On Building a Response Prioritization Engine for Southwest Air

Debashri Mukherjee, Kathryn Tooker, and Tyreke White

Abstract—For companies, being able to quickly prioritize and respond to tweets is highly valuable. Machine learning can augment this process by building powerful classification models which accurately detect the sentiment of a tweet (to better inform company representatives) while also making it possible to tell the customer what the estimated response time from a service agent will be. This paper explores a use case of Southwest Air Twitter data, where sentiment analysis is carried out on tweets sent to the official Southwest Air Twitter account. Sentiment classification and response time prediction models are created using tweet information. However, the sparsity in our Twitter dataset makes response time prediction a substantial endeavor.

1 Introduction

No matter the size of a company, one thing is certain: customer service is a top priority for any business, and promptly addressing the concerns of disgruntled customers should be streamlined to maximize satisfaction across all parties.

Airlines in the US have come under much scrutiny recently, and social media has enabled customers to both voice their grievances and share their praise directly with airlines in real time. On a daily basis, airlines must filter through and reply to high volumes of tweets/posts to maintain customer loyalty and a positive brand image. This is a daunting task, yes, but it can be significantly automated with machine learning.

Our project will address this need in modeling a specific use-case for Southwest Airlines, by applying sentiment analysis to tweets that are directed towards Southwest Air (e.g. "@" mentions) as well as modeling and predicting how long it will take a customer service agent to respond to the tweet in order to inform response prioritization. Such a tool might assist Southwest Air in training its representatives to ensure that every tweet is being answered at the *right* time in the *right* order.

Our project contains two components. The first piece involves predicting whether a tweet expresses negative or positive sentiment and exploring which models (Naive Bayes, Support Vector Machines or Neural Networks) do best at correctly classifying a tweet as positive or negative. The second portion of our project then explores prediction of a SW agent's estimated response time based upon sentiment. We build a widget - included in the Appendix - wherein a user types a tweet (the input) and the output returned is the estimated time to first response that a customer should expect based on a regression model. We experimented with predictions based on Lasso Regularization, Ridge Regression and Partial Least Squares, ultimately choosing Ridge Regression for the widget as it was the highest performing model that did not present major implementation challenges in the widget.

2 Literature Review

The application of sentiment analysis to Twitter information is not a new exercise. In fact, many studies have extended this application in novel ways. Lacking in the literature, however, is a thorough investigation of response time predictions.

To perform supervised machine learning on Twitter data, which is unlabeled in its raw form, previous authors have used systematic data collection methods to label tweets as positive or negative before building models. Pak and Paroubek searched Twitter for tweets containing positive and negative emoticons and labeled the tweets accordingly [1]. Similarly, Kouloumpis et al. (2011) queried Twitter for frequently used hashtags that are known to have an associated positive or negative sentiment [2]. In both papers, the emoticons or hashtags serve as proxies for labels, and then the models learn to predict the labels using the words in the tweets. Although these methods make labeling simple and easy, they have shortcomings. By only collecting tweets that have emoticons or polarized hashtags, the authors are training their models on only "extreme" sentiments, making prediction challenging for tweets with less extreme sentiment. While we ultimately faced this challenge as well, we attempted to avoid it by using VADER to label our tweets (see Section 3.2).

Naive Bayes, Support Vector Machines (SVMs), and Neural Networks (NNs) are common text classification techniques employed in existing literature. Go et al. (2009) implemented Naive Bayes and SVMs on labeled Twitter data and achieved accuracy values of 81% and 82%, respectively [3]. Similarly, Wang and Manning (2012) find that SVM slightly outperforms Multinomial Naive Bayes on sentiment classification of short text snippets [4]. Additionally, Liu et al. (2013) find that Naive Bayes is scalable to datasets containing millions of movie reviews with accuracy over 80% maintained [5]. Lastly, Borele and Borikar (2016) propose artificial neural networks as a strong alternative to traditional classification techniques [6]. We

TABLE 1
Description of Tweet Attribute Data

Attribute Name	Description
id	Specifies the unique ID of the tweet.
created_at	The timestamp at which a tweet was published.
in_reply_to_status_id_str	The ID of an existing tweet that the update is in reply to.
text	The string text of the tweet.

thus explored these three methods to compare our findings with that in the literature. Important to note, however, is the fact that our experiment used different labeling techniques.

On the issue of predicting response times to tweets, there is limited literature. Mahumad et. al (2013) estimate a given user's response time based on an exponential distribution wait time model. They also explore the effects of the day of the week and the time of the day [7] on tweet response times. There is still much to be done on response time prediction, however.

3 Dataset and Features

We sought to model tweet sentiment and first response times, but our data had to be gathered and converted into a standardized form before we could conduct experiments.

3.1 Data Collection

We used the Python 2.7 *Tweepy* package to connect daily to Twitter's API and download the 10,000 most recent tweets (the daily maximum allowable) containing the mention "@SouthwestAir". The tweets span the time-frame between November 12 and December 11, 2017. Additionally, we downloaded all tweets sent from the official Southwest Air Twitter account in the same time-frame. This resulted in 2 separate datasets: one dataset containing all customer tweets and comments and another dataset containing all of Southwest Air's Twitter feed history. We extracted tweets using **four** attributes included within *Tweepy* shown in Table 1.

Note that while we could have extracted many more features per tweet, this iteration of the project only required these four attributes in subsequent data manipulation and analysis.

3.2 Data Preprocessing

Before we conducted sentiment analysis or response time prediction, we had to merge Southwest Air's Twitter feed data with customer mentions so that we could have one combined dataset containing: a customer's tweet "at" Southwest Air, Southwest Air's response to that customer's tweet, and the timestamps associated with the set of tweets. This merge was carried out by converting each dataset into a Pandas dataframe and matching Southwest Air's "in_reply_to_status_id_str" attribute with the customers' tweet "ID" attribute (these two attributes are equal if one tweet is in reply to another). Note that since we were only interested in times to *first* response, we removed all other matched tweets, even tweets that were continuations of conversations between SW agents and customers. This matching process reduced our dataset from a size of over 50,000 tweets to approximately 1,000 tweets.

Once we created the combined dataset, we needed to calculate the time to first response and assign each customer's tweet a sentiment label. The time to first response was calculated by subtracting the time at which a customer tweeted to SW and the time at which a SW agent first responded to that particular customer. For the purposes of our analysis, we only considered response time in units of minutes. This also made sense given that there were minimal cases (< 5) where agent responses took longer than 1 hour.

Sentiment labeling was carried out with the Python package *VADER*, which stands for Valence Aware Dictionary and sEntiment Reasoner. VADER classification is valence-based, meaning it assigns sentiments on a scale to capture sentiment intensity. VADER contains a dictionary of strings and their associated sentiment ratings, and computes a compound sentiment score for a string of text by summing all sentiment scores in the text, normalized to a range between -1 and 1.

$$normalized = \frac{score}{\sqrt{score^2 + \alpha}} \quad (1)$$

Compound scores closer to -1 indicate more negative sentiment, and scores closer to 1 indicate more positive sentiment. VADER performs particularly well on social media text, as it captures sentiments associated with punctuation, emoticons, abbreviations, and Internet slang. This makes it a strong fit for our application, unlike other methods involving word-only sentiment dictionaries [8].

After obtaining the compound scores for each tweet in our dataset, we experimented with different labeling schemes by segmenting the scores in different ways. We tested our models on five labels, three labels, and two labels (binary) and determined that binary classification performed the strongest due to a high volume of tweets classified in the neutral range. For binary classification, we considered a compound score greater than 0.5 to indicate positive sentiment and a score less than -0.5 to indicate negative sentiment, removing all neutral tweets in between as VADER did not classify them as accurately.

3.3 Feature Space

Initially, we classified the 1,000 customer tweets into 5 sentiment labels ranging from 0 to 5, with 0 representing extremely angry tweets and a label of 5 representing a thoroughly pleased customer. This initial classification scheme caused our subsequent methods to perform very poorly (most likely due to the size of our dataset), so we settled on two labels: 0 (negative tweets) and 1 (positive tweets).

For the tweets remaining after we carried out the above processing, we created a sparse "bag-of-words" matrix, with

each entry *ij* representing the occurrence of a word *j* in tweet *i*. Each entry of the matrix was further transformed using a TF-IDF vectorizer - common stop words were completely removed, and frequently occurring expressions like "SouthwestAir," "flying," etc. were appropriately weighted across all tweets. The size of the feature matrix used for the rest of this analysis is 329 x 174.

4 METHODS

In this section, we discuss the supervised machine learning methods we implemented to build, train and test predictive sentiment classifier models on the tweet data. We also discuss the regression models used to predict SW agent first response times.

4.1 Sentiment Classification

Sentiment Classification can be modeled by many different methods, but we explored the three most aligned with existing literature: Naive Bayes Classifiers, Support Vector Machines, and Neural Network Classifiers.

4.1.1 Naive Bayes Classifier

Naive Bayes classifiers, a family of classifiers that are based on the popular Bayes probability theorem, are known for being relatively easy to implement and robust baseline models [9], especially in the fields of document classification. Abstractly, Naive Bayes is a conditional probability model where,

$$p(C_k|x) = \frac{p(C_k) \cdot p(x|C_k)}{p(x)} \tag{2}$$

Here, C_k refers to all the possible outcome classes (in our case, these are only two - either 0 or 1). We make the strong assumption that all the features x_i are conditionally independent of every other feature x_j. Naive Bayes Classifiers typically work well on text classification problems for many reasons, but most importantly because of the high-dimensional and noisy nature of text data ($n<<p$). The naive assumption of the independence of the predictors means that the covariance matrix of our model only has non-zero entries on the diagonal. This independence assumption leads to a highly interpretable model. [10]

4.1.2 Support Vector Machines

Support Vector Machines are discriminative classifiers that construct hyperplanes in n-dimensional space. The SVM classifier is equivalent to minimizing the following expression:

$$[\frac{1}{n}\sum_{i}^{n}\max(0, 1 - y_i(w \cdot x_i - b))] + \lambda \parallel w \parallel^2 \tag{3}$$

This made them a suitable candidate to build our text classification model. We chose to use a Support Vector Classifier with a linear kernel. The linear kernel was mostly chosen for 2 reasons: most text classification datasets are linearly separable [11] and we have a high-dimensional input space. Because SVMs are able to perform robustly in high-dimensional spaces, they eliminate the need for feature selection and have been shown to outperform other classification methods [12].

4.1.3 Neural Networks

We primarily cite [13] in our application of Neural Networks to this sentiment classification problem. As found by Zagloul et al. (2009), the performance of Neural Networks is statistically comparable to that of the SVM classifier even when a significantly reduced document size is used. We use a ReLU (REctified Linear Unit) activation function for our hidden layer of neurons.

$$g(z) = max(0, z) \tag{4}$$

The working of neural networks has been shown to peak in high-dimensional settings with low sample size data - a condition which is ideal for our problem. [14]. With a performance that's comparable, if not better in some cases, to Support Vector Machines, Neural Networks are an ideal tool to model our problem of sentiment analysis.

4.2 Response Time Prediction

For response time prediction, we explored Ordinary Least Squares Regression (OLS) in addition to three other models: Lasso Regularization, Ridge Regression, and Partial Least Squares Regression (PLS).

4.2.1 Lasso Regularization

Recall that Lasso is a penalized form of an ordinary least squares regression, and it seeks to minimize the penalty function:

$$\frac{1}{2}||Y - X\theta||_2^2 + \lambda||\theta||_1 \tag{5}$$

The addition of the *L1* term makes the penalty function non-smooth and encourages sparsity in the weight coefficient vector, θ, chosen to minimize the penalty. This sparsity serves as a feature selection tool which can deal with high multicollinearity in data. An increase in the regularization strength, λ, can lead to an increase in the sparsity of the weight vector.

4.2.2 Ridge Regression

Another popular technique used to regularize a linear predictor model is Ridge regression, also known as *L2* regularization. A Ridge regression models seeks to minimize the penalty function:

$$\frac{1}{2}||Y - X\theta||_2^2 + \frac{\lambda}{2}||\theta||_2^2 \tag{6}$$

Whereas Lasso encourages sparsity in the weights vector, θ, Ridge regression reduces model complexity by penalizing large individual weights. Ridge regression is typically another technique used when data has high multicollinearity and traditionally yields a lower mean squared error (MSE) than a standard least squares estimator [15].

4.2.3 Partial Least Squares (PLS)

As mentioned previously, standard regression techniques fail when there is high multicollinearity within the feature space. This is certainly an issue with Twitter data, as negative and positive tweets are limited to a small vocabulary space (people typically use a finite number of different types of words when angry or happy). PLS is a technique that combines both Multivariate Linear

Regression (MLR) and Principal Component Regression (PCR) that attempts to remedy this. Given a feature matrix, X, and a dependent variable, y, PLS maximizes the covariance between X and y to capture maximum variance in X (PCA) while simultaneously capturing maximum correlation in X and y (MLR) [16].

5 Experiments & Results

For both Naive Bayes and Support Vector Machines, we relied on the default parameter settings included within SciKit-Learn [17]. As noted previously, a linear kernel was chosen for our SVM classifier.

We conducted a grid search over the parameter space of the Neural Network classifier to find the parameters that yielded the best cross validation score. Based on these analyses, we determined that the default hidden layer size of 100 yielded the best results. We used default parameters for all other hyperparameters.

The results displaying variation in accuracy with an increase in cross validation fold size are shown in Figure 1. Naive Bayes' performance seems to plateau at a fold size of

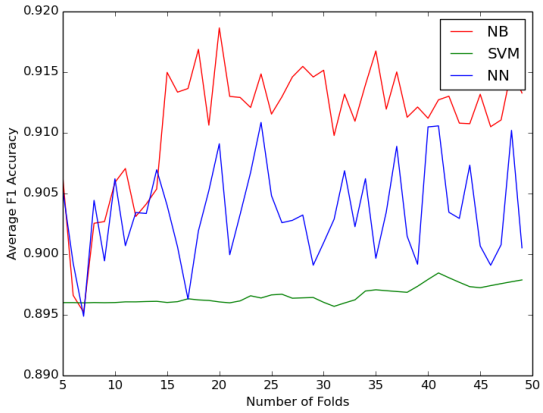

Fig. 1. Accuracy results for classifiers vs. Cross Validation Fold Size.

15. The Neural Network accuracy indiscernibly fluctuates but remains between 0.895 and 0.91, and the accuracy of the SVM classifier only shows signs of improvement at a fold size of 40 and greater. We decided to strike a balance between computational speed of our subsequent models and satisfactory accuracy. We thus settled for a fold size of 15.

To measure the performance of our sentiment classification models, we used the F1 score. The F1 score is defined as the harmonic mean of *Precision* and *Recall*, and it provides a more realistic, robust measure of a classifier's performance:

$$2 \cdot \frac{recall \cdot precision}{recall + precision} \tag{7}$$

Given the nature of our Twitter data, we suspected the "bag-of-words" feature matrix to possess a high degree of multicollinearity. To test this, we first examined the entries of its Covariance Matrix, Σ, but found most of them to be very close to 0 (most likely attributed to the original feature matrix's sparsity). However, in a separate step, we conducted a Principal Component Analysis, whose results in Figure 2 suggest some multicollinearity. Given the sparsity in the feature matrix, we were not surprised by the results, but it presents interesting computational challenges going forward with this project.

To address the possible multicollinearity within our data when predicting response times, we implemented Lasso Regularization. In subsequent steps, we applied Ridge Regression and Partial Least Squares which are other techniques commonly used to increase the predictive power of sparse data such as ours. Because Scikit-Learn v0.19.1 only contains an r^2 scoring metric for these models, we use this metric to compare response time prediction performance across the three models. The results of the regression analyses are presented in Table 2. Notice that Lasso has the least amount of predictive power (the lowest r^2 value) while PLS has the greatest amount. PLS's predictive power is better than Ridge by more than 40%, and it is better than Lasso by more than 50%.

TABLE 2
Performance of Models on predicting response times.

Model Name	Score
LASSO	0.05824
Ridge	0.13852
PLS	0.56680

6 Discussion

Because the initial data preprocessing stage led to a substantial decrease in the size of our feature space, the results presented are preliminary, as more data, additional pre-processing methods and the inclusion of more features are likely to increase the predictive power of our models.

First we note the performance of Naive Bayes, SVMs, and Neural Networks on the sentiment classification problem described in Section 4.1. Across all models, our classifiers performed substantially better than those found in existing literature (Section 2) with default parameters supplied by Scikit-Learn. However, our feature space (and dataset) is significantly reduced compared to other studies, which may make it more straightforward for our algorithms fit the optimal decision boundary to the dataset.

As mentioned in Section 5, the sparsity of our feature matrix presented interesting computational challenges. Initially, we applied Ordinary Least Squares to create a baseline for the predictive power of our "bag-of-words" feature matrix. The resulting MSE for the model was 340.467 using a standard 10-fold Cross Validation. This was unsurprising given the existing literature on methods related to data with high multicollinearity. When the feature matrix is highly multi-collinear, one would expect an OLS

model to be a poor predictor of response times. The results in Figure 2 also support the notion of high multicollinearity in our data, where the original feature space of size 174 was converted to a lower-dimensional space of 158 principal components to explain maximum variance in the data.

To more accurately model response times, we turned to Lasso and Ridge Regression, which are standards in implementing feature selection and decreasing model complexity, respectively. However, we observed poor performance for both models even when we increased the strength of the tuning parameter, λ. This could be for many different reasons. In particular, we suspect that Lasso reduces our covariate space by diminishing the effect of, or removing, covariates entirely. The same goes for Ridge Regression, with the exception of Ridge not zeroing out any covariates but rather, diminishing the effects of certain covariates. Because of the nature of our data (i.e. sparsity and multicollinearity), we posit that this feature reduction gets rid of possibly important factors in our data in an effort to regularize it.

Even when we run PLS, with all of our principal components, the model produces an r^2 of 56.6%, meaning that the model only explains 56.6% of the total amount of variance in the original dataset. This implies that our initial assumptions when creating the original hypothesis are possibly flawed. Indeed, we found that based on our limited data, there was a negligible difference between the response time of an agent between negatively labeled and positively labeled tweets ($\approx$ 8.5 minutes). Recall we posited that SW agent response times should be dependent on the sentiment associated with a tweet and that this is how Southwest should assign priority to different responses. In reality, there could be many more factors that impact customer service response times such as: resource constraints, the number of employees available, the time of day at which a tweet was sent, the prioritization in a queue of tweets at any given time, etc. As our models don't take into account any of these exogenous factors, our baseline hypothesis may not be capturing the true nature of the data and its relationship with the outcome variable.

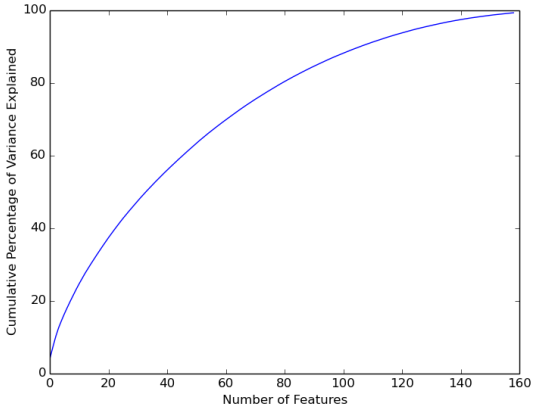

Fig. 2. Cumulative Distribution for PCA results.

7 Conclusion

While sentiment classification on Twitter data is well-scoped and studied, the issue of predicting the time to first response based solely on tweet data is less so. Our analysis is a step in this direction. Our approach makes the strong assumption that the time it takes a service representative to respond is directly and significantly correlated with the sentiment of a given tweet and that a Southwest Air agent should prioritize his/her response based on this sentiment. However, given our analysis, we know this is not solely the case. While the sentiment of a tweet may be a significant factor in determining the response time of an agent, it is difficult to model this effect in the absence of other relevant factors.
Moreover, we faced issues related to the reduction in our dataset's original size and the sparsity in our feature matrix. A future iteration of this project might explore a Word2vec transformation which could significantly reduce the problem of data sparsity. A future iteration might also attempt to model *all* responses, not just the time to *first* response. Regardless of these modeling difficulties, we recognize this type of analysis as a crucial step in beginning to understand a company's response time to customers. We hope that through future versions of our project, this engine could prove to be a useful tool to help a company learn how to prioritize its responses to customers.

8 Appendix

Included below are some sample interactions with our response time widget. Due to computational issues with implementing PLS predictions, the widget is built on the Ridge Regression solution. We presented this demo at the CS229 Project Fair.

Interaction 1:

1) **Please enter a tweet:** "Amazing ride with amazing staff! love you guys so much!"
2) **Your estimated response time by a SW Air Twitter Agent** is approximately 8.968284 minutes.

Interaction 2:

1) **Please enter a tweet:** "I hate this airline! Never riding with you all again. :("
2) **Your estimated response time by a SW Air Twitter Agent** is approximately 8.44797366 minutes.

References

[1] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," *LREc*, 2010.
[2] E. Kouloumpis, T. Wilson, and J. Moore, "Twitter sentiment analysis: The good the bad and the omg!" *Proceedings of the Fifth International Conference on Weblogs and Social Media*, 01 2011.
[3] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," 01 2009.
[4] S. I. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *ACL*, 2012.
[5] B. Liu, E. Blasch, Y. Chen, D. Shen, and G. Chen, "Scalable sentiment classification for big data analysis using naive bayes' classifier," in *2013 IEEE International Conference on Big Data*, Oct 2013, pp. 99–104.
[6] P. Borele and D. A. Borikar, "An approach to sentiment analysis using artificial neural network with comparative analysis of different techniques," pp. 64–69, 4 2016.

[7] J. Mahmud, J. Chen, and J. Nichols, "When will you answer this? estimating response time in twitter," 2013. [Online]. Available: https://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/view/6072

[8] J. Burchell, "Using vader to handle sentiment analysis with social media text," *Standard Error Full Atom*. [Online]. Available: http://t-redactyl.io/blog/2017/04/using-vader-to-handle-sentiment-analysis-with-social-media-text.html

[9] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: Sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, ser. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 79–86. [Online]. Available: https://doi.org/10.3115/1118693.1118704

[10] S. Kim, K. Han, and H. Rim, "Some effective techniques for naive bayes text classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 11, pp. 1457–1466, 2006.

[11] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," *Proceedings of the European Conference on Machine Learning*, N/A.

[12] K. L. Zhijie Liu, Xueqiang Lv and S. Shi, "Study on SVM Compared with the other Text Classification Methods," *International Workshop on Education Technology and Computer Science*, 2010.

[13] W. Zaghloul, S. M. Lee, and S. Trimi, "Text Classification: Neural Networks vs Support Vector Machines," *Industrial Management & Data Systems*, 2009.

[14] Y. Mao, Y. Shen, G. Qin, and L. Cai, "Predicting the popularity of online videos via deep neural networks," *arXiv preprint arXiv:1711.10718*, 2017.

[15] G. Muniz and B. M. G. Kibria, "On some ridge regression estimators: An empirical comparisons," *Communications in Statistics - Simulation and Computation*, vol. 38, no. 3, pp. 621–630, 2009. [Online]. Available: https://doi.org/10.1080/03610910802592838

[16] B. Wise, "Properties of partial least squares (pls) regression, and differences between algorithms," *Eigenvector Research, Inc.*, 2017.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

9 Individual Contributions

Debashri Mukherjee: Method benefits and trade-offs, implementation of Naive Bayes, SVM, Neural Network, interactive widget development, Methods section, Hyperparameter Tuning

Kathryn Tooker: Data labeling (sentiment dictionaries and VADER), modeling (Naive Bayes), interactive widget development, Literature Review, Modeling Scoping

Tyreke White: Data collection, data pre-processing, response time prediction, interactive widget development, Dataset and Features, Experiments & Results, and Discussion sections