QUORA INSINCERE QUESTIONS REPORT

HAMISH PROSSER

INTRODUCTION

As the world becomes more globally connected through the internet, there is an increasing number of malicious actors turning to the internet to rile people up and spread hate. The website Quora was launched in 2010 to create an environment where people all around the world can have their questions answered. However, one problem the website has had is many people asking what Quora considers "insincere" questions (examples in appendix 1A). Questions that have a specific motivation behind them, whether deliberately inflammatory, rhetorical, racist and many other similar categories. These questions are problematic as they degrade Quora's reputation as a forum to get reliable answers and are designed to stir up conflict. As more people use Quora the number of these questions increase and therefore Quora has been and is using machine learning to help identify these questions and flag them to make the moderators job easier, and to reduce this type of behavior on their website. Quora set up a competition on Kaggle https://www.kaggle.com/c/quora-insincere-questions-classification/data to allow data-scientists from around to world to contribute to solving the problem.

The competition is to, given a sentence (question) of English text, classify the sentence as either sincere or insincere. To solve the problem multiple methods of binary classification are examined along with different methods of text preprocessing including word stemming and removal of stop-words. The evaluation method for the competition is F1 score which is derived from the number of positive class classifications and false negative classifications. This topic was chosen to: Allow the exploration of different models for binary classification. To explore different methods of text processing. To examine methods to deal with imbalanced classes dataset and, to examine the competitiveness of a non-deep learning model in a deep learning dominated competition.

IMPLEMENTATION

THE DATASET

The dataset provided by Kaggle contains 1.3 million training instances. The training data was sampled to run support vector machines and random forest classifiers in a reasonable amount of time. Sampling was done by taking the first 10.5% (137142 rows) and splitting that into a 75% training set (102856 rows) and 25% independent test set (34286 rows). The data set has a significant class imbalance. The whole dataset is 6% insincere (class 1) questions and 94% sincere (class 2) questions. This class imbalance is preserved in the subset used for training and testing (Appendix 2A).

Class imbalance can pose a significant problem for binary classification tasks as models can learn to classify based on the class distribution in the data rather than learn how features properly correlate to a class (Krawczyk, 2016). This is especially important for this problem as the class we are more interested in is the minority class. There are many methods for dealing with this imbalance, the primary method used here is using the balanced class-weight parameter for the sklearn models, which adjusts the class weights inversely to their proportion in the data, the use of a random forest model which has been shown to be robust to class imbalanced problems (Krawczyk, 2016), and the metric evaluation of choice the F1 score.

TEXT PROCESSING METHODS

To use text as features in a machine learning model a common approach to use is the bag of words method. The bag of words approach converts each question sentence into a numerical vector representation of the occurrence of a given word in a sentence given the entire corpus of words in all the training set. Sklearn has many methods to achieve this processing. For this project both CountVectorizer and TfidfVectorizer are used. CountVectorizer creates a sparse matrix of token counts whereas TfidfVectorizer calculates the token frequency relative to the inverse of how many times it occurs in all documents. TfidfVectorizer decreases the weight given to tokens that occur many times in many documents and gives greater weight to tokens that occur fewer times in all documents.

Before being converted into the bag of words representation, all sentences had punctuation removed and all words less than 2 characters were removed as a requirement for Sklearns' vectorizers. Additionally, all characters are converted to lowercase.

Stemming words is a process that has been shown to be affective in many text classification problems (Bounabi, Moutaouakil, & Satori, 2019). Stemming words is the process of removing or changing suffixes on word primarily with different temporality or plurals to singulars, to reduce the feature set size and the ambiguity of words in the corpus. For example, lies and lied both become lie. Both words have very similar meaning, however the machine learning models will treat them separately if they are not stemmed. This project uses the natural language toolkit (NLTK) porter stemmer for stemming.

Stop word removal is often an important step in improving the performance of text classification models. Stop words are words that occur at a high frequency in a given language, such that they have no statistical significant impact upon a classifier. For example, common stop words in English include "the", "and", "at". However, as the dataset is short questions asked on Quora rather than long book passages for example, stop words may comprise a large proportion of the text itself.

The vectorizers above by default only vectorize individual words, therefore the models will only look at the impact of the words themselves with disregard to the occur in which they occur in a sentence. To preserve some order of words models are tested using different values of ngram_range. Ngrams refer to the number of words that comprise a single token, for example a two ngram is a single token that is composed of two words.

MACHINE LEARNING ALGORITHMS

Binary text classification is a task that has been studied extensively in machine learning literature. Many models have been shown to consistently perform well including, logistic regression, multinomial naïve bayes, support vector machines, and random forest (Bounabi, Moutaouakil, & Satori, 2019). These models are all used in this project and the final model combines the predictive power of all four.

MODEL HYPERPARAMETERS

Each model estimated has several hyperparameters that can be tuned to find the optimal level for a given dataset. To find the optimal parameters, Sklearn grid search cross validation was used to find the best parameters for each model. 3-fold stratified cross validation was used for each grid search for time concerns, ideally 5-fold cross validation would have been used for more robust results. For all models that allow it, the class weight hyperparameter was set to balanced. In addition, ngram_range and stopwords parameters of the vectorizers are grid searched.

LOGISTIC REGRESSION

Solver used was liblinear, max iterations increased to 300 to allow adequate time for convergence.

HYPERPARAMETER	VALUES TESTED	MEANING

PENALTY	L1, L2	Penalty term used for regularization. L1 adds the absolute value of the magnitude of the coefficients as a penalty. L2 adds the squared value of the coefficients as a penalty.
C	0.01, 0.1, 1, 10, 100	Inverse of regularization strength. Larger values imply less regularization.
VECTORIZER: NGRAM_RANGE	(1,1), (1,2),(1,3), (1,4)	For each tuple (x_min,x_max), all values of ngrams of $\min_x \leq x \leq \max_x$, are used. (1,2) means both 1 token ngrams and 2 token ngrams are used.
VECTORIZER: STOP_WORDS	None, "English"	Whether stop words as defined by Sklearn will be removed.

MULTINOMIAL NAÏVE BAYES

HYPERPARAMETER	VALUES TESTED	MEANING
ALPHA	0.01, 0.1, 1, 10, 100	Amount of Laplace smoothing. Adding dummy counts of word frequency to the corpus to not allow a 0-probability prediction occurring for unseen words.
VECTORIZER: NGRAM_RANGE	(1,1), (1,2),(1,3), (1,4) ,(2,2)	See logistic regression above
VECTORIZER: STOP_WORDS	None, "English"	See logistic regression above

SUPPORT VECTOR MACHINE

As the dataset used is large, the sklearn LinearSVC implementation of the SVM is used for much faster computation. The LinearSVC only supports the linear kernel. Loss is set to square hinge and dual is false to use L1 regularization. SVM max iterations set to 2000 to allow convergence.

HYPERPARAMETER	VALUES TESTED	MEANING
PENALTY	L1, L2	Same as logistic regression
C	0.01, 0.1, 1, 10, 100	Same as logistic regression
VECTORIZER: NGRAM_RANGE	(1,1), (1,2),(1,3)	See logistic regression above
VECTORIZER: STOP_WORDS	None, "English"	See logistic regression above

RANDOM FOREST CLASSIFIER

As the dataset is large and large random forests can take a substantial amount of training time, the max_features allowed for each tree parameter is set to log2 to reduce computation time and reduce the chance of overfitting.

HYPERPARAMETER	VALUES TESTED	MEANING
N_ESTIMATORS	25, 50, 100, 250, 500	The number of decision trees in the forest.
MIN_SAMPLES_SPLIT	10%, 5% 1%	As a percentage of the training set, the number of data points needed to split an internal node on. Lower values imply deeper trees.
VECTORIZER: NGRAM_RANGE	(1,1), (1,2)	See logistic regression above. Fewer used as ngram_range creates a very large amount of

		features which significantly slows down training speed.
VECTORIZER: STOP_WORDS	None, "English"	See logistic regression above

MODEL EVALUATIONS

The method of evaluation for submissions on Kaggle is the F1 score between predicted labels and the true labels. The F1 score is defined as:

$$F_1 = \frac{TruePositive}{TruePositive + \frac{1}{2}(FalsePositive + FalseNegative)}$$

This F1 score is the harmonic mean of precision and recall and is used for evaluating all the models presented here. The F1 metric heavily penalizes false positives and negatives which is important for the dataset given the significant class imbalance. For example, a model that predicts class 0 on every label in the test set would score very high on model accuracy, however the number of false positives would be substantial and F1 score would be terrible. Therefore, the F1 score works very well for this problem.

To determine the best model for this problem, the models were evaluated in 4 stages all on the F1 metric:

A. Baseline models no hyper parameter tuning, no vectorizer manipulation.
B. Baseline models no hyper parameter tuning, English stopwords removed and words porter stemmed
C. Grid Search 3-fold cross validation with Count Vectorizer:
    1. Words are stemmed
    2. Words are not stemmed
D. Grid Search 3-fold cross validation with Tfidf Vectorizer
    1. Words are stemmed
    2. Words are not stemmed

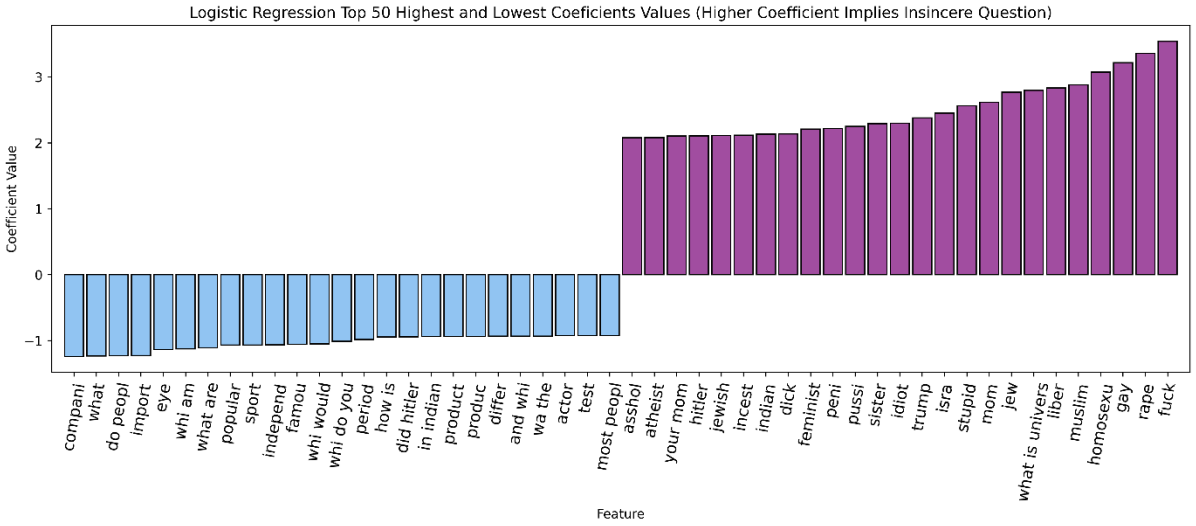
EXPERIMENT RESULTS

The best models for each stage of A-D gridsearchCV are included in appendix 3A. From those results we can find the best model for each classifier to be, the default parameters from the implementation section are not included here:

MODEL	TEXT PROCESSING METHOD	VECTORIZER	PARAMETERS	MEAN 3-FOLD CROSS VALIDATION F1 SCORE
LOGISTIC REGRESSION	Porter stemming	Count, ngram:(1,3), stop words: None	C:1, penalty:L2	0.583
MULTINOMIAL NAÏVE BAYES	No porter stemming	Count, ngram: (1,3) , stop words: None	Alpha:0.1	0.536
SUPPORT VECTOR MACHINE	Porter stemming	Tfidf, ngram: (1,2) , stop words: None	C:1, penalty:L2	0.556
RANDOM FOREST	Porter stemming	Count, ngram:(1,1) , stop words: None	Min_samples_split: 1%, n_estimators: 500	0.559

From the data there are some important things to note, removing stop words caused all models to perform worse. Additionally, all the models except for the naïve Bayes performed better with porter stemming, and using the count vectorizer, except for the SVM. L2 regularisation strictly outperformed L1 regularisation in both faster training speed and better performing models. Larger forests of trees outperformed smaller forests.

Increasing the number of ngrams range also made for a huge increase in performance of the models. Extracting the coefficients and named features from the best logistic regression model we can analyse what words have the largest swing towards an individual class.



Unsurprisingly, racial, sexual and politically motivated terms are the most predictive towards an insincere question. However, it is important to note the significance of the ngrams, many of the most predictive tokens for both classes are bi and trigrams. It is observed that the unigram of “hitler” implies an insincere question, however the bigram of “did hitler” implies a sincere question shows that the model is learning language patterns that demonstrate sincerity. This analysis explains why bi and trigram models significantly outperform unigram models.

The precision, recall and F1 scores for each of the models for both classes on the independent test set is shown in appendix 3B. The scores used are from the class 1 graph.

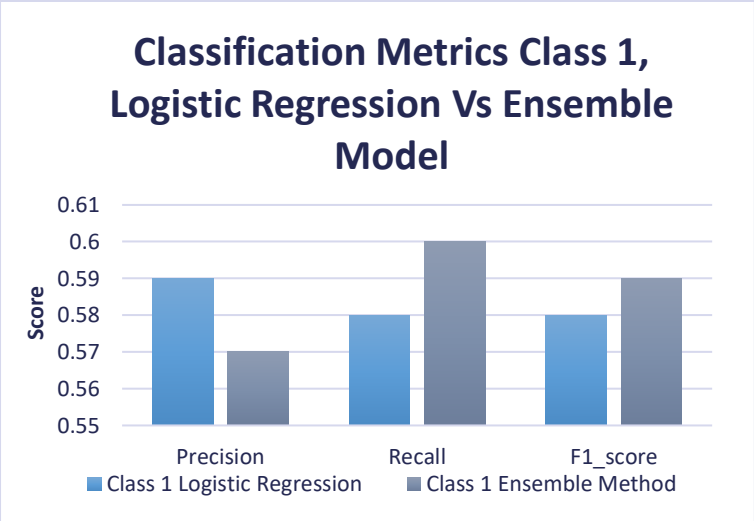
From the data, logistic regression has the highest F1 score for the training dataset, at 0.58. Of note we can see logistic regressions recall metric for class 1 is lower than the other models, and especially against the random forest model. Recall is defined as the number of true positives divided by true positives + false negatives. Therefore, the logistic regression is wrongly predicting class 1 more than the other models. The variance in classification metrics indicates there is room to improve the f1-score by combining predictions of models.

FINAL MODEL (COMBINING PREDICTIONS)

By combining predictions of all the models, the F1 score on the test set can be improved. Using a simple weighted averaging method of Logistic Regression 130%, multinomial 110%, SVM 110%, and random forest 50% (code in BestModel.py and Appendix 3C) the ensemble model achieves a better score than the sole logistic regression model. Where a model predicts class 0 the weighting has no impact, only when a model predicts the class 1 does the weighting have an effect.

The optimal threshold for maximising the F1 score for the ensemble model is 0.55. Indicating that at least 2 models need to predict class 1 for that to be the final prediction. This was found by creating a grid of thresholds from 0 to 1 in 0.5 increments and calculating the f1 score. Graph of f1_score to thresholds is in appendix 3D.

This ensemble model with optimised threshold can achieve an F1 score of 0.59, 0.01 higher than base logistic regression. These results are shown the graph below.



The ensemble model sacrifices some of the logistic regression precision on class 1 however, the recall is significantly higher which allows the F1 score to be better which is more important for this problem.

COMPETITION RESULTS

This competition was a kernel only competition, competitors only have access to the unlabelled test set, and competitors submit a Kaggle notebook. The submitted file is submitted.py however the file only works in Kaggles' virtual environment. The competition also provided 6gb file of word embeddings, pretrained vectors of word combinations that are used as the input for text classifications in neural networks (Liu, Liu, Yang, Yi, & Zhu, 2019).

The submission made to Kaggle was trained on the entire dataset using the ensemble model above, as text classification models greatly improve from seeing a much wider variety of words. The model took slightly less than 3 hours to run and scored a private score of 0.63.

The competition closed in early 2019 and all the winning models were neural network based. The highest private score was 0.713 (the competition winning score). Of the 1400 people that submitted a kernel (4000 joined the competition), my score of 0.63 places at about 1285 beating some of the simpler neural networks other competitors submitted.

The competition had a kernel run time length of 6 hours; my model took 3 hours to run primarily due to a random forest that was inappropriately large for the full dataset. Overall, considering the performance of the model hyperparameter tuned on only 10% of the data and using only half the available kernel runtime the model performed very well. Additionally, the model is explainable as to how it is making classifications, something that can be much harder with more sophisticated models such as neural networks.

CONCLUSION

Overall, the model was able to classify insincere questions with reasonable success relative to the winning score. The methods used to ensure the class imbalance did bias the models worked well. Porter stemming was successful for all but the naïve Bayes model and the term-frequency inverse document frequency vectorizer showed some success for the SVM.

Text classification is becoming more and more dominated by deep neural networks. From the code discussion page (https://www.kaggle.com/c/quora-insincere-questions-classification/code?competitionId=10737&sortBy=voteCount) it is immediately apparent that recurrent neural networks and long-short-term-memory models dominate this competition. Having surpassed many of the neural networks using embedding by combining simpler classification models with text preprocessing methods, demonstrates the value of these simpler models.

LIMITATIONS AND DISCUSSION

For singular models the overall success rate was logistic regression, support vector machine, multinomial naïve bayes and random forest classifier. Each model had their own strengths in terms of predictive power. Ideally, the multinomial naïve bayes and logistic regression models would have been hypertuned on a larger amount of the data as they were the fastest models to train by far, but to accommodate the SVM and random forest a small sample had to be used to avoid training models on different subsamples.

The random forest was capped at 500 estimators and a 1% node split. It is likely that increasing the size of the forest and decreasing the node split threshold would increase random forest performance up to a point, however the computing demands increase greatly with additional trees and features.

For future consideration, I would have liked to experiment with other methods of text processing such as lemmatization and other stemmers such as snowball. Further, for this binary classification task more time working with the individual classification thresholds of the models could make a large difference to the final F1 score of the model.

Finally, testing other more abstract methods of feature extraction such as the number of spelling mistakes in a sentence may prove useful in insincere question classification.

REFERENCES

Bounabi, M., Moutaouakil, E. K., & Satori, K. (2019). A comparison of text classification methods using different stemming techniques. *Computer Applications in Technology*, 1-7.

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 221-232.

Liu, W., Liu, P., Yang, Y., Yi, J., & Zhu, Z. (2019). A <word, part of speech> embedding model for text classification. *Expert Systems*, 1-3.

Nguyen, V.-T., & Anh-Cuong, L. (2016). Improving Question Classification by Feature Extraction and Selection. *Indian Journal of Science and Technology*.

NLTK. (n.d.). *Natural Language Tool Kit Documentation.* Retrieved from NLTK: https://www.nltk.org/

Sklearn. (n.d.). *CountVectorizer.* Retrieved from Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Sklearn. (n.d.). *Linear Support Vector Classifier.* Retrieved from Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC

Sklearn. (n.d.). *Logistic Regression.* Retrieved from scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Sklearn. (n.d.). *MultinomialNB.* Retrieved from Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

Sklearn. (n.d.). *Random Forest Classifier.* Retrieved from Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Sklearn. (n.d.). *TfidfVectorizer.* Retrieved from Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

APPENDIX

EXAMPLE TRAINING INSTANCES (1A)

qid	question_test	target
000685d7619cd29b567d	What is the difference between a politician and a leech?	0
000688582e3a2059d832	Why does China not sponsor a regime change in North Korea?	0
00068875d7c82a5bcf88	Why do Europeans say they're the superior race, when in fact it took them over 2,000 years until mid 19th century to surpass China's largest economy?	1
00069d1cac3cfd3c2ec4	Why does the cytoplasm built up inside my lip?	0

000a7ce3b24ed2a4098f	How hard is it to be accepted to CS MSc at UniversitÃ© de MontrÃ©al?	0
000a898069f8ee9c963f	With the introduction of GST, what is the meaning left of Maximum Retail Price of products as there is one nation one tax now?	0
000a898565e80fe124bf	Why are liberal minorities so voilent towards poeple with diffrent poltical beleifs? Should supporting trump be a sentence to be imprisoned or savegely attacked?	1
000e67648fce55f011be	Why do the Liberals who run schools choose not to have controlled access? The kids in Florida were killed due to an unlocked door.	1
000e6a4eaa936ee491d6	Among these practices, which do you consider as strenghts that any preschool should possess?	0
000e6a6cc25e7ca35108	How does the society react during your term breaks at NDA?	0
000e852896013d66612d	Do intelligent people ask questions?	0

CLASS BALANCE ACROSS DATASET AND SAMPLE 2A

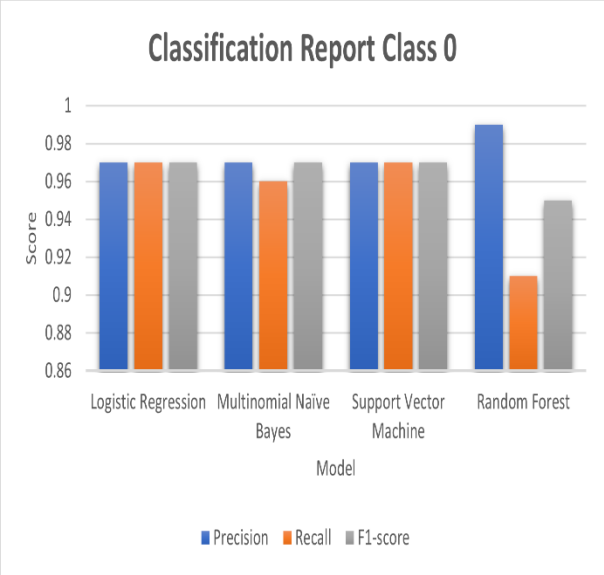
```
Full dataset: 1306122, 0.06187017751787352
Sampled Dataset: 137142, 0.06147642589432851
```

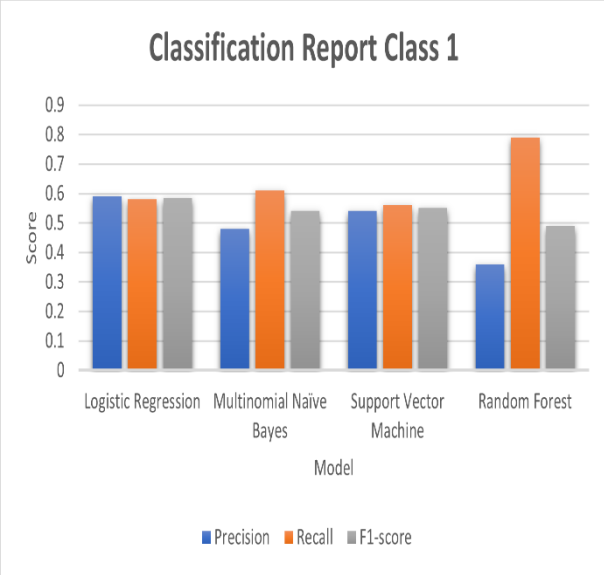
First value is size of dataset, second is the mean of the target values. For binary classification value is the class distribution.

MODELS A-D FULL CROSSVALIDATION SCORES (3A)

Baseline Models (A)	Best Parameters	Best Mean CV Score
Logistic Regression		0.546
Multinomial Naïve Bayes		0.481
LinearSVC		0.465
RandomForest (25 trees, 1% split)		0.47
Stemmed and Stopwords Removed(B)		
Logistic Regression		0.525
Multinomial Naïve Bayes		0.46
LinearSVC		0.396
RandomForest (25 trees, 1% split)		0.469
Grid Search Count Vectorizer (C,1) Stemmed	**Best Parameters**	**Best Mean CV Score**
Logistic Regression	Penalty : L2, C : 1, Ngram: (1,3), Stop_words: None	0.583
Multinomial Naïve Bayes	Alpha: 0.1, Ngram:(1,3), stop_words:None	0.53
LinearSVC	Penalty : L2, C : 0.01, Ngram: (1,3), Stop_words: None	0.556
RandomForest	N_estimators: 500,min_samples_split: 1%, Ngram: (1,1), stop_words:None	0.559
Grid Search Count Vectorizer (C,2) Not_stemmed		
Logistic Regression	Penalty : L2, C : 1, Ngram: (1,3), Stop_words: None	0.573
Multinomial Naïve Bayes	Alpha: 0.1, Ngram:(1,3), stop_words:None	0.536
LinearSVC	Penalty : L2, C : 0.01, Ngram: (1,3), Stop_words: None	0.552
RandomForest	N_estimators: 500, min_samples_split: 1%, Ngram: (1,1), stop_words:None	0.557
Grid Search Tfidf Vectorizer (D,1) Stemmed		
Logistic Regression	Penalty : L2, C : 10, Ngram: (1,2), Stop_words: None	0.582
Multinomial Naïve Bayes	Alpha: 0.01, Ngram:(1,3), stop_words:None	0.453
LinearSVC	Penalty : L2, C : 1, Ngram: (1,2), Stop_words: None	0.556
RandomForest	N_estimators: 500, min_samples_split: 5%, Ngram: (1,1), stop_words:None	0.54
Grid Search Tfidf Vectorizer (D,2) Not_stemmed		
Logistic Regression	Penalty : L2, C : 10, Ngram: (1,2), Stop_words: None	0.576
Multinomial Naïve Bayes	Alpha: 0.01, Ngram:(1,2), stop_words:None	0.463
LinearSVC	Penalty : L2, C : 1, Ngram: (1,3), Stop_words: None	0.555
RandomForest	N_estimators: 500, min_samples_split: 5%, Ngram: (1,1), stop_words:None	0.545

PRECISION, RECALL, F1-SCORE 3B



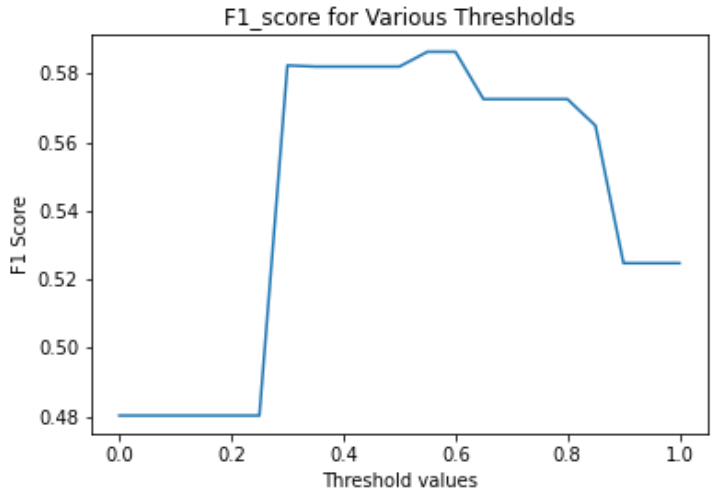


ENSUMBLE WEIGHTED PREDICTION ALGORITHM 3C

Code is included in a python file as well however I think it is important to have it here as it is quite important to the report in isolation.

```
def ensemble_predictions(predictions):
    #predictions is an 4 x len(train) array
    preds = []
    for i in range(len(predictions[0])):
        pred = 0
        for model in range(4):
            if model == 0: #logistic regression
                pred += predictions[model][i] *1.3
            elif model == 4: #random forest
                pred += predictions[model][i] *0.5
            else: #svm or MNB
                pred += predictions[model][i] *1.10
        if pred / 4 > 0.55: #threshold for class classification
            preds.append(1)
        else:
            preds.append(0)
    return preds
ensumble_preds = ensemble_predictions(predictions_all_models)
```

F1-SCORE FOR DIFFERENT THRESHOLD VALUES 3D



EXPERIMENTAL RESULTS

CORRELATION BETWEEN MEDIAN AND AVERAGE WORD LENGTH AND INSINCERE QUESTION

METRIC	CORRELATION
MEDIAN WORD LENGTH	0.041
MEAN WORD LENGTH	0.031

There is a slight positive correlation shown above for both measures. There may be merit to adding one of the above measures as a feature in a classifier.