Statistics 101C Final Project

Predicting Ratings from Yelp Reviews

Nick Monozon (705516863), Sarah Sotoudeh (205599342), Sean Tjoa (805257053)

Departments of Mathematics and Statistics, UCLA

Contents

1 Introduction

In this paper we propose our analysis of the Yelp Dataset, which was provided to the public as a part of the Yelp 2014 Dataset Challenge [6]. The Yelp Dataset is a snapshot of the Yelp ecosystem including data on Yelp businesses, reviews, and users. Our task was to train and test various machine learning models to best predict what rating a user will give to a business. The features we were given to work with include: user ID, business ID, the amount of useful/cool/funny reactions a user received on a review and in total, the text of the review itself, the state and city of the business, the average star rating of the business, the user's review count, the years in which the user had elite Yelp status, how many fans the user has, and finally, the average star rating the user gives.

We trained the following models using after feature engineering and recursive feature elimination: Logistic Regression, Support Vector Machines (SVM), Quadratic Discriminant Analysis (QDA), and Random Forest. For standard features, Random Forest performed best with a testing accuracy score of 81.10%, but raised some concerns other predictions on imbalanced data. Taking it a step further, we explored state-of-the-art transformer architectures and used Sentence-BERT to generate sentence embeddings, achieving an accuracy score of 88.44% with Logistic Regression. Below we go into detail and discuss our data preprocessing steps, experimental design, and results and analysis.

2 Data Preprocessing

Before training our models, we underwent many necessary preprocessing steps. Our preprocessing included simplifying the machine learning problem to a binary class prediction, employing language models to derive meaningful information from the `Review` text feature, attempting to extract information from imbalanced features (geographic data), and removing columns which provided extraneous information or were otherwise poor predictors of a user's star rating.

2.1 Creating the Binary Star Variable (Target)

We first investigated user ratings, namely the `Star` column. As seen below in this bar plot, most of the reviews received a rating of 4 or 5. Because the `Star` column is highly imbalanced, it would be very difficult to predict user star ratings since multi-class classification models require lots of training data from each class, and ideally a similar number of observations from each class. However, if we were to undersample the 2 to 5 star reviews to form an even distribution of data, we would not have sufficient training data for a robust predictive model.

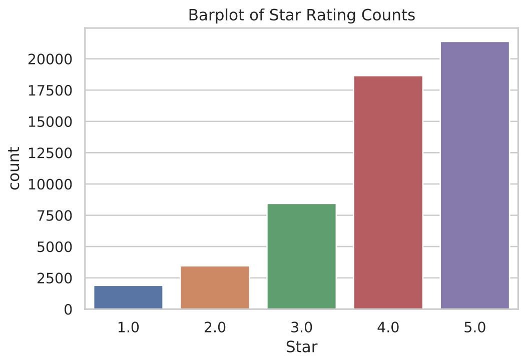

Figure 1: Imbalanced distribution of stars given

In response to the imbalanced distribution of our data, we decided to convert the quintic-class `Star` rating into a binary class. We did so by re-encoding ratings of 1, 2, and 3 stars to 0 and ratings of 4 and 5 to 1. This produced the following revised distribution:

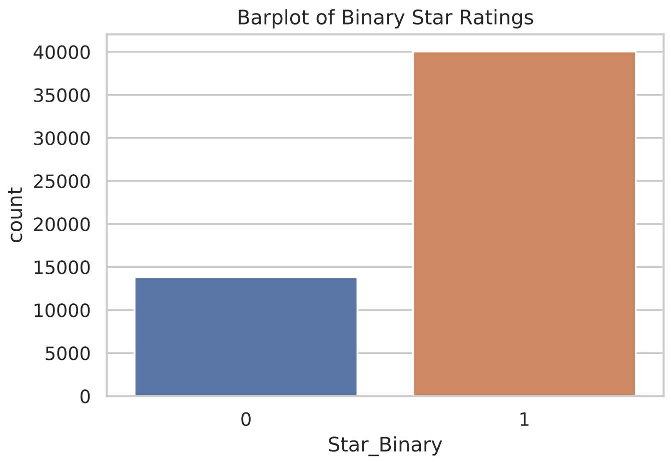

Figure 2: Label (binary star) distribution after feature preprocessing

While the binary star column is still imbalanced, it is significantly easier to work with when creating a predictive model. In addition, the model still solves a useful business problem: can we predict whether a user likes a business (4 or 5 star rating) or not (between 1 to 3 star rating)? Because the binary star column is imbalanced, we still need to be careful in our modeling procedures to make sure that the model does not simply predict the majority class. However, we decided to train our models under the assumption that each class of the target variable should be treated equally (prediction errors of either class are given the same weight) although this could be changed if necessary by adjusting the loss function to prioritize a specific class.

2.2 Extracting Sentiments from Review Text Data

One of the most important features of our model was developing an accurate sentiment score of the Yelp review text. Sentiment analysis intends to quantify the emotion of a sentence. Given our research goal to best predict a user's star rating of a business, producing a high quality sentiment analysis score of the review was one of our utmost priorities.

2.2.1 VADER

VADER (**V**alence **A**ware **D**ictionary and s**E**ntiment **R**easoner) is one of the most popular unsupervised learning algorithms used to calculate sentiment scores for social media posts, such as Yelp reviews. VADER is a lexicon and rule-based sentiment analysis tool which aims to map words to sentiment by referencing a "dictionary of sentiment" [7]. VADER is specially trained to handle abbreviations, capitalizations, repeat punctuation, and emojis, thus minimizing the time needed to clean the text data. Not only does VADER make sentiment analysis more efficient, but it also makes it more accurate. VADER is trained to assess subtleties in text that models such as bag of words cannot recognize [1]. For example, it recognizes typical negations ("not good"), contractions as negations ("wasn't good"), and modifiers ("wasn't very good"). VADER is also able to recognize the use and meaning behind punctuation ("great!!!") and capitalization ("GREAT") to suggest increased sentiment intensity. What made VADER most fitting for this analysis, however, is that it is familiar with recognizing sentiment-laden slang words ("sux"), modifiers ("kinda" or "friggin"), emoticons (`:D`), and acronyms ("lol"). VADER's clear advantage in analyzing sentiment of social media posts like Yelp reviews thus made it the most appropriate choice for calculating sentiment scores of the reviews. Below is a table representing how VADER successfully accounts for subleties in text.

Table 1: Sample VADER sentiment scores by category

Input Text	Negative	Neutral	Positive	Compound
"This computer is a good deal."	0	0.58	0.42	0.44
"This computer is a very good deal."	0	0.61	0.39	0.49
"This computer is a very good deal!"	0	0.57	0.43	0.58
"This computer is a very good deal!! :)"	0	0.44	0.56	0.74
"This computer is a VERY good deal!! :)"	0	0.393	0.61	0.82

We chose to analyze the compound score produced by the VADER sentiment analysis because it is the sum of all lexicon ratings which is then normalized between -1 and +1 [7]. Thus, we felt the compound score was the most encompassing metric to use for analyzing a review's sentiment as opposed to using an ordinal category of the score or only the positive, neutral, negative scores of a review.

2.3 Removing "User_id" & "Business_id" Columns

The easiest decision to make in this project was to remove the `User_id` and `Business_id` columns. First, these columns are not interesting to us in the context of our objective since we do not want to predict how a particular user will make a rating or how a specific business may be rated. Second, these columns do not provide us with useful information that would inform our model, especially since we already have the two most important pieces of information regarding each: `User_Ave_Star` and `Bus_Ave_Star`. Hence, we decided to remove the `User_id` and `Business_id` columns from the dataset.

2.4 Removing "State" & "City" Columns

Upon further investigation, we discovered that all reviews shared the same value for `State`, namely California, but that some reviews appeared to actually originate from businesses in Nevada (Reno and Sparks are in Nevada, but were mislabeled as California cities). Hence, we decided to drop this column because it did not provide much information. We then analyzed the `City` column and discovered that similarly, most reviews were concentrated in Santa Barbara and 5 nearby cities in Santa Barbara County. The bar plot below shows the counts of the top 5 cities in the dataset, all of which are in the Santa Barbara area:

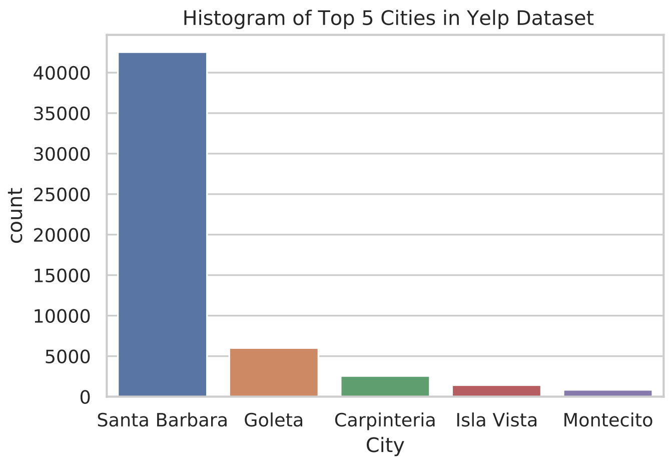

Figure 3: Distribution of review locations for top 5 cities

There were also around 16 other cities with less than 15 entries in each. This confirmed that the `City` column, although more rich than the `State` column, was not meaningful enough to put into the model. Because the "City" column was a highly imbalanced categorical variable, we thought we could potentially make it more balanced by making a column that represented the county that the business is located in. We believe that a business' county may impact users' reviews of the business because different geographic regions have different cultural values and levels of wealth which may impact a business' quality and reviewers' disposition towards rating. However, we decided to scrap the `County` column because over 99% of businesses were located in Santa Barbara County, so it would not have provided much unique information if it were in the model.

2.5 Refining the "Elite" Column

The `Elite` column in the dataset notes the years that the reviewer attained elite Yelp status. We thought that it would be ideal if we could turn this feature into a binary column where the entries are "yes" if the reviewer was elite at the time of the review and "no" if the reviewer was not elite at the time of the review. However, this dataset lacked information about the time of the review, so creating this binary column would not be possible. Instead, we thought that this column could be made more useful by tallying up the number of years in which a user was elite. This could be a useful feature in our model because a user that attained elite status for a longer number of years may be a more thoughtful reviewer on Yelp whose rating would carry more weight than someone with fewer years of elite status on Yelp. Thus, we decided to create a new column called `Years_Elite` which gives the number of years in which a given reviewer attained elite status.

2.6 Evaluating the "Bus_Ave_Star" Column

At first, we considered removing the `Bus_Ave_Star` column, as it appeared to leak information into the model about the "average" star rating of the business. However, it is important to remember that Yelp users see the distribution of a business' ratings prior to contributing a review, and this information may impact their star rating. For example, if a large number of Yelp users have left a rating of 1 or 2 stars, an elite Yelp user may feel inclined to agree with them and give a low rating to appeal to the community and gain more "useful" reactions and fans in order to maintain their elite status. Below is a summary of the columns of the original dataframe and the preprocessing methods we invoked to produce our final list of columns.

Table 2: Summary of feature engineering steps

Original Column	Preprocessing Action	Final Column
`User_id`	Removed	
`Bus_id`	Removed	
`Star`	Converted values to 0s (rating 1-3) and 1s (rating 4-5)	`Star_Binary`
`Useful`	Normalized and used in the final model	`Useful`
`Cool`	Normalized and used in the final model	`Cool`
`Funny`	Normalized and used in the final model	`Funny`
`Review`	Used to calculate compound VADER sentiment score	`Sentiment_Score`
`State`	Column removed because of imbalanced classes	
`City`	Column removed because of imbalanced classes	
`Bus_Ave_Star`	Normalized and used in the final model	`Bus_Ave_Star`
`User_Review_count`	Eliminated during feature selection	`User_Review_count`
`User_Useful_count`	Eliminated during feature selection	`User_Useful_count`
`User_Funny_count`	Eliminated during feature selection	`User_Funny_count`
`User_Cool_count`	Eliminated during feature selection	`User_Cool_count`
`Elite`	Converted into number of years as elite Yelper (not used)	`Years_Elite`
`User_Fans`	Eliminated during feature selection	`User_Fans`
`Users_Ave_Star`	Normalized and used in the final model	`Users_Ave_Star`

The columns on the right represent the final columns which we conducted our analysis on and used for feature selection. Before we discuss how we trained our models, it is important to note that we took the important step of scaling our data, which is the process of transforming data so that it fits within a specific scale, in this case the standard normal distribution $Z \sim \mathcal{N}(0, 1)$. Transforming data is especially impactful for improving the performance of support vector machines (SVM), as SVM relies on measures of how far apart data points are. We observed in training our SVM model that accuracy improved greatly after scaling our data which confirms the importance of completing this preprocessing step.

3 Experimental Setup

After completing the feature engineering process, we decided to consider the subset of features that we would use when building our predictive models. First, we decided to check the correlations between our predictors, but the majority of the variables had very low correlation values (less than or equal to 0.1). Plotted below is a correlation matrix of the predictors most correlated with `Binary_Star`. While the predictors don't have high correlation values with the target variable, fortunately, they do not display high multicollinearity.

Table 3: Correlation matrix of possible predictors and target

	`Bus_Ave_Star`	`Users_Ave_Star`	`Sentiment_Score`	`Rating`
`Bus_Ave_Star`		0.100	0.212	0.313
`Users_Ave_Star`	0.100		0.164	0.268
`Sentiment_Score`	0.212	0.164		0.397

`Rating`	0.313	0.268	0.397

Using recursive feature selection (RFS) to find the best subset of k features for $k = 1, \ldots, 11$ we fit a simple logistic regression model and determined that the model with the best 6 features provided the best tradeoff between 5-fold cross validation accuracy and model complexity. The accuracy of each logistic regression model for the best subset of k features is in the plot below.

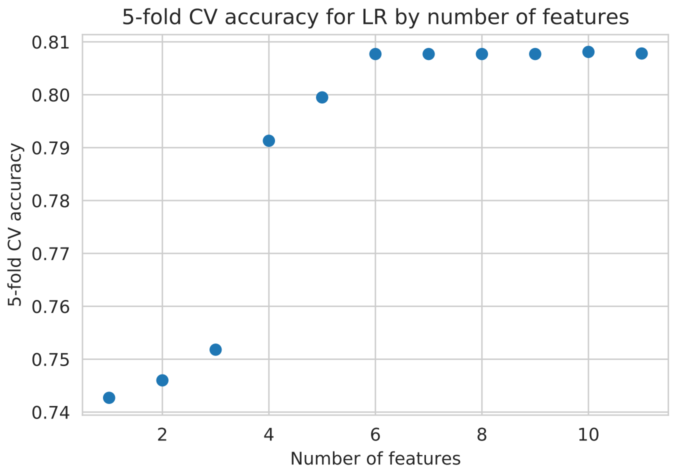

Figure 4: Logistic regression 5-fold cross validation accuracy per best subset of $k \in \{1, \ldots, 11\}$ features

These six features were `Useful`, `Cool`, `Funny`, `Bus_Ave_Star`, `Users_Ave_Star`, and `Sentiment_Score`. Upon examining the pairwise correlation matrix, we noted that `Useful`, `Cool`, and `Funny` were strongly correlated features. As such, we thought that by employing principal component analysis (PCA), it would be possible to retain predictive accuracy while reducing the dimensionality of the feature space. As such, we performed PCA on these 3 feature vectors, finding that the first and second principal components explained 96.7% of variance and 2.13% of variance in the original data, respectively. Hence, we opted to replace the `Useful`, `Cool`, and `Funny` features with a single feature corresponding to just this first principal component. Fitting our reduced 4-feature logistic regression model, we achieved a 5-fold cross validation accuracy of 0.7897, a reduction from our previous accuracy of 0.8077 with the 6 original features. Due to the decreased accuracy, we chose not to use the first principal component of these feature vectors as a predictor, and proceeded to fit additional models using the six features we previously identified through RFS.

With our list of features identified, we proceeded to randomly split our data into 80% training data and 20% testing data. Despite the noted class imbalance discussed in Section 2.1, we decided that it made sense to classify observations from the distribution as given: in general, positive ratings (i.e. label 1 observations) on Yelp are more common than negative or neutral (i.e. label 0 observations). For our project, the models we sought to compare were Logistic Regression, Support Vector Machines (SVM), Quadratic Discriminant Analysis (QDA), and Random Forest. The processes and approaches for fitting each of these four models is outlined below.

1. **Logistic Regression**. Logistic Regression is a discriminative model which estimates the conditional probability that a sample $\mathbf{x}_n$ belongs to class 1, denoted by $\mathbb{P}(Y = 1 | X = \mathbf{x}_n)$ [2]. Although we already fit a logistic regression model to perform recursive feature selection, we performed a grid search over

possible penalty functions and penalty strength and found that introducing an L2 penalty term helped to improve testing performance.

2. **QDA**. QDA is a generative model that creates a decision boundary by assuming that the target and the predictors follow a joint normal probability distribution [3]. We decided to try QDA given the flexibility of the model's nonlinear Bayes decision boundary. In order to find the best fit of the model, we performed a grid search over QDA's covariance regularization parameter. We found that regularization parameters between 0.2 and 0.4 helped to avoid overfitting on the training data.

3. **SVM**. SVM is a supervised machine learning algorithm commonly used for classification and regression. SVM works by translating data to a high-dimensional space in order for the data to be classified using a hyperplane separator [4]. SVM is known for performing well with limited data, and since we lack samples for negative reviews, this advantage of SVM was of great value to us.

4. **Random Forest**. Random Forest is an ensemble learning method used for classification. The random forest algorithm works by aggregating predictions from a set of decision trees that are given random subsets of the data and classifying based upon the majority vote of the trees [5]. We decided to use Random Forest because it is an ensemble model that tends to produce good accuracy when given data without strong associations to the target variable. The Random Forest has many hyperparameters that can be tuned; we performed random grid search cross validation over the number of estimators (number of Decision Trees), max depth of the trees, and max number of features of the trees. Although there are many other hyperparameters that can be tuned in the Random Forest, attempting to do so with this model would take a long time.

4 Results and Analysis

After training each of our 4 models for the hyperparameters identified, we evaluated each on the testing data, which made up 20% of the dataset after preprocessing. The accuracy of each model is given in the table below.

Table 4: Training and testing performance across classical models

Model	Training Accuracy	Testing Accuracy
Logistic Regression	0.8111	0.8080
QDA	0.8098	0.8079
SVM	0.7991	0.7966
Random Forest	0.8172	0.8110

As reflected above, Random Forest was our best performing model with a training accuracy of 0.8172 and a testing accuracy of 0.8110. This isn't too surprising, as Random Forest is essentially classifying by employing a majority vote across an ensemble of many Decision Trees trained on different subsets of data, providing for accurate predictions. The classification report for the Random Forest evaluated on the testing data is given below.

Table 5: Classification report of Random Forest for testing data

	Precision	Recall	F1 Score	Support
0	0.38	0.77	0.51	1391
1	0.96	0.81	0.88	9378
Accuracy			0.81	10769
Macro Avg	0.67	0.79	0.69	10769
Weighted Avg	0.88	0.81	0.84	10769

Due to the large costs associated with performing a grid search over hyperparameters in a Random Forest, in the future we would like to explore more hyperparameter tuning of the Random Forest by experimenting with bagging, split criteron, and max leaf nodes. Upon looking at precision and recall, it seems like our model is better able to predict labels of 1 (positive reviews) rather than 0 (negative reviews). In a future revision, it would be beneficial to explore balancing classes to ensure that the models are not biased toward predicting the majority class.

Although we are proud of the thorough analysis we have completed on the Yelp dataset, there are a few areas in which we have identified for improvement. Primarily, having access to more data would increase model performance in terms of testing accuracy. Due to there being a major class imbalance between class 0 and class 1 for the `Star_Binary` column, undersampling the majority class would substantially reduce the total amount of available data. Ideally, we would have more data to inform our models with. This may also allow us to build models to conduct multi-class classification and predict the `Star` column instead of the `Star_Binary` column. This task would have been difficult to do with the goal of having high accuracy due to the limited data on lower star reviews.

Another point of consideration is to build a model to predict a variable that would have a greater business value than `Star_Binary`. A company like Yelp does not necessarily gain valuable insight from being able to predict a specific user's star rating from given business and user features. Although this project offered great technical insight into model prediction, we believe that it would also be an interesting approach to consider analyzing problems that could benefit Yelp or businesses on Yelp.

In addition, we decided to utilize VADER to calculate sentiment scores for our model, but we could have extracted more features from the `Review` text to experiment with. It would be interesting to utilize a large language model to compute sentiment and compare the differences to VADER, a rule-based sentiment model, but we have no sentiment labels for our dataset to use for comparison. On the other hand, we could have experimented with simpler features such as the TF-IDF of positive and negative words. As a follow-up, however, we discuss how we implemented large language models—specifically Sentence-BERT— to obtain more accurate predictions in the next section.

Acknowledging the different avenues of analysis we could have taken with this dataset, we are still pleased with our results. We successfully trained four different machine learning models and achieved testing accuracy scores of 80% or higher for every model besides SVM. More importantly, we followed important processes and techniques including feature engineering, feature selection, cross validation, and more, which allowed for the success of this analysis.

5 Employing Large Language Models for Accurate Predictions

For our final project, we wanted to use classical models with relatively simple features to create an interpretable model because Yelp stakeholders may be interested in understanding the model's output. However, we decided to experiment with deep learning techniques such as BERT and Sentence-BERT to obtain higher accuracy in our models. Therefore, we will briefly discuss the application of transformer-based models within our problem of predicting Yelp ratings.

Although difficult to explain why, large language models like BERT and Sentence-BERT generally outperform lexicon and rule-based models such as VADER due to their ability to capture context/attention and the enormous number of parameters and training data they use. The architecture of these models are hard to describe, but the output of both BERT and Sentence-BERT is simple to grasp: a relatively low-dimensional numerical vector. The purpose of these models is to translate a high-dimensional input, such as a sentence or paragraph, into a lower-dimensional output that captures the semantic meaning of the input.

BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) is a transformer-based model released by Google in 2018 that attempts to learn contextual embeddings for words [8]. Fortunately, there are many BERT models that have been pre-trained for sentiment analysis that are available for free use on the HuggingFace platform. We experimented with BERT models that were pre-trained on Twitter posts for sentiment analysis. Essentially, we used model architectures that incorporated BERT's architecture, plugging the embeddings from BERT into a separate model that was designed for sentiment classification. However, we estimated that the time to calculate sentiment scores for the 53,845 rows of our dataset would take approximately 8 hours, so we decided not to use it out of convenience. Although BERT models would most likely produce superior sentiment scores for the data compared to VADER, we do not have the ability to compare their performances on our dataset because we do not have the "true" sentiment scores of each of the data points. Still, BERT would probably outperform VADER since it achieved state-of-the-art performance in sentiment classification of Twitter data in 2021 [9].

Evidently, the models we experimented with were incredibly computationally-expensive, so we decided to try a different architecture that also incorporated parts of the BERT architecture called Sentence-BERT. Sentence-BERT's architecture is much more computationally-efficient and captures semantically-meaningful sentence embeddings [10]. In addition, we decided to not use a pre-trained model on top of the Sentence-BERT encoder and to only use the first 128 words of each Yelp review because the runtime and memory requirement to make sentence embeddings grows quadratic with the input length [11]. Thus, we obtained 728-dimensional numerical "sentence" embeddings of the reviews (you can encode paragraphs instead of single sentences).

We then decided to train a Logistic Regression model on the sentence embeddings from the reviews, `Bus_Ave_Star`, and `Users_Ave_Star`, achieving an eight-percent boost in accuracy from the previous model. It seems like the precision of class 0 predictions is significantly better in this model compared to our previous model (0.69 vs 0.38), so we speculate that the sentence embeddings better capture the overall sentiment of the `Review` data than the VADER sentiment score. Note that the model was trained with an L2 penalty term to better generalize to new data, so the testing accuracy was higher than training accuracy in this case.

Table 6: Training and testing performance

Model	Training Accuracy	Testing Accuracy
Logistic Regression (Sentence Embeddings)	0.8831	0.8844

Table 7: Classification report of LR with SEs for testing data

	Precision	Recall	F1 Score	Support
0	0.69	0.82	0.75	2298
1	0.95	0.90	0.92	8471
Accuracy			0.88	10769
Macro Avg	0.82	0.86	0.84	10769
Weighted Avg	0.89	0.88	0.89	10769

Therefore, we see that employing large language models has the potential to greatly improve accuracy for our problem. It is important to note that the embeddings could be improved by not truncating the `Review` text to only 128 words and that we used the raw embeddings in our model instead of using an additional model to process those embeddings into a sentiment score. Although sentence embeddings greatly improved the accuracy of our model, we do not understand the intricacies of how they work at our level of knowledge and run the risk of using them improperly. Hence, we would recommend using the previous model if the predictions of this model had real-world impact (i.e. used by stakeholders to make decisions). However, we found that applying transformer-based models to our project was fascinating and fun, and we would like to learn more about deep learning in the future.

References

[1] Hutto, C., Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. Proceedings of the International AAAI Conference on Web and Social Media, 8(1), 216–225. https://doi.org/10.1609/icwsm.v8i1.14550

[2] Lawton, G., Burns, E., Rosencrance, L. (2022, January 20). What is logistic regression? . Business Analytics. Retrieved December 9, 2022, from https://www.techtarget.com/searchbusinessanalytics/definition/logistic-regression

[3] Quadratic Discriminant Analysis (QDA). PennState: Statistics Online Courses. (n.d.). Retrieved December 9, 2022, from https://online.stat.psu.edu/stat508/lesson/9/9.2/9.2.8

[4] TechTarget. (2017, November 29). What is support Vector Machine (SVM)? TechTarget. Retrieved December 9, 2022, from https://www.techtarget.com/whatis/definition/support-vector-machine-SVM

[5] What is Random Forest? IBM. (n.d.). Retrieved December 9, 2022, from https://www.ibm.com/cloud/learn/random-forest

[6] Yelp open dataset. Yelp Dataset. (n.d.). Retrieved December 9, 2022, from https://www.yelp.com/dataset

[7] Welcome to Vadersentiment's documentation. VaderSentiment 3.3.1 documentation. (n.d.). Retrieved December 9, 2022, from https://vadersentiment.readthedocs.io/en/latest/

[8] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K. (2019, May 24). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv.org. Retrieved December 9, 2022, from https://arxiv.org/abs/1810.04805

[9] Chiorrini, A., Diamantini, C., Mircoli, A., Potena, D. (2021). Emotion and sentiment analysis of tweets using BERT. CEUR Workshop Proceedings. Retrieved December 10, 2022, from https://ceur-ws.org/Vol-2841/DARLI-AP_17.pdf

[10] Reimers, N., Gurevych, I. (2019, August 27). Sentence-bert: Sentence embeddings using Siamese Bert-Networks. arXiv.org. Retrieved December 9, 2022, from https://arxiv.org/abs/1908.10084

[11] Reimers, N. (n.d.). Computing sentence embeddings. Sentence-Transformers Documentation. Retrieved December 9, 2022, from https://www.sbert.net/examples/applications/computing-embeddings