Supplementary Material: Classifying Hate Speech Using a Two-Layer Model

Yiwen Tang*
Department of Mathematics and Statistics, Wake Forest University
and
Nicole Dalzell
Department of Mathematics and Statistics, Wake Forest University

July 29, 2019

SUPPLEMENTAL MATERIALS

Wikipedia dataset: the data used in Section 3 are available **here** (.csv file)

Twitter dataset: the dataset used in Section 4 are available **Here** (.csv file)

Code: the codes used to run the analyses in Section 3 and 4 are available on GitHub **here** (.rmd file)

R Packages: the R packages used for the applications in Sections 3 and 4 include: caret [5]; dplyr [14]; tm [4]; readr [15]; stringr [13]; text2vec [10]; magrittr [2]; textclean [9]; hunspell [7]; Basic R [8].

*The authors gratefully acknowledge funding from the Wake Forest Summer Research Fellowship and the Starr Travel Grant, both offered by the URECA Center at Wake Forest University.

References

[1] Asad Abdi, Siti Mariyam Shamsuddin, Shafaatunnur Hasan, and Jalil Piran. Machine learning-based multi-documents sentiment-oriented summarization using linguistic treatment. *Expert Systems with Applications*, 109:6685, 2018.

[2] Stefan Milton Bache and Hadley Wickham. *magrittr: A Forward-Pipe Operator for R*, 2014. R package version 1.5.

[3] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[4] Ingo Feinerer and Kurt Hornik. *tm: Text Mining Package*, 2017. R package version 0.7-3.

[5] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan, and Tyler Hunt. *caret: Classification and Regression Training*, 2017. R package version 6.0-78.

[6] H. P. Luhn. Key word-in-context index for technical literature (kwic index). *American Documentation*, 11(4):288–295, 1960.

[7] Jeroen Ooms. *hunspell: High-Performance Stemmer, Tokenizer, and Spell Checker*, 2017. R package version 2.9.

[8] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017.

[9] Tyler W. Rinker. *textclean: Text Cleaning Tools*. Buffalo, New York, 2018. version 0.9.2.

[10] Dmitriy Selivanov and Qing Wang. *text2vec: Modern Text Mining Framework for R*, 2018. R package version 0.5.1.

[11] Julia Silge and David Robinson. tidytext: Text mining and analysis using tidy data principles in r. *JOSS*, 1(3), 2016.

[12] Peng Sun, Lihua Wang, and Qianchen Xia. The keyword extraction of chinese medical web page based on wf-tf-idf algorithm. *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2017.

[13] Hadley Wickham. *stringr: Simple, Consistent Wrappers for Common String Operations*, 2017. R package version 1.2.0.

[14] Hadley Wickham, Romain Franois, Lionel Henry, and Kirill Mller. *dplyr: A Grammar of Data Manipulation*, 2018. R package version 0.7.5.

[15] Hadley Wickham, Jim Hester, and Romain Francois. *readr: Read Rectangular Text Data*, 2017. R package version 1.1.1.

A Appendix: L2CL Features

A.1 Feature Engineering for Layer 2: Use the Scores (cont.)

In this section, we show the process of constructing (1) the P1 Feature, (2) the P2 Feature, and (3) the P3 Feature for each comment under every sub-category. For every sub-category j, the four scores (the SCS and P1 - P3 scores) are used to generate four features for each comment $c = 1, \ldots, n_{target}$. The four features are called the Sub-Category Feature (SCF), P1 Feature, P2 Feature, and P3 Feature.

Suppose we are constructing the scores for comment c for sub-category j. We first consider the words w_{ij} in L2CL_j, where $i = 1, \ldots, K = 100$. Let $\alpha^{cj} = (\alpha_1^{cj}, \ldots, \alpha_{100}^{cj})$ be a vector of length 100 indicating if each word w_{ij} in L2CL_j is in comment c. For instance, if comment c contains the second word in L2CL_J, $\alpha_2^{cj} = 1$; otherwise, $\alpha_2^{cj} = 0$. To create the SCF for comment c for sub-category j, we compute,

$$SCF_{cj} = \sum_{i=1}^{K} \left(s_{i_j} \times \alpha_i^{cj} \right). \tag{1}$$

In other words, the SCF is the sum of all s_{i_j} such that word w_{ij} is in comment c. Similarly, to create the P1-P3 Features for comment c for sub-category j, we compute,

$$P1F_{cj} = \sum_{i=1}^{K} \left(P1_{i_j} \times \alpha_i^{cj} \right), \tag{2}$$

$$P2F_{cj} = \sum_{i=1}^{K} \left(P2_{i_j} \times \alpha_i^{cj} \right), \tag{3}$$

$$P3F_{cj} = \sum_{i=1}^{K} \left(P3_{i_j} \times \alpha_i^{cj} \right). \tag{4}$$

The $P1_{i_j}$, $P2_{i_j}$, and $P3_{i_j}$ are defined in Section 2.3.2 and Section 2.3.3.

B Appendix: Feature Engineering for Layer 1

In contrast to classification situations with numeric or categorical features, data for sentiment analysis often contains text data rather than distinct features. We use our text data to create 15 features to use to classify comments in Layer 1. These features are:

1. **The Number of Characters (*sentence_length*)**: This feature reveals the length of each comment. Negative comments tend to be shorter than the neutral ones.

2. **The Number of Capital Letters (*num_cap*)**: We chose to include this feature because we believe capital letters tend to convey stronger emotional messages. Negative comments tend to have a large number of capital letters than neutral ones.

3. **The Proportion of Capital Letters (*cap_density*)** ((2) divided by (1)): Even though internet users may not strictly follow the standard rules of grammar, many of them tend to begin every sentence with a capital letter. Thus, the number of capital letters is positively correlated (0.46) with the length of the comments. The density of capitalized letters in negative comments is greater than that in neutral comments.

4. **The Number of Words (*num_word*)**: Similar to feature (1), this feature reveals the length of the comments. However, since we do not expect words to have the same number of characters, the number of words may be a good indicator of the number of messages in the comments. Based on the box plot, negative comments tend to have less word than the neutral ones do.

5. **The Number of Question Marks (*num_question*)**: The number of question marks in a comment tells us the possible number of questions in the comment. It may not give us the exact number because one can put more than one question mark after the end of a question. For instance, the first sentence in comment *1dcbc678cd6966bc* is "Where are the evidence???". Additionally, this feature conveys the intensity of emotion for some comments.

6. **The Number of Exclamation Points (*num_excla*)**: Similar to feature (5), when one is angry, upset or offended, one may use exclamation points to express the feelings.

7. **The Number of Periods (*num_period*)**: This feature is included because it serves as a representation of the number of sentences in a comment.

8. **The Number of Stop Words (*num_stopword*)**: Hans Luhn introduces the idea of stop list in his presentation in 1959 [6]. A stop list is a list of common words which contain minimal information, such as and and also. Many past researchers chose to remove these in the preprocessing stage [1]. However, we choose to keep the stop words in order to maintain the integrity of the comments. Words which are stop words in general speech may, in online comments, contain information helpful for classification. We extract the stop list from the R package tidytext [11] and count the amount the stop words in each comment. In the training data, neutral comments tend to have more stop words than negative ones.

9. **The Stop Word Ratio (*stopword_ratio*)** ((8) Divided By (4)): Similar to feature (3), we are interested in the ratio because it takes the number of words in the comment into account. The proportion of stop words in the comments allow us to conduct cross-comparison.

10. **The Number of Top 20 Negative Words by Frequency (*num_neg_word*)**: To build this feature, we subset our training data to only the negative comments, and identified the top 20 most common words. We then counted the number of these top 20 negative words that appeared in each comment.

11. **The Negative Word Ratio (*neg_word_ratio*)** ((10) Divided By (4)): We included this feature due to a reason similar to the reason we added the feature (9).

12. **The Number of Unique Words(*num_unique_word*)**: Feature (12) was needed because, in the original dataset, many users copied and pasted a single sentence multiple times to strengthen the intensity of the comments. Therefore, a comment could be 600 words long, but contain only the phrase I hate that! repeated over and over. Feature 12 reduced such comments to the unique words contained within the comment.

13. **The Unique Word Ratio (*unique_word_ratio*)**: We included this feature due to a reason similar to the reason we added the feature (9).

14. **The Number of Top 20 Negative Words by TF-IDF (*num_neq_word_tfidf*)**: This feature is very similar to feature (10), but in contrast, we chose the top 20 words by TF-IDF(Term Frequency-Inverse Document Frequency), a popular term-weighting scheme used in many lexicon-based sentiment analysis studies [12]. However, after examining the top 20 TF-IDF negative words, we found that 11 out of 20 words were actually the derivative forms (e.g. incorrectly spelled, truncated, past tense or third person singular) of some high-frequency words. Because of the typos and other data quality concerns, TF-IDF proved to be an ineffective metric for our dataset.

15. **The TF-IDF Negative Word Ratio (*num_neg_tfidf_ratio*)** ((14) Divided By (4)): We included this feature due to a reason similar to the reason we added the feature (9).

The distributions of the features are presented in Table 3.

C Appendix: Wikipedia Data Layer 1 Training

Using the features described in Appendix Section B, we used a variety of machine learning algorithms to classify comments in the training data as *Negative* or *Neutral*. We used the *caret* package in R to run these models and compare results. For each method, we conducted 5-fold Cross-Validation and repeated this process there times. Table 4 displays the distributions of accuracies of each algorithm.

XGBoost Tree stands for "Extreme Gradient Boosting Tree", and it is a popular machine learning algorithm that is motivated by the concept of gradient boosting, similar to other boosted tree models. However, XGBoost Tree adopts a more regularized model to reduce over-fitting, and therefore it tends to preform better than other tree models. For more details, please refer to [3].

According to Table 5, using the XGBoost Tree, 83.1% of the *Neutral* comments and 68.2% of *Negative* comments are correctly captured in the test dataset. Table 6 displays

the 95% confidence interval for the accuracy of the Xgboost approach. It also shows a naive approach which classifies all comments as neutral and yields an accuracy of 50%.

We display the distributions of the accuracies of the Random Forest models in Table 7. The importance of the final scores in these six binary classification model is displayed in six diagrams, starting from Figure 2.

Random Forest is another widely-used machine learning model, and relies on using bootstrapping to re-sample from the original sample. On each bootstrapped sample, a classification tree is grown. Predictions are then made either by a voting system, or other technique of combining the multiple predictions to obtain a single result. Since multiple trees are used for prediction, it tends to be more robust than a single tree model to small changes or outliers in the data. For more information, please refer to here.

D Appendix: Tables

Table 1: The Counts and Percentages of the Negative and Neutral Comments

Classification	Neutral	Negative
Proportion	.898	.102
Count	143346	16225

Table 2: Summary Statistics of the comments.

Features	Min	1st Qu.	Median	3rd Qu.	Max.
Character Length	5	96	205	434.5	5000
Number of Capital Letters	0	3	7	15	4717
Proportion of Capital Letters	0	0.02	0.03	0.05	0.996
Number of Words	1	17	37	77	1403
Number of Question Marks	0	0	0	1	209
Number of Exclamation Marks	0	0	0	0	4942
Number of Periods	0	1	3	5	682

Table 3: Distributions of the 15 features.

Features	Min	1st Qu.	Median	3rd Qu.	Max
Number of Characters	8	76	168	370	6247
Number of Capital Letters	0	3	6	15	4717
Proportion of Capital Letters	0	0.02	0.03	0.05	0.996
Number of Words	1	14	30	66	1403
Number of Question Marks	0	0	0	1	209
Number of Exclamation Points	0	0	0	0	4942
Number of Periods	0	1	2	5	682
Number of Stop Words	0	6	14	26	175
Stop Word Ratio	0	0.33	0.41	0.5	0.91
Number of Top 20 Negative Words by Freq.	0	0	1	2	15
Negative Word Ratio	0	0	0.01	0.04	0.8
Number of Unique Words	1	12	25	47	447
Unique Word Ratio	0.0008	0.71	0.81	0.91	1
Number of Top 20 Negative Words by TF-IDF	0	0	0	0	6
TF-IDF Negative Word Ratio	0	0	0	0	0.67

Table 4: The Distributions of the performances of the 10 methods on the training data. MMLPNN stands for Monotone Multi-Layer Perception Neural Network, NNFE stands for Neural Networds with Feature Extraction, and GLRLER standard for Generalized Linear Regression with Lasso and Elastinet Regularization.

Method	Min	1st Qu.	Median	Mean	3rd Qu.	Max
Classification Tree	0.657	0.671	0.683	0.679	0.69	0.7
Logistic Regression	0.666	0.699	0.704	0.702	0.708	0.716
Random Forest	0.734	0.737	0.743	0.742	0.746	0.749
XGBoost Tree	0.743	0.752	0.754	0.755	0.76	0.762
MMLPNN	0.747	0.754	0.755	0.756	0.758	0.769
NNFE	0.722	0.729	0.732	0.735	0.74	0.745
Naive Bayes	0.591	0.594	0.599	0.598	0.602	0.603
Boosted Classification Tree	0.703	0.706	0.707	0.708	0.709	0.722
GLRLER	0.692	0.697	0.703	0.701	0.704	0.706
Gradient Boosting Machine	0.706	0.714	0.722	0.721	0.727	0.731

Table 5: The Confusion Matrix of the XGBoost Classification Tree.

(By Count)	True Neutral	True Negative
Predicted Neutral	831	318
Predicted Negative	169	682

Table 6: The Prediction Accuracy of the Xgboost Classification Tree in the Wikipedia Training Data.

Accuracy	0.7565
95% Confidence Interval	(0.7371,0.7752)
No Information Rate(NIR)	0.5

Table 7: The Distributions of the Accuracies of the Random Forest Models in the Wikipedia Training Dataset. We conduct 5-fold cross validation for 3 times to see the general performance of the models. The five number summaries below indicate the results across these repetitions.

Random Forest Classification Models	Min.	1st Qu.	Median	3rd Qu.	Max.
Toxic	0.9396	0.9406	0.9409	0.9409	0.9415
Severe Toxic	0.9015	0.9034	0.9061	0.9079	0.91
Obscene	0.7549	0.7617	0.7662	0.7708	0.7849
Threat	0.9705	0.9719	0.9728	0.9734	0.9747
Insult	0.6562	0.6642	0.6682	0.6705	0.6808
Identity-Hate	0.913	0.914	0.9146	0.9151	0.9159

Table 8: The Confusion Matrix of the XGBoost Classification Tree in the Twitter Training Dataset.

(By Count)	Actually Positive	Actually Negative
Predicted Positive	160	43
Predicted Negative	40	157

Table 9: The Performances of the Random Forest Models on the Test Wikipedia Dataset.

RFs on Test Dataset	Accuracy	Difference
Toxic	0.729	
Toxic w/ L2CL Based Features	0.775	0.046
Severe Toxic	0.934	
Severe Toxic w/ L2CL Based Features	0.950	0.016
Obscene	0.813	
Obscene w/ L2CL Based Features	0.890	0.077
Threat	0.985	
Threat w/ L2CL Based Features	0.9855	0.0005
Insult	0.811	
Insult w/ L2CL Based Features	0.840	0.029
Identity Hate	0.951	
Identity Hate w/ L2CL Based Features	0.954	0.003

Table 10: The Distribution of the Negative Reasons. The proportion column indicates the proportion of the negative comments that fall into each sub-category.

Reason	Count	Proportion
Customer Service	2910	.32
Late Flight	1665	.18
Cancelled Flight	847	.09
Lost Luggage	724	.08
Bad Flight	580	.06
Flight Booking	529	.06
Flight Attendant	481	.05
Long Lines	178	.02
Damaged Luggage	74	.01
Can't Tell	1190	.13

E Appendix: Figures

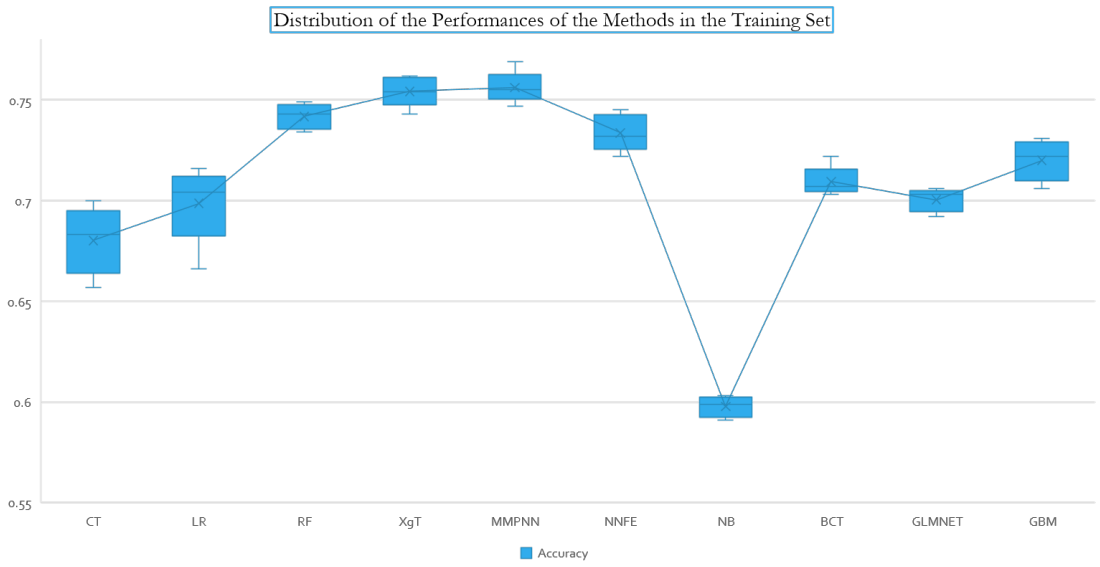

Figure 1: Distribution of Accuracy rates on the training sets across different methods.

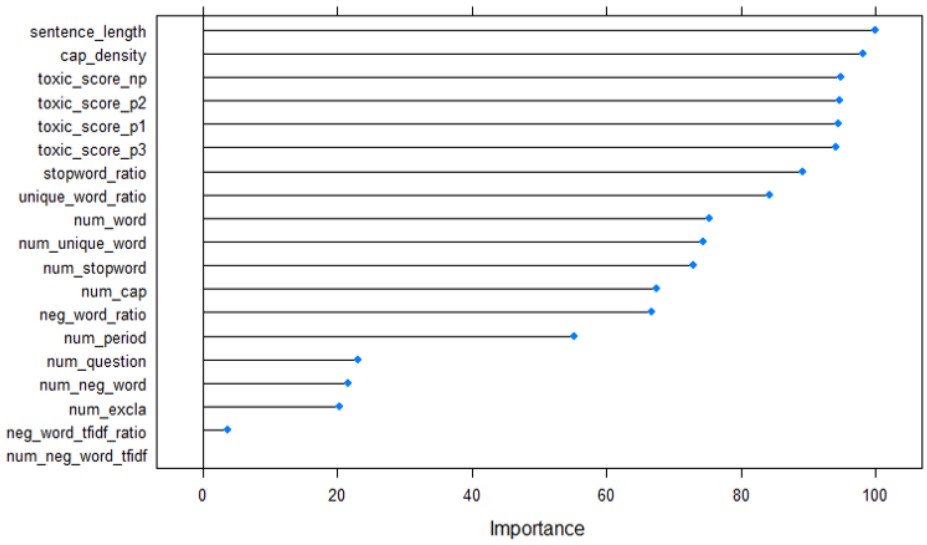

Figure 2: Variable importance plot for trained **Toxic** model. Here, *toxic_score_p*1 is the P1 Feature, *toxic_score_p*2 is the P2 Feature, *toxic_score_p*3 is the P3 Feature, and *toxic_score_np* is the SCF.

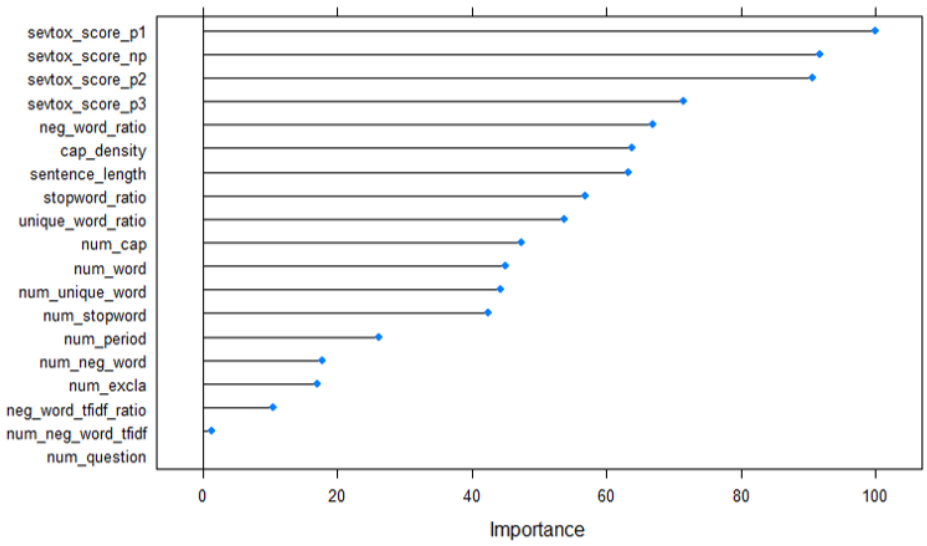

Figure 3: Variable importance plot for trained **Several Toxic** model. Here, *sevtox_score_p*1 is the P1 Feature, *sevtox_score_p*2 is the P2 Feature, *sevtox_score_p*3 is the P3 Feature, and *sevtox_score_np* is the SCF.

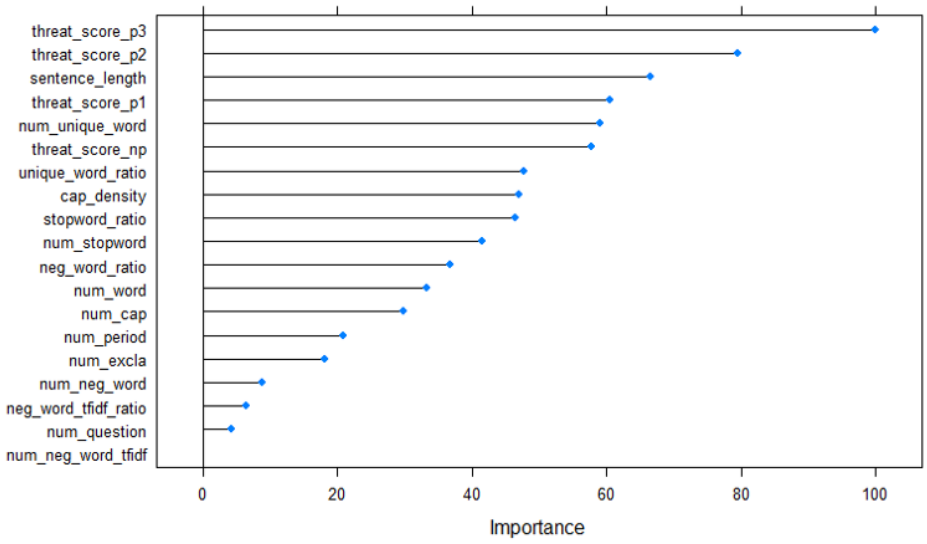

Figure 4: Variable importance plot for trained **Threat** model. Here, *threat_score_p*1 is the P1 Feature, *threat_score_p*2 is the P2 Feature, *threat_score_p*3 is the P3 Feature, and *threat_score_np* is the SCF.

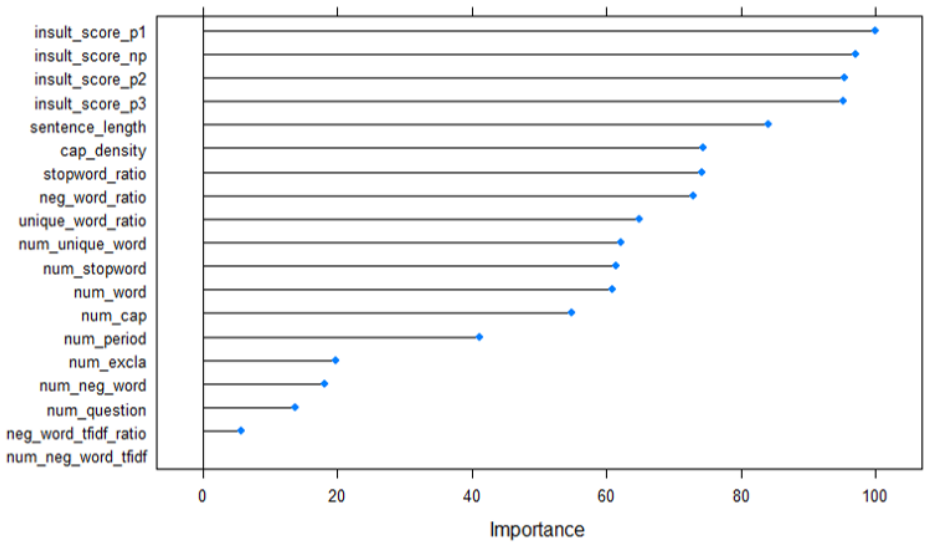

Figure 5: Variable importance plot for trained **Insult** model. Here, *insult_score_p*1 is the P1 Feature, *insult_score_p*2 is the P2 Feature, *insult_score_p*3 is the P3 Feature, and *insult_score_np* is the SCF.

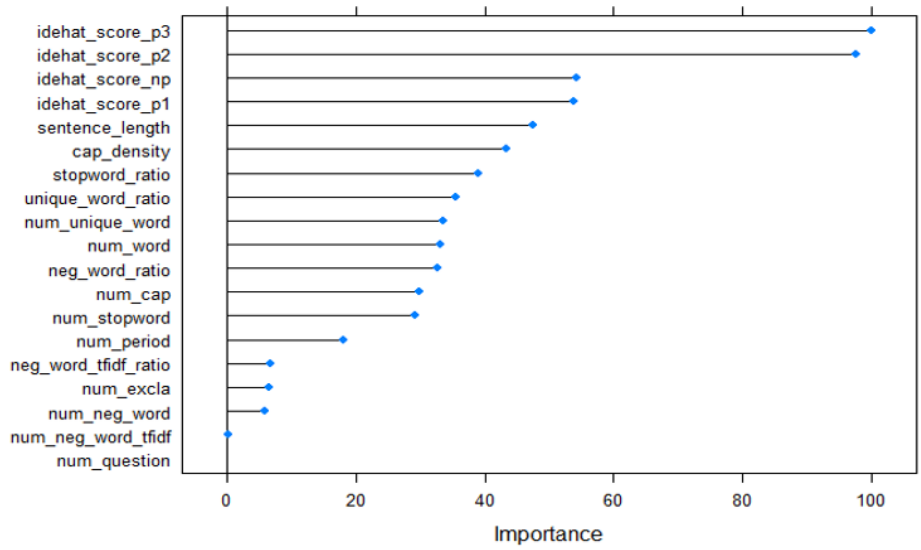

Figure 6: Variable importance plot for trained **Identity Hate** model. Here, *idehat_score_p*1 is the P1 Feature, *idehat_score_p*2 is the P2 Feature, *idehat_score_p*3 is the P3 Feature, and *idehat_score_np* is the SCF.

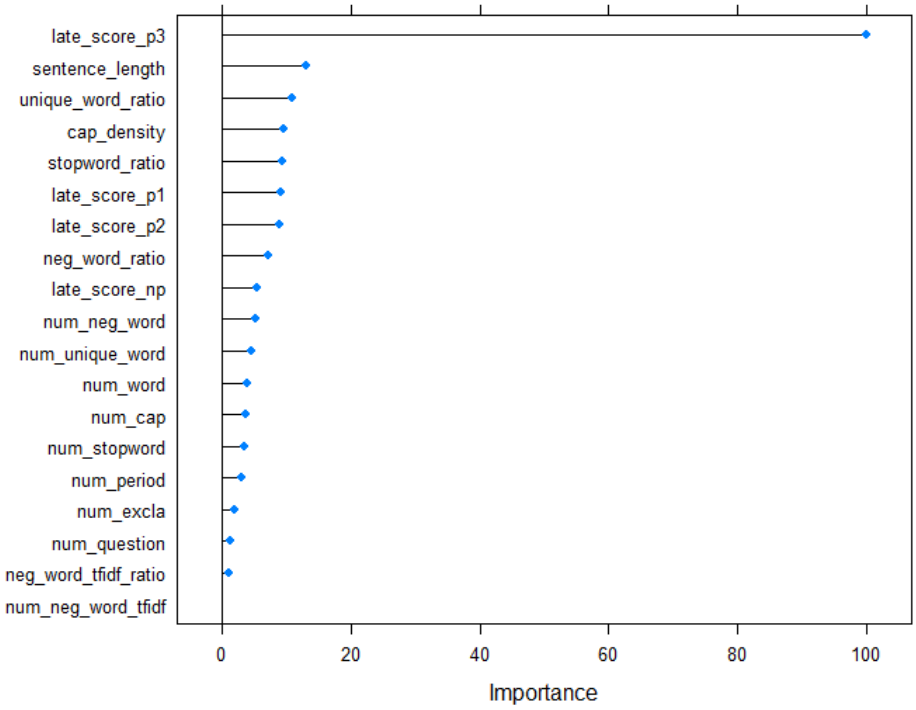

Figure 7: Variable importance plot for trained **Late Flight** model. Here, *late_score_p*1 is the P1 Feature, *late_score_p*2 is the P2 Feature, *late_score_p*3 is the P3 Feature, and *late_score_np* is the SCF.

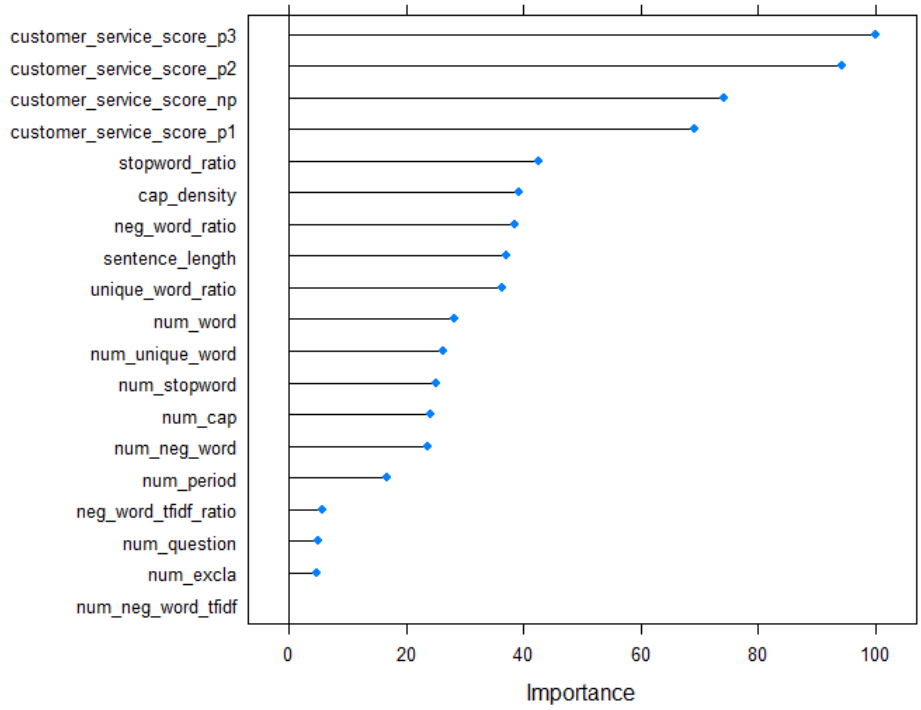

Figure 8: Variable importance plot for trained **Customer Service** model. Here, *customer_service_score_p*1 is the P1 Feature, *customer_service_score_p*2 is the P2 Feature, *customer_service_score_p*3 is the P3 Feature, and *customer_service_score_np* is the SCF.

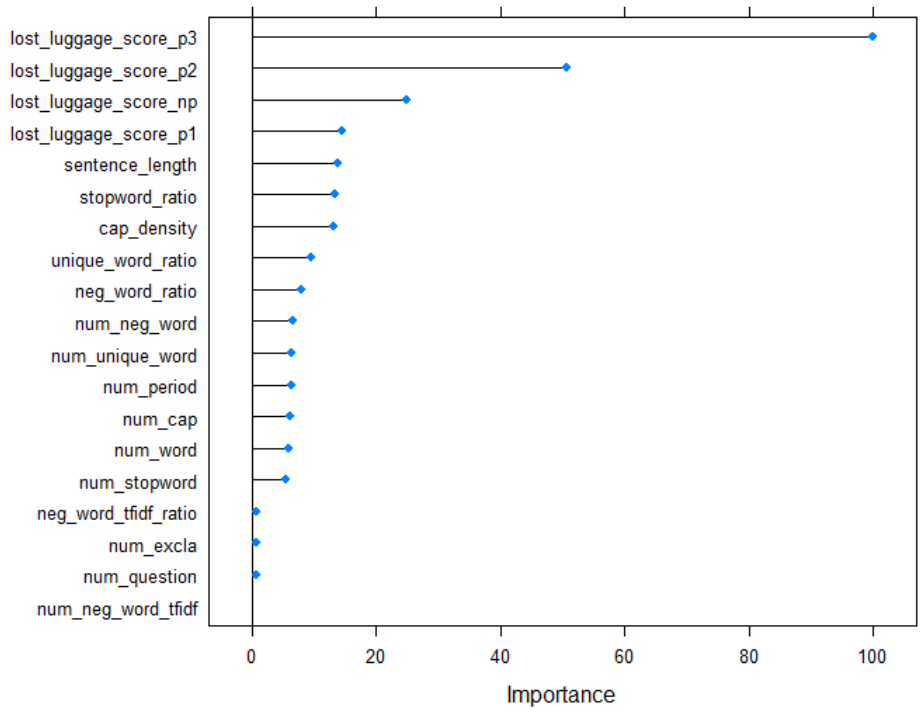

Figure 9: Variable importance plot for trained **Lost Luggage** model. Here, *lost_luggage_score_p*1 is the P1 Feature, *lost_luggage_score_p*2 is the P2 Feature, *lost_luggage_score_p*3 is the P3 Feature, and *lost_luggage_score_np* is the SCF.