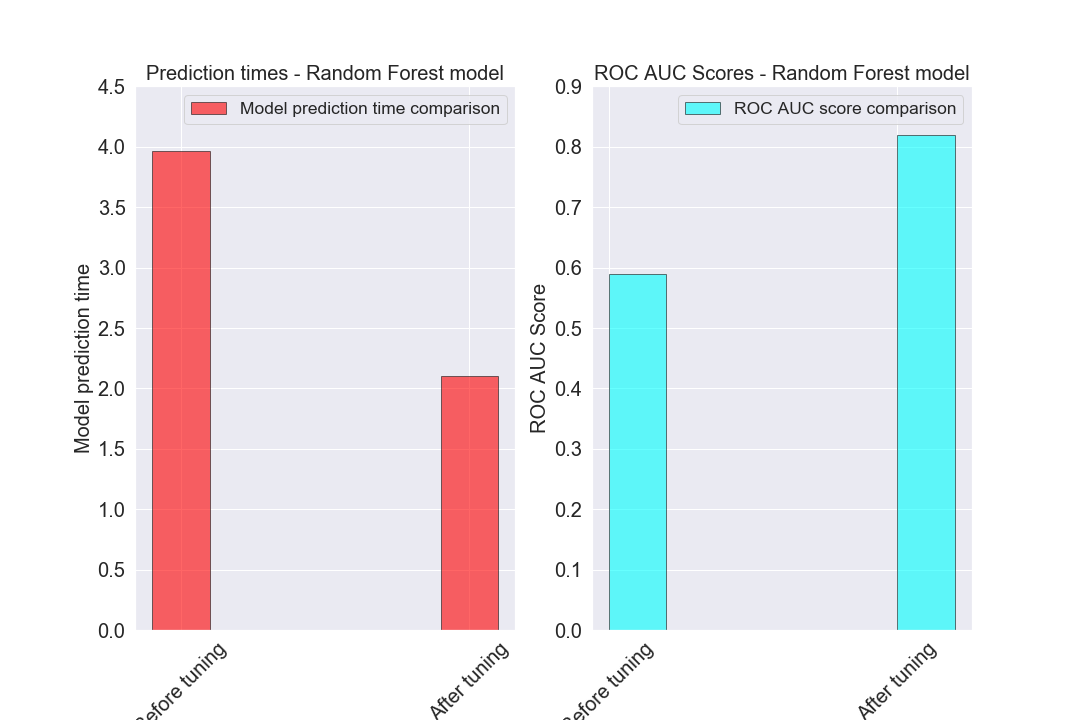
* Decision tree has a better weighted f1 score than Random forest classifier model.
* Gradient boosting model has a better ROC AUC score than Random forest classifier model.
* All other less complex models have a better prediction time than Random forest classifier model.
  + Prediction time on the validation dataset after fitting the random forest model to the training data is the worst for Random forest model (as seen in Image 7), yet for the purpose of this project and further model hyperparameter tuning, Random forest is chosen.

### Hyperparameter tuning

Applying Grid search CV for hyperparameter Tuning:

* Additionally, hyperparameter tuning was done on the best performing shortlisted model(s) – in this case, **Random forest model** to enhance performance.
* Also, Cross validation using Grid search was applied to pick up random pairs of training and validation data by splitting the training set into k smaller sets, where a model is trained using k-1 of the folds as training data and the model is validated on the remaining part.
* It was observed that there was a great improvement in ROC AUC score when hyperparameter turning was done on Random Forest model. However, the mean of f1-weighted score after hyperparameter tuning and cross validation was almost around the same range as before
* Hence for this use case, Random forest model was deemed best amongst other models in terms of precision and ROC-AUC scores
  + ROC AUC score - **0.82**
  + f1 score (weighted) - **0.917**



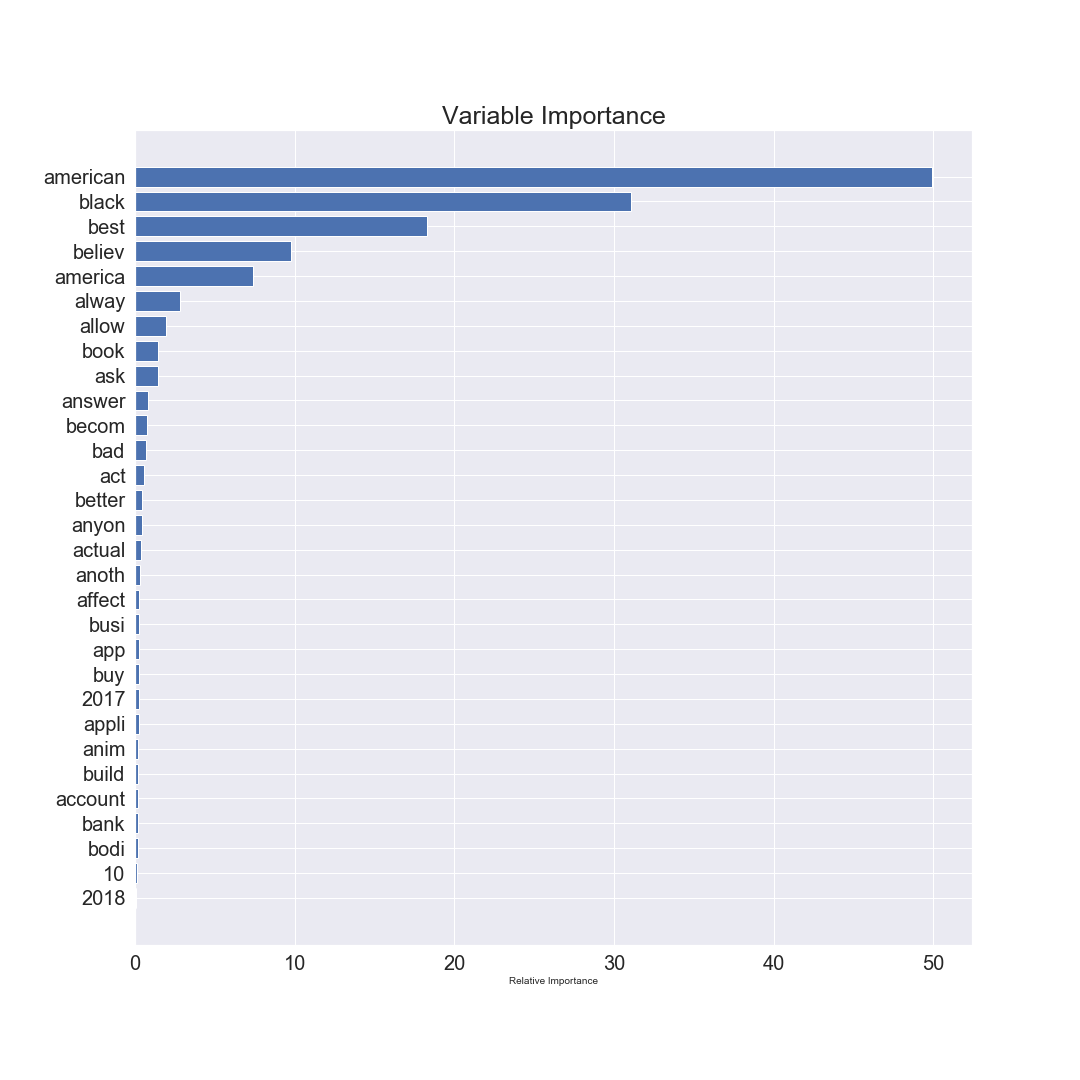
**Image 8: ROC AUC score (on validation data)**

**Image 9 – Comparison of metrics before and after tuning**

**Conclusion** **2**- This clearly indicates that apart from the ROC AUC score improvement, the prediction time on validation data set also fell sharply to ~2.1 seconds, a remarkable improvement for Random forest classifier model

### Variables with most feature importance

See below plot to see what features (unigrams) held maximum importance in classification and Random Forest classifier model prediction



### Saving the model

After optimizing the Random forest classifier model, the model was saved

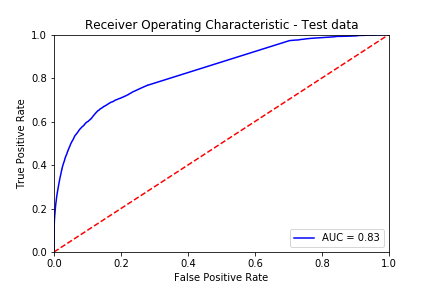
[Saved model](https://github.com/shalin4788/Springboard/tree/master/Capstone%20Two/models)

### **Prediction on unseen test data**

**Conclusion 3 -** The scores below are pretty good and very close to the scores obtained on validation dataset after performing cross validation, which is conclusive that model selection was good and the classification done using Random forest model is a good indication of a significantly accurate prediction on unseen test data set consisting of 307K Quora questions

This model was applied on the test data set to achieve the following scores:

* Test dataset: Prediction time – 1.67 seconds
* ROC AUC score - 0.83
* f1 score (weighted) - 0.916



**Image 9: ROC AUC score (on unseen test data)**

### **Future Score**

* Hyperparameter tuning can be performed on **Gradient boosting** models but limiting scope of this notebook to only tuning Random Forest classifier model
* Also, the number of max features aka categorical features chosen for this modeling step was only 200 due to CPU constraints causing overcommit memory issues
* In a real world, with the usage of sophisticated processors and availability of more resources, model performances can be enhanced further by choosing 500- 1000 max features in the feature engineering step without getting rid of important features and performing hyperparameter tuning on multiple parameters as a baseline, but that couldn't be covered in the scope of this project