Data 410 Project Rough Draft

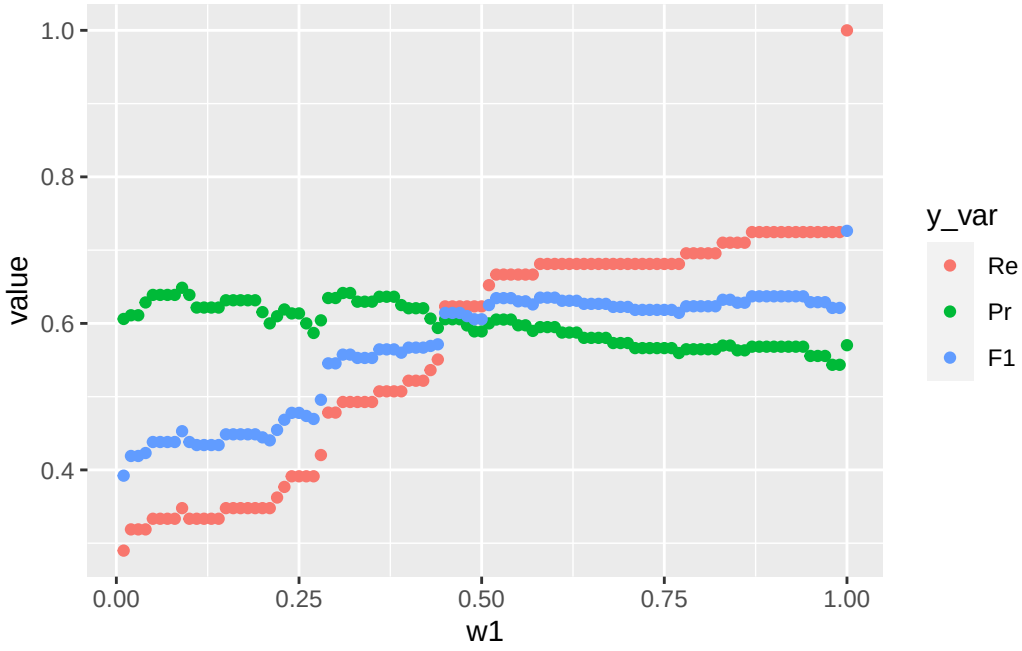
Daniel Krasnov, Keiran Malott, Ross Cooper

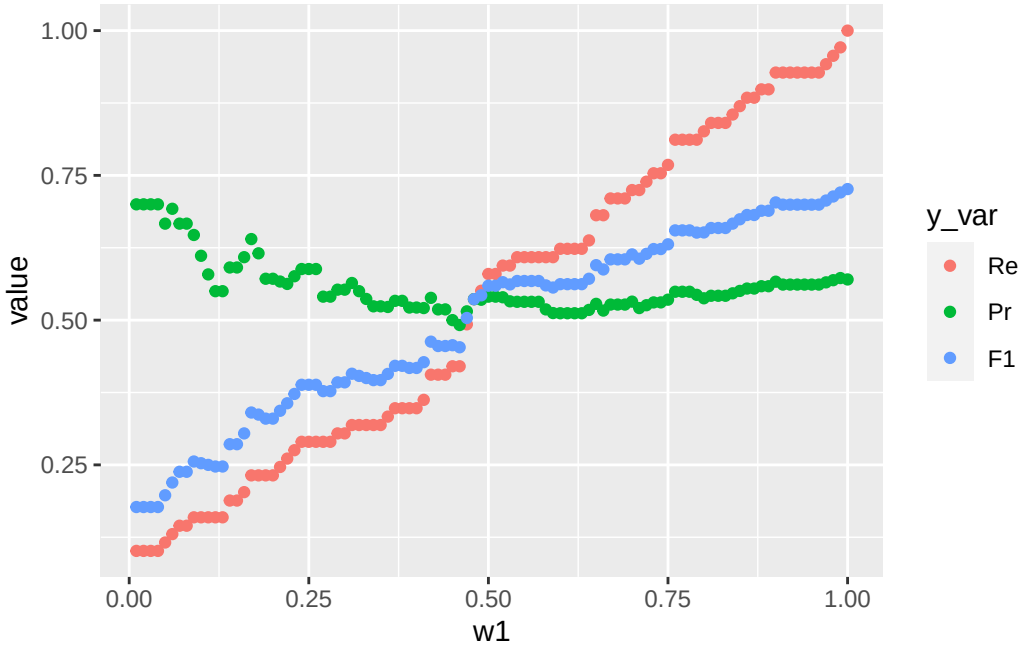
4/7/23

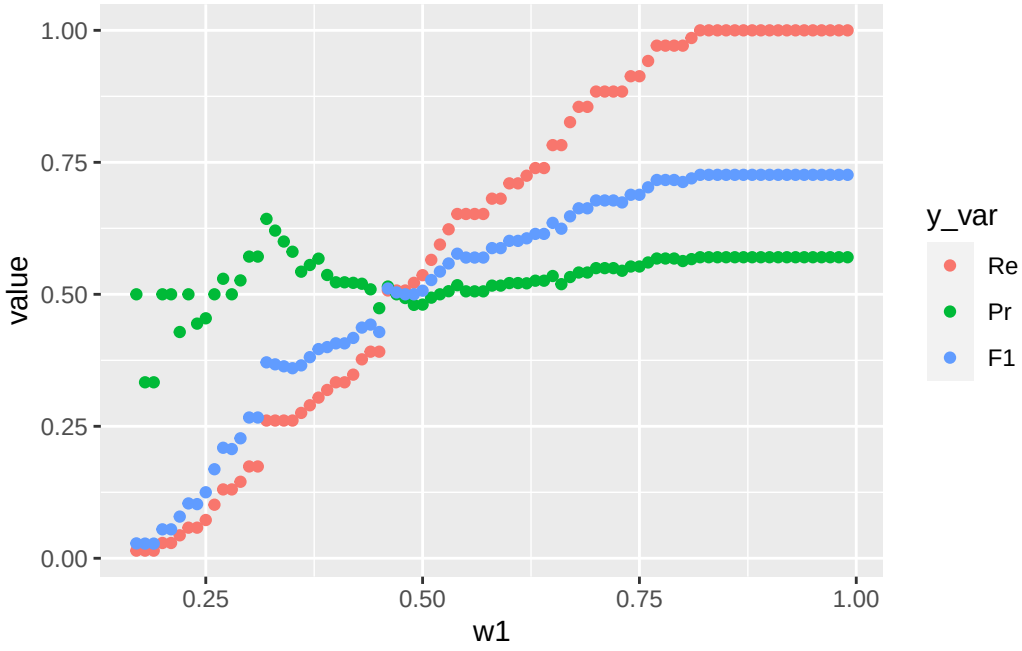
```
[1] FALSE
```

```
INFO  [17:20:09.205] epoch 1, loss 0.2236
INFO  [17:20:09.239] epoch 2, loss 0.0898
INFO  [17:20:09.251] epoch 3, loss 0.0427
INFO  [17:20:09.257] epoch 4, loss 0.0211
INFO  [17:20:09.263] epoch 5, loss 0.0141
INFO  [17:20:09.268] epoch 6, loss 0.0098
INFO  [17:20:09.273] epoch 7, loss 0.0071
INFO  [17:20:09.277] epoch 8, loss 0.0052
INFO  [17:20:09.283] epoch 9, loss 0.0039
INFO  [17:20:09.288] epoch 10, loss 0.0030

INFO  [17:20:12.068] epoch 1, loss 0.2220
INFO  [17:20:12.073] epoch 2, loss 0.0868
INFO  [17:20:12.079] epoch 3, loss 0.0451
INFO  [17:20:12.084] epoch 4, loss 0.0218
INFO  [17:20:12.089] epoch 5, loss 0.0137
INFO  [17:20:12.094] epoch 6, loss 0.0095
INFO  [17:20:12.100] epoch 7, loss 0.0069
INFO  [17:20:12.106] epoch 8, loss 0.0051
INFO  [17:20:12.112] epoch 9, loss 0.0038
INFO  [17:20:12.118] epoch 10, loss 0.0029
```




1.00
0.75
0.50
0.25
value
0.00
0.25
0.50
0.75
1.00
w1
y_var
Re
Pr
F1


1.00
0.75
0.50
0.25
0.00
value
0.25
0.50
0.75
1.00
w1
y_var
Re
Pr
F1

Introduction

Reddit is an American social news website that hosts discussion boards where users can share, comment and vote on various posts (Reddit wikipedia). These posts are housed in subreddits which are communities on Reddit focused on a specific topic.

When writing comments on Reddit, users will often write /s at the end of their post to indicate their comment is Sarcastic. This, coupled with Reddit's web scrapping Python API, provides a self labeled data set of sarcastic comments.

The goal our analysis will be to use the /s as a binary indicator of a comment being sarcastic and fit a Logistic regression model using various feature extraction methods. We can then explore this model's efficacy and optimize it for prediction.

Data Collection Method

On the subreddit dataisbeautiful one user posted the following figure (figure citation):

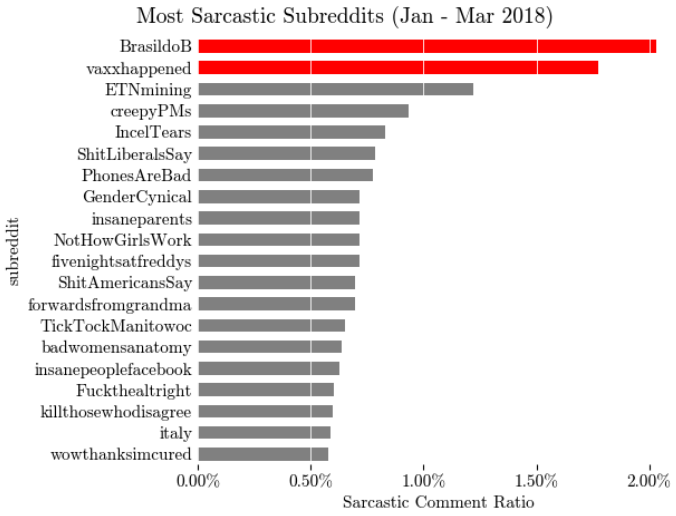


Figure 1: sarcastic_subreddits

We began by scrapping the top 10,000 posts from each of the above subreddits. We found that all the subreddits had approximately a 1:100 ratio for sarcastic to non-sarcastic comments. We constructed our first data set by sampling from all the above subreddits however, we found the data to be too 0 heavy and no model specification could learn an underlying relationship between words and sarcasm. We then attempted to fit models to various ratios of sarcastic to non-sarcastic comments. We found that Logistic regression began to perform reasonably well at a ratio of 1:1 sarcastic to non-sarcastic. We also found that models tended to perform

far better if all comments came from a single subreddit as opposed to multiple. As per our prelimnary results we opted for a single subreddit at a ratio of 1:1 sarcastic to non sarcastic comments. NotHowGirlsWork was found to have the largest count of Sarcastic comments at 321 therefore we selected this subreddit for our data set.

Variable Description

Our data set is constructed as follows:

Variable Name	Data Type
Body	String
Sarcastic	Binary Integer

Figure 2: data set table

where Body is the raw comment string and Sarcastic is 1 when /s is present in the comment and 0 when it is not.

Data Preprocessing

In Natural Language Processing there are various text preprocessing steps that are common to employ (text as data citation):

- Punctuation, whitespace, and number removal - any punctuation characters such as !, @, #, etc. as well as empty space and numbers are removed.
- Stopword Removal - removal of words that fail to provide much contextual information, e.g., articles such as 'a' or 'the'.
- Stemming - identifying roots in *tokens*, individual words, and truncating them to their root, e.g., fishing and fisher transformed to fish.

In our data set we first removed the /s from every sarcastic comment and preformed the above preprocessing steps.

Feature Extraction Methods

In order to use text as data in a Logistic regression we must numerically encode our strings. There are a plethora of feature extraction methods in NLP. For our analysis we compare TF-IDF, Word2Vec, and GloVe.

TF-IDF

Term frequency inverse document frequency (TFIDF) is a heuristic to identify term importance (text mining in R citation). It calculate the frequency with which a term appears and adjusts it for its rarity. Rare terms are given increased values and common terms are given decreased values (text as data citation).

TFIDF is given by

$$\text{TFIDF}(t) = \text{TF}(t) \times \text{IDF}(t)$$

where

$$\text{TF}(t) = \frac{\text{\# of times term t appears in a document}}{\text{\# of terms in the document}}$$

and

$$\text{IDF}(t) = \ln\left(\frac{\text{\# total number of documents}}{\text{\# number of documents where t appears}}\right)$$

In our analysis a document is a Reddit comment. After being preprocessed, the text of each comment is separated into tokens and has its TFIDF calculated. From there the TFIDF values are placed in a *Document Term Matrix* (DTM). This matrix has document ids as rows and tokens as columns. It is therefore a sparse matrix where entries are the TFIDF scores for corresponding tokens.

The DTM acts as the design matrix for our Logistic Regression model:

```
<<DocumentTermMatrix (documents: 6, terms: 8)>>
Non-/sparse entries: 1/47
Sparsity           : 98%
Maximal term length: 10
Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
Sample             :
    Terms
Docs common forevaaaaa husband lost potenti surviv two      will
  10      0          0       0    0       0      0   0 0.0000000
  5       0          0       0    0       0      0   0 0.0000000
  6       0          0       0    0       0      0   0 0.2352558
  7       0          0       0    0       0      0   0 0.0000000
  8       0          0       0    0       0      0   0 0.0000000
  9       0          0       0    0       0      0   0 0.0000000
```

Word2Vec

Word2Vec is a group of predictive models for learning vector representations of words from raw text. Word2Vec uses either the *continuous Bag-of-Words architecture* (CBOW) or the *continuous Skip-Gram architecture* (Skip-Gram) to compute the continuous vector representation of words. Both CBOW and Skip-Gram use shallow neural networks to achieve this, but CBOW predicts words based on the context and Skip-Gram predicts surrounding words given the current word (Efficient Estimation of Word Representations in Vector Space paper citation).

Each word is represented as a vector, and words that share common context are close together in vector space (Deep Learning Essentials textbook citation). Document vectors are representations of documents (Reddit comments) in vector space. A document vector can be constructed by summing the the word vectors from a common document and then standardizing them (word2Vec package citation). The design matrix for logistic regression can be constructed with the rows of the matrix as the document vectors. The resulting design matrix therefore has one row per Reddit comment and is as follows:

GloVe

Global vectors for word representation (GloVe) is an unsupervised learning algorithm which creates a vector representation for words by aggregating word co-occurrences from a corpus. The resulting co-occurrence matrix X contains elements X_ij representing how often word i appears in the context of word j (citation).

Next, soft constraints for each word pair are defined by:

$$w_i^T w_j + b_i + b_j = log(X_{ij})$$

where w_i is the vector for the main word, w_j is the vector for the context word j, and b_i and b_j are scalar biases for the main and context words. Finally, a cost function is defined:

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij})(w_i^T w_j + b_i + b_j - \log X_{ij})^2$$

Here f is a weighting function chosen by the GloVe authors to prevent solely learning on extremely common word pairs (citation):

$$f(X_{ij}) = \begin{cases} \left(\frac{X_{ij}}{x_max}\right)\alpha & \text{if } X_{ij} < XMAX \\ 1 & \text{otherwise} \end{cases}$$

To create the design matrix below, a vocabulary of the words in the corpus was created. Since this method creates a co-occurrence matrix, we prune all words which appear less than five times to reduce bias from less common words (citation). From there we constructed a term-co-occurrence matrix and factorized it via the GloVe algorithm. The resulting matrix consists of word vectors as rows, which are added together to create sentence vectors that are used to train the model:

Evaluation Metrics

Model performance is assessed based on classification performance. In sentiment analysis the most common metrics to tune model for performance are Precision, Recall, and F1 Score (citation).

Precision is the number of true positive divided by the number of true and false positives. Recall is the number of true positive divided by false negatives and true positive. It is the true positive rate. F1 Score is the harmonic mean of Recall and Precision (python learning citation) (add a CM and write the formulas).

For our analysis a true positive is a correctly predicting a comment is sarcastic.

Regression Analysis

This analysis is a comparison of logistic regression model performance when using 3 types of feature extraction. For each feature extraction we fit a base model. We then perform Princial Componenet Analysis (PCA) to reduced dimensionality and deal with multicollinearity. Finally we investigate LASSO models a means for dimensionality reduction.

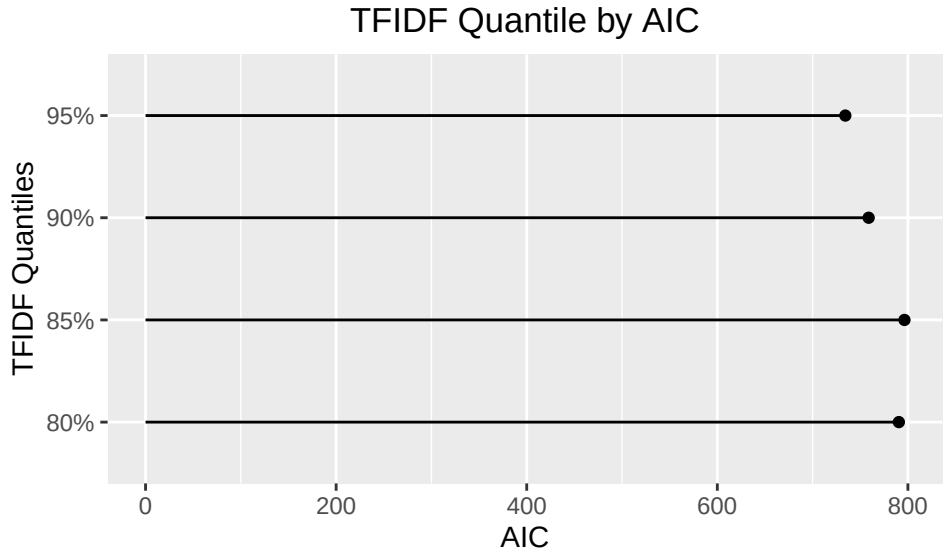
After model fitting we perform weighting on all 3 model types and decide a best model for each feature extraction method. Weighting in done by multiply each predictor by w, where $w \in (0, 1)$, if a comment is sarcastic and by $1 - w$ if a comment is not sarcastic. This is done for every w starting from 0.01 to 0.99 in increments of 0.01. We record testing and training metrics foe each model and discuss our optimal selection.

We hypothesis that models using GloVe as the feature extraction method will perform similarity to Word2Vec. TF-IDF will perform the worst. TF-IDF numerically encodes text based on rarity. We think this is too simple an approach to capture important sarcastic words. Word2Vec captures context and GloVe interprets word co-occurrences which we believe could both be suitable strategies to capture sarcastic structure in text.

Variable Selection

TF-IDF

After performing text preprocessing and TFIDF calculations the resutling DTM was 642×2074. This matrix has far too many columns compared to rows and so some dimensionality reduction was required. One way to do so is to filter away unimportant terms. This can be decided by the percentiles of the TF-IDFs. For a given percentile we can exclude columns of the DTM based on whether or not their values fall within that percentile. We do this in increments of 5 from the 5th percentile to the 95th percentile, fit a model, and recording the corresponding AIC. We opt to keep the DTM that produced the model with the lowest AIC.



The model corresponding to the lowest AIC had a DTM filtered to disclude TFIDF values below the 95th percentile resulting in a 512×74 matrix.

Word2Vec

Before fitting the base model or creating the design matrix, Word2Vec requires the user to specify the dimensions of the word vectors to be created and whether the CBOW or Skip-gram architecture should be used. The Skip-gram architecture has been shown to have higher semantic accuracy than the CBOW architecture (citation) so we believe it will preform better for predicting sarcasm, and have chosen to use it over the CBOW architecture when fitting the Word2Vec model. To decide on an appropriate vector dimension, the Word2Vec model was fitted with a range of dimensions from 1 to 200 in increments of 1. For each dimension of the Word2Vec model fitted, a logistic regression was fitted on that model and the testing

F1 score was calculated. However, the testing F1 score of the model is varies between the same vector dimension because Skip-Gram determines the vector representation of the text is stochastically. This means the predictors used for logistic regression and any evaluation metric, such as testing F1 score, are stochastically determined too. Therefore, to determine the optimal vector dimension, each dimension was fitted 5 times and a graph of the testing F1 score vs Word2Vec vector dimensions was produced (Figure xyz). We have chosen Testing F1 score to determine the optimal vector dimension because it is the primary metric used to evaluate the predictive performance of our fitted models.

Based on Figure xyz the optimal dimension of the Word2Vec model appears to be roughly dimension = 140. This is because lower dimensions have lower test F1 scores and higher dimensions have the same F1 score. So the best and most parsimonious model appears to be that with dimension = 140.

GloVe

Before fitting word vectors and creating the design matrix, we must select a dimension for word vectors. To do this we ran a for loop to create models with vector dimensions of 1 up to 200. In each loop, we created a model using sentence vectors of that length for 5 iterations. Since the GloVe algorithm is an unsupervised model, each creation of the word vectors will be different. From there, we checked the F1 score given by performance metrics of the testing set and graphed them against dimensionality. From there we were able to determine that a vector length of 150 is the best length for word vectors in this dataset. However, this graph shows a lot of variability indicating that the best dimension relies heavily on the model generated.

Fitting, Evaluations, and Violations

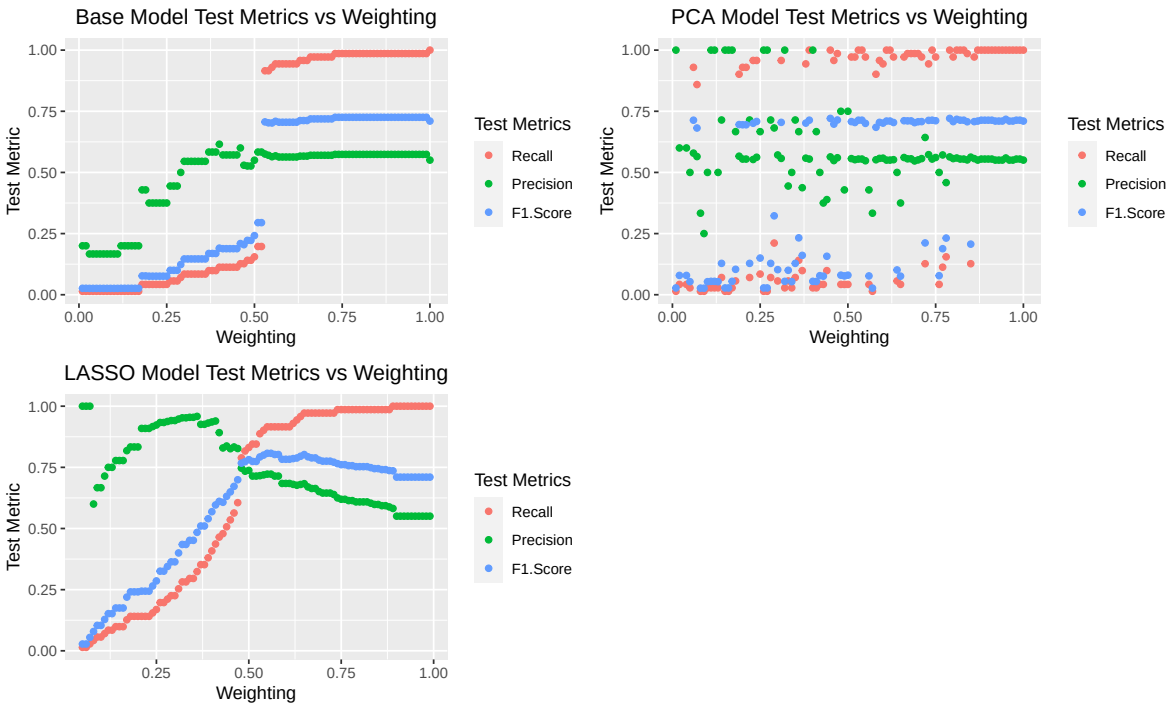
TF-IDF

For the base model we take the minimum AIC model found during variable selection. This result model has a 14% reduction in deviance and an AIC of 734. R fails to estimate several predictor coefficients and outputs them of NA. This suggests investigation of multicollinearity is needed. After examining the model's VIFs and the correlation between predictors it is was found that 6 variables have VIfs that are over 10 and several predictors are perfectly correlated. To deal with multicollinearity PCA and LASSO are explored.

For PCA we kept a cumulative proportion of up to 90% which resulted in using 24 principal components. Fitting our model to the data the AIC was 17135 and the deviance increase by a factor of 24. Clearly the model does not fit the data well. However, no VIFs were found to be over 10 so the multicollinearity was removed.

Next we employed cross validation to fit a LASSO model. An optimal λ of 0.025 was selected and the resulting model produced a 34.55% reduction in deviance.

We know move onto optimal weight selection. We seek to achieve the best balance of Precision, Recall, and F1 Score.



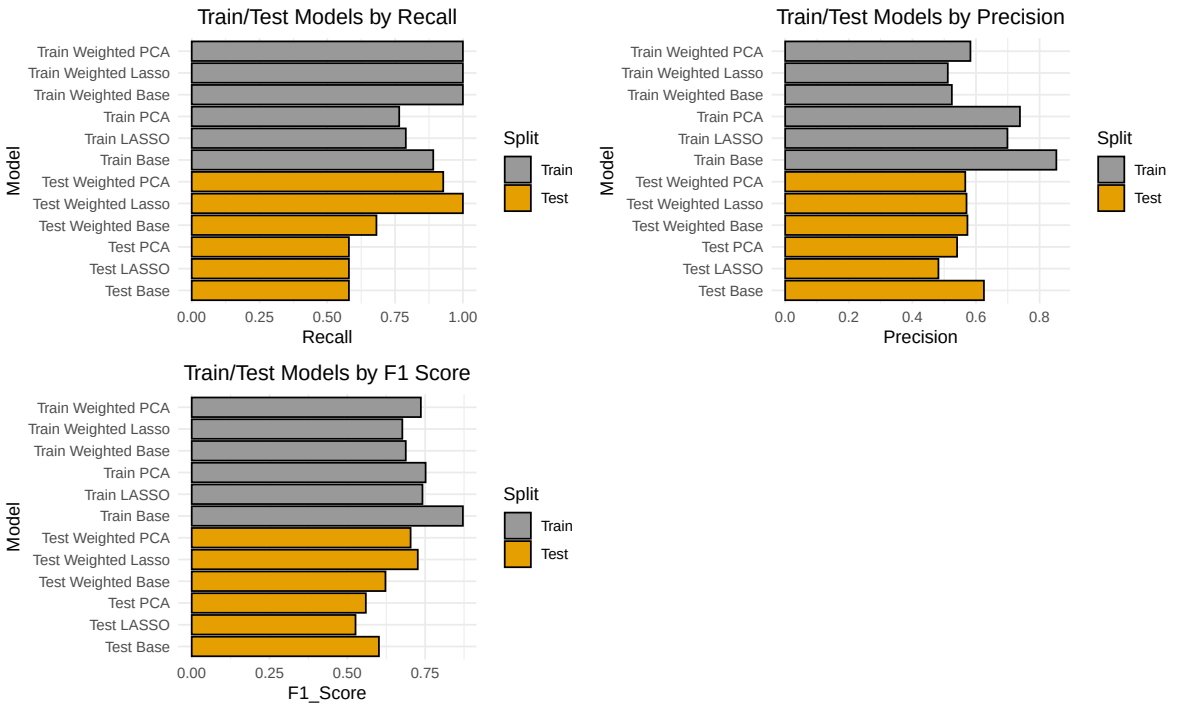
Examining the above graphs the best weightings are 0.75 for the base model, 0.62 for the PCA modek, and 0.55 for the LASSO model.

With model selection finished we may now compare all models and pick the best TFIDF model.

Weighted LASSO is the superior model. While it has lower Recall than Weighted PCA and Base, it does the best in F1 Score which indicates it is the most balanced model. Both Weighted PCA and Weighted base have a poor Precision and F1 Scores

Word2Vec

First, a baseline logistic regression model was fitted from the Word2Vec model using the Skip-Gram architecture and a vector dimension of 140. This baseline model showed a significant percent decrease in deviance over the null model at .%. However, of the 140 variables used to fit the model _ had VIF > 10, indicating serious multicollinearity issues within the model. Next, PCA was used to reduce the dimensionality and multicollinearity of the baseline model. We opted to select the first principal components of the model to be used for logistic regression, which accounted for 90% of the variation in the model. The PCA model showed a lesser



percentage decrease in deviance than the baseline model at .%, but this decrease is still significant. Although, the fit of this model was slightly worse than the baseline, none of the variables used to fit the PCA model had VIF > 10, indicating the multicollinearity issues in the baseline model have been addressed. We opted to use LASSO to as an alternative method to reduce the dimensionality of the model while hopefully retaining or improving the performance of the baseline model. This involved first using 5 fold cross-validation to select an optimal shrinkage parameter (lambda) for LASSO, and then fitting a logistic regression with the optimal lambda found. LASSO resulted in of the variables being shrunk to zero. The fitted LASSO model had a .% decrease in deviance which is significant.

We thought the prediction of the models could be further improved by weighting each of the 3 previously fitted models. The optimal weights will be found for each model by examining the graph of w vs Testing F1 Score, Precision, and Recall and selecting the w that achieves a balance of F1 Score, Precision, and Recall. (Put weight graphs here) Starting with the baseline model, the optimal weights were found to be for the sarcastic class and for the non-sarcastic class by examining Figure xyz. This model has a significant decrease in deviance at .% however it suffered from the same multicollinearity issues of the unweighted baseline model with __ variables having VIF > 10. For the weighted PCA model, the optimal weights were found to be for the sarcastic class and for the non-sarcastic class by examining Figure xyz. The weighted PCA model has a .% decrease in deviance which is (less/more) than the baseline model. Similar to the unweighted PCA model the weighted PCA model had no variables with VIF > 10. For the weighted LASSO model, the same optimal lambda parameter from the unweighted model

was used and optimal weights were found to be for the sarcastic class and for the non-sarcastic class by examining Figure xyz. The weighted LASSO model had a .% decrease in deviance which is significant. The testing and training metrics for all 6 models are presented in Figure xyz. The ___ model preformed the best prediction of sarcastic comments as evidenced by it having the highest testing F1 score. (Put results table here)

GloVe

Once we determined the best length of word vectors for the GloVe model, we fit a base model using the sentence vectors with a vector dimension of 150. This baseline model showed a _ percent decrease in deviance over the null model at _%. However, we found that **of the __ variables had a VIF $>$ 10, indicating multicollinearity issues in the model. Next, we used PCA to configure a new model to eliminate multicollinearity and reduce dimensionality. This model was regressed on the first** principal components, which account for 90% of the variation in the model. We found that this model performed very similarly to the baseline model, however it had a smaller F1 score indicating a lower sarcasm prediction rate. The variables used to fit the PCA model had VIF's $<$ 10, showing that the multicollinearity had been addressed. Next, we chose to create a LASSO model as an alternate method to reduce dimensionality. This is conducted in the same way as in Word2vec using 5 fold cross-validation. This model improved on the baseline model barely with a slightly larger F1 score and recall, but precision decreased indicating there was not a significant improvement in the model.

Since the F1 score of each of these models were all between 0.5 and 0.6 we thought the prediction of the models could be improved by weighing each of the 3 previously fitted models. The optimal weights for each model were found by plotting w_1 against Testing F1 Score, Recall, and Precision. We found that Precision remained relatively constant throughout each model while the F1 Score and Recall increased as w_1 increased. Next, we selected a w_1 value of 0.85 as this is the best weighting where Recall and F1 Score plateaued as the model tended not to improve significantly past that point. Starting with the baseline model we found that there was an improvement over the previous models as F1 Score and Recall are slightly larger. Additionally, there was a significant decrease in the amount of VIF's $>$ 10, indicating a large decrease in the multicollinearity. The weighted PCA model was found to have an optimal w_1 value of 0.88 and performed much better than the previous models with an F1 Score of **. The Recall of this model also increased significantly, indicating that more sarcastic comments were predicted correctly. Similar to the unweighted PCA model, no VIF's were greater than 10. Finally, we created a weighted LASSO model with a w_1 value of 0.8 as this model plateaued to a Recall of 1 much sooner than the rest. This resulted in the highest F1 Score of** , and a Recall of ___. This indicates that the weighted LASSO model performed the best at predicting sarcastic comments as seen by a significantly larger F1 Score. The training and testing metrics for all 6 models are presented in Figure xyz. The graphs from which we selected the best weightings can be found in Figure xyz.

Other Findings

TF-IDF

As weighted LASSO was found to be the best TF-IDF model we have access to the set of words that were not shrunk to 0. This provides us we insight as to which words predict best for sarcasm in the subreddit NHGW. The words are:

tate	written	companionship	reduct	final
remov	report	nowaday	gold	realis
asexu	sister	dump	iron	page
pictur	puberti	gal	act	broke
ignor	crime	cop	polic	pedo
jail	depend			

The results are quite interesting. NHGW is a subreddit about making fun of those who seemingly unaware of why women act the way they do and we see terms related to sexuality, relationships, and crime. Notably we also see *tate* a highly controversial figure for his views on women.

Word2Vec

GloVe

Conclusion

Model Comparison

of all methods which had the best balance of metrics

Limitations

The largest issue with this analysis is the dataset. Logistic regression is not equipped to handle such 0 heavy data and without us artifically cosntructing the ratio of sarcastic to nonsarc comments this analysis likely would not work.

discainer about data set and stocahstic nature of Glove and Word2Vec

Final Remarks

Summary of everything

References

https://en.wikipedia.org/wiki/Reddit#References

https://www.reddit.com/r/dataisbeautiful/comments/9q7meu/most_sarcastic_subreddits_oc/

Text as Data Barry DeVille, Gurpreet Singh Bawa

This paper shows we can use these metrics for this: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9

for definition of recall, precision, and f1 use: Hands-On Ensemble Learning with Python