Sentiment Analysis & Visualization on Hotel Review Data

Noah Sealy
B00726289
Master of Computer Science
Dalhousie University
Halifax NS, Canada
noah.sealy@dal.ca

***Abstract*—Text reviews can provide key insights to businesses which can lead them to newfound success. Unfortunately, processing text reviews can be very time consuming. The data set used in this project relates to text reviews for hotels in Europe. This project implements and evaluations a support vector machine to classify text review polarity; a process known as sentiment analysis. A visualization of the data is also provided using maps to analyze data in the context of time and space. Overall, this project has developed working tools to assist a hotel management team in analyzing customer review trends.**

1. Introduction

The modern Internet contains a vast web of online shops and services. This pocket of the Internet ranges from food delivery to full online stores. Many of these sites allow for its customers to leave text reviews, along with scoring systems to rate the quality of the services provided. This text driven review system does not just stop at online services, but is applicable for customers who even shop in person, with the help of web applications such as Yelp and Business.com. A very popular use of this review system is in the hotel industry. Here, customers typically review the quality of service, amenities, and staff after a stay. Overall, this service provides a powerful method for a hotel to gain feedback on their various services by those experiencing them first hand.

The problem here is not typically the data itself, rather the processing of such large amounts of text data. Depending on many factors, a hotel chain may see thousands of guests each month, with a potentially large subset of those guests leaving reviews. This feedback can provide beneficial information for the hotel, which if acted upon can provide a free method of increasing the quality of their services. This has the potential to increase the number of visitors they have, which will ultimately increase their overall income. Even when fully processed, the implications of the data are not easily established by just parsing through a .csv file.

The goal of this project is to provide tools for which a hotel management can use to process and analyze data yielded by such reviews. This project has the intention of providing a solution to the problem of inefficient data analysis relating to these reviews. This solution has the potential to directly benefit hotels which deal with text based customer review software. This project investigated various machine learning related models which are able to quickly process a large set of text based review data. The text processing on the review data was based from the concept of sentiment analysis. Lastly, the project also implemented a visualization component in order to analyze this data in the context of time and space.

Based on the broad topics of data science covered in CSCI 6515 at Dalhousie University, three topics were chosen to make up the project. These three topics were building a predictive model, evaluating said model, and visualization of its data. Following these topics, the chosen model was implemented and evaluated in order to show the effectiveness of automated text processing. The model options were rule-based models, binary logistic regression, support vector machines, and multi-class support vector machines. The evaluation metrics used were model accuracy, precision, recall, and f1 score. The data was presented in a data visualization web application. The visualization method options were matplotlib, Plotly, and Dash. Further detail of this investigation, the choices made , and implementation behind each topic will be discussed throughout.

The data set itself was a collection of over 500000 hotel reviews from Booking.com [1]. The data set is also discussed in the Background section, in terms of its statistical significance, as well its validity relating to the properties of big data.

2. Background

This section provides an investigation into various solution options to the topics discussed in the Introduction section. As mentioned, these topics are training a model, evaluating that model, and data visualization. Before these are discussed however, a brief description and investigation of the data set itself is provided.

2.1. The Data Set

As discussed in the first lecture, there are five properties which classify a set of data as, big data. These properties relate to volume, velocity, variety, veracity, and value of the data [2]. Though a lot of text review data was available on Kaggle [1], the selected data set stood out due to its qualifications in regards to these properties.

The data set that was selected is named 515K Hotel Reviews Data in Europe, or just the Hotel Review data set. As mentioned, it was found on Kaggle [1]. The data set contains information related to over 500000 instances of hotel reviews for luxury hotels in Europe. Each hotel in the data set has multiple instances of reviews from different visitors, along with an average review rating for the hotel, and some geographical information. Each review instance has data showing some demographic information of the reviewer, such as their nationality. Finally, the review itself is broken down into positive components, negative components, and an overall review score out of ten.

This data set also exhibits a subset of the provided big data properties introduced prior. The properties that are most frequently displayed in this data set are variety, veracity, and value. This subsection will end with a brief analysis of these three properties in regards to the Hotel Review data set.

2.1.1. Variety. The Hotel Review data set exhibits two qualities of variety which makes it appealing for a study involving big data. The first is that there is a variety of data types expressed throughout the data set. Examples of this include the ”Review Date” column as a date value, ”Average Score” as a floating decimal value, ”Reviewer Nationality” as various specific classes, and the ”Positive and Negative Review” columns as full length text data. This allows for various kinds of processing on the data, which may allow for analysis involving classification, regression, and natural language processing.

The second quality of variety is within the data instances themselves, which allow observers a better perspective of the data. Examples of this are, while the data set contains data relating to the reviews and scoring by the reviewer, it also offers data relating to the reviewer's nationality. Geographical data is also given for the hotel itself, as well as its average review score. With this variety, researchers are able to gain insight on the effects location and nationality may have on a review. Additionally, it provides an idea of how different a single instance of a review score is compared to an average review score. Overall, this sort of variety adds a lot of power and potential to the Hotel Review data set, as it provides insight into many more factors than just the reviews.

2.1.2. Veracity. In the context of data, veracity checks both the quality and validity of a data set. It is important for data to have this trait, as with it researchers are able to analyze trends and make predictions relative to the real world. The Hotel Review data itself is provided from the public review data taken from Booking.com [3]. Booking.com is a website devoted to helping travelers find and book flights, hotels, and other vacation services across the globe. Upon selection of an available hotel, numerous reviews are available. These hotels are given by users of the Booking.com website which have previously stayed at the hotel. Data relating to the hotel, review, and reviewer are all tracked through the website itself. This gives ground to data validity as it was generated from real people who made real reviews.

A potential concern relating to the veracity of this data set is that the subset of reviews which represents each hotel, does not appropriately represent the average review of the hotel. This may cause issues to arise, as the overall sentiment of the reviews may not be the true reality given the sample data. In order to validate that the data properly represents each hotel, correlations can be found between the averaged review scores per hotel, and the actual average review score provided by the hotel. This data corresponds to the Average Score and Reviewer Score columns in the data set.

Figure 1 shows the correlation between these averaged values when plotted against each other. Pearson's correlation was chosen as the correlation metric as the data is linear, numeric, and fit to a normal distribution. Calculated with the Pandas .corr() function, Pearson's correlation coefficient is $r = 0.9635$, indicating a strong linear correlation between the averages. Given this strong linear correlation, it is evident that the review score data properly represents the review score of each hotel in the data set. In order to show this is true for the population, and not just occurring by chance given this mean score, a test for statistical significance of this data is provided.

Figure 1. Plot depicting the strong linear relationship between the calculated average sample review scores, and the hotel review score provided by the hotel. Pearson's correlation coefficient is $r = 0.9635$.

According to [4], a correlation value can be tested for statistical significance using the following formula, where r is the sample correlation and n is the number of samples. In this case, $r = 0.9635$, and $n = 1492$ [4].

$$t = \frac{r * \sqrt{n-2}}{\sqrt{1-r^2}}$$

The null hypothesis in this case is that the population correlation is equal to zero, while the alternative hypothesis is that the population correlation is anything but zero [4]. A significance level of $\alpha = 0.05$ was used. The t-value found from the equation was $t = 138.93$. Using online software, the p-value found was $p < 0.00001$ for a two tail test, which can be expressed as $p = 0$. Given that $p = 0$, the null hypothesis is rejected, and it can be said that the population data has a significant linear correlation. Thus, the sampled review scores properly represent each hotel they were sampled for, further validating the veracity of this data set.

2.1.3. Value. As described in the presentation slides of [2], data should be looked at by a company as an asset. In order to be viewed this way, the data must provide value to that company upon exploitation. The Hotel Review data set has the ability to bring value to companies who are willing to process it. As mentioned in the Introduction section, the data set will provide hotel businesses insight into what their customers enjoyed about their visit as well as, perhaps more importantly, what they disliked. Based on this data, the hotel may adjust their services and amenities accordingly, in order to provide a better experience for the customer. This experience has the potential to increase visitors, thus increasing the hotel's income.

Though hotel exploitation for customer feedback is an obvious value, this data also contains various data points relating to the demographics of the customers. This data set is thus able to provide an insight into the opinions of customers in the context of culture and nationality. Once this data is processed, it may be able to grant the hotel insight into specific ways of accommodating for different people, in order to further maximize the overall visit quality for a specific customer.

2.2. Training a Model

The core of a machine learning project is the model itself. Choosing the most robust model given the situation can have a huge impact on the success of the application. Throughout the literature provided for the sentiment analysis problem, there are three recurring types of models which can work as a solution. The options fall under the concepts of rule based models and supervised models [5]. Solutions related to both options will be briefly investigated, all in reference to the specific problem and data set present in this project.

2.2.1. Rule Based Classification Models. According to [6], rule based classification models are represented as a set of IF statements. In the context of text processing, these conditional statements classify input data depending on hard coded rules or patterns set by the developer. In the context of sentiment analysis, those rules and patterns would relate to the polarity of words within the input text. For example, the word "fantastic" may contribute a count to a positive count, while the word "terrible" may contribute to the negative. At the end of parsing through the text, the overall polarity of the document can be determined from which count was greater. For example, if the rules deemed 12 of the words in the input text as negative, and only 3 as positive, the model would classify the overall sentiment of the text data as negative. The options for rule based models are self developed rules and a pre-built rule based; both will be discussed.

In order to self develop a rule base, one must hard code all possible words and patterns into conditional statements. For example, the developer may set words which contribute to a positive count as ["fantastic", "great", "enjoyable", "perfect", "awesome"], while they may set words to contribute to a positive count as ["waste", "regretful", "idiotic", "disgusting", "terrible"]. Developers must also create rule sets for patterns in language that would otherwise contradict just the rules for the words themselves. For example, "not so great" should contribute to the negative count, but just the word rules would classify it as positive. The complexity of the model grows with these rule sets, as the model will be able to evaluate text more accurately with more knowledge of the language. Though beneficial for those able to develop large rule sets, a limitation is presented here; creating a large scale rule base which catches the majority of words and language patterns independently is an extremely time consuming task. This extreme amount of time consumption is a detrimental limitation to self developed rule based models.

There is a solution to the immense development cycle of a self developed rule based model though. This involves the process of downloading and importing a pretrained rule based model. Textblob offers a library which includes a model that performs sentiment analysis, based on a developed pattern library [7]. The Textblob library will process the text with a black box sentiment analysis algorithm, and return the overall polarity score of the text.

The robustness of a rule based model comes from its potential to have a deep understanding of the human language. The models could catch edge cases in the language that other models may not pick up on so easily. As mentioned, the power to catch patterns that are contradictory to the polarity of the words used within is extremely valuable.

Textblob compensates for the issues described with the self developed rule based model, thus making it the more appealing model to choose for rule based classification models. However there are still a few issues present with using Textblob as the preferred model for this project. As mentioned, Textblob returns a polarity metric which is between -1 and 1, to denote negative and positive classifications respectively. This is more complex than simply labelling the output of the model as positive or negative. A full scoring function would have to be created in order to properly represent these values as positive or negative polarity. This scoring function needs to take into the range of values this polarity may have, which could lead to a complicated implementation. Additionally, the Textblob model utilizes algorithms and libraries which are hidden to developers who import them. Due to this, there will be a substantial amount of theory lacking during further discussions of the model throughout this project. Though this is not a limitation of

the model itself, it is preferred that more theory for the model of choice is present.

2.2.2. Supervised Classification Models. In the terms of sentiment analysis, supervised learning requires training a model with text data that is classified as either positive or negative [8]. The positive class corresponds to text that has a positive polarity, while negative is the opposite. Through training, the model will approximate a function which fits the data in some n-dimensional space. Training involves updating model parameters based on its performance compared to the ground truth values, that are represented as a label for each instance in the data set. As mentioned, the labels will be a mixture of words ranging from very negative polarity to very positive. If this function is fit correctly, the supervised model can take unseen text data as input and predict which label it will be classified as.

Currently, the polarity of each review in the Hotel Review data set is represented as a review score between 0 and 10. A limitation of a supervised approach is that scores will have to be converted to refined classes, such as "Positive" and "Negative". A function will be used to convert anything lower than 5 to a negative, and anything else a positive. This conversion is simply a hyperparameter, and may have to be tuned as the data is analyzed closer. This is somewhat of a limitation, as some depth of the overall reviews are lost. For example, a review score of 10 is a lot different than a review score of 5.2; though with this system both are considered positive reviews.

The other limitation of the supervised approach is that the model will train based on the text data itself, and not from a source that understands the whole language. This is an issue as the model may fit to words incorrectly based on their use throughout the text. An example of this is an instance when a reviewer is sarcastic, by saying "having bugs in your room on your birthday is great"; giving a score of 4.3. The model may begin to associate the word "great" with negative reviews. Though this limitation can be detrimental to the model, it should not be an issue if there are not many instances of uses of positive words in negative reviews, or vice versa. In order to validate this, an analysis of word counts can be made to find if any misplaced words commonly appear. Figure 2 shows the top words in the negative count and positive counts, after review score labeling and text processing on the text data. Overall, it is evident that our data should not fit misplaced words as they do not occur frequently enough. This is the case for just this data set though, this limitation may become relevant with unseen data.

In terms of available models for supervised learning, there are numerous options which could be chosen for this problem, with various ranges of complexity. As this project is of a smaller scale, extremely complex models which would take a long time to train have been ruled out as options. This is mainly due to the time constraints of the project, as an ideal model would not take too long to train and test. Two models stand out which are relatively less complex and may be appropriate for this problem

Positive	Negative
room	room
staff	hotel
location	staff
hotel	location
breakfast	bed
good	breakfast
great	small
bed	night

Figure 2. A glimpse at the most frequent word list in positive (left column) and negative (right column) labelled subsets of text reviews in Hotel Review data set.

based on the reviewed literature [9], [10]. The supervised classification models to be investigated are Binary Logistic Regression and Support Vector Machines (SVMs), with multi-class SVMs. These options will be briefly discussed as potential model options for this project.

Binary logistic regression is a kind of classification model which aims to fit a function in n-dimensional space in order to be able to differentiate between two classes. This usually results in an S shaped logistic curve in two dimensions, as the classes can be depicted as 0 and 1 on the y axis [9]. The function will output a value that represents the probability of the data instance to be a class. There will be a threshold set which determines whether that class falls into class 1, or else class 0 [11]. Training of this logistic curve is based on fitting the curve through maximum likelihood estimation. This involves testing out different functions which split the data before taking its log, and calculating the likelihood of the data given the logistic curve produced. Taking the logs of the original data in terms of a fitted line will yield the probability of that data instance belonging to each class, as mentioned prior. The final logistic curve is based on the original data line which yielded the greatest likelihood value; thus calculating the maximum likelihood of observing the data. The final curve can be tested for statistical significance after training by calculating its R^2 value, and with that the p value. If the null hypothesis is rejected, it can be said that the logistic curve fit is able enough to predict other data within the input samples population. Thus during testing and beyond, this curve can be used to predict the binary classes of other data instances.

Binary logistic regression would provide an elegant solution of essentially squishing text data in an n-dimensional vector space down into a logistic curve, which maximizes the likelihood of that data occurring. The class labels used with the model would be "positive" and "negative" to symbolize the overall polarity of each review. Given statistical

significance, that function could then be used to predict the polarity of other word vectors relating to hotel reviews. The limitation to binary logistic regression is that it fits data to predict for only two classes, in this case ”positive” and ”negative” [11]. As discussed previously, a solution to a potential limitation of the depth of the selected model is to expand these binary labels into labels which depict partial values, in order to further the range of classes available. As this solution would add more labels to the data set, it directly clashes with the use of the logistic regression model, thus one would have to be sacrificed in the development process.

This leads to the supervised classification model known as support vector machines, or SVMs. Support vector machines are another type of binary classification model which use a threshold in the dimensional space of the data. This threshold will determine which class a point of data falls into. This threshold is at a fixed midpoint between a space which divides the threshold and the data, called a soft margin [12]. This is called a soft margin as some outliers in the data may fall into it, which are also known as support vectors. The threshold, also known as the hyper plane, can sometimes have trouble fitting to data presented in its original dimension. In order to compensate for this issue, support vector machines use a kernel function, which will represent the data in a different dimension than the original [13]. For the sake of efficiency, the data will not actually be projected, rather the calculations on each data point during training are calculated in the higher dimension; this is known as a kernel trick. As this model is not probabilistic in nature, testing for statistical significance will not be required after training. SVMs provide an effective method of classifying binary labelled data at high dimensions, and have been previously trained for sentiment analysis [10]. With the use of the kernel trick, the high dimensional text data can be classified efficiently with the use of a hyper plane with support vectors.

As it seems that SVMs were built for the processing of high dimensional data, they were chosen over the choice of the binary logistic regression model, as finding the maximum likelihood estimate in such a high dimensional space which the text data will be presented in may prove to be much more complex. However, SVMs present the same limitation as binary logistic regression as they can only train with two labels. In order to compensate for this limitation, a multi-class SVM classification model will be briefly discussed, as it may have implications to solving this problem in the future. SVMs are thus more flexible for advancements in future work compared to a binary logistic regression model.

Multi-class SVM functions as a binary SVM would, with the exception that for every class pair a binary SVM hyper plane is fit [14]. Though this is advantageous as multi-class classification is now available, fitting the model may take much longer as it is a much more complex task. The advantage of this, as discussed, is the expansion of the range of polarity. The project only processed data with two classes, but because of the future implications, the SVM was chosen as the model to be implemented.

2.3. Evaluating the Supervised Model

This section will briefly discuss the metrics chosen for evaluation of the SVM classification models. The metrics will not be compared as they are all commonly used, and all possess different kinds of importance to model evaluation. The evaluation metrics used are the model testing accuracy, precision, recall, and f1 score [10].

The accuracy will simply measure the correct instances of model prediction in regards to all instances of model prediction. This will yield a percent score which judges the models correctness in prediction. This accuracy may be misinforming due to errors in model training, especially if the data set is unbalance, or the model becomes over fit to the training data. As accuracy alone is not the most descriptive measure, it is important to include precision, recall, and f1 score.

The precision will measure the correctness of the model in terms of total instances of model prediction [15]. The precision will yield the proportion of correct answers that are actually correct, and not false positives. The recall will measure the correctness of the model in terms of incorrect model predictions [15]. The recall will yield the proportion of correct answers, that were not missed by false negatives. Together precision and recall are important in evaluating if the model was over fit or trained on unbalanced data. These metrics bring depth to the accuracy score, as they tell if the model is actually classifying at that rate, or if there are other issues related to the model itself or data.

The f1 score will further process the recall and precision score in order to investigate the false positive and false negative rates of the model [15]. A high F1 score will mean a low false positive and false negative rate, which could further reinforce the grounds of having a robust model.

Overall, the listed metrics are crucial to deeply evaluating a model. These metrics will not just score the model, but also indicate incorrect implementations with it and the data.

2.4. Data Visualization

This section will very briefly describe the process of choosing a visualization framework to use. From previous experience with visual analytics, it seems as though there are three main choices for visualization libraries in Python: matplotlib, Plotly, and Dash. Though matplotlib and plotly are both compelling tools to visualize data in numerous ways, they are limited to the output consoles and GUIs of the Python development environment. The goal for this project is to develop an application which summarizes the data sets and models used. This application will help hotel owners to quickly review the overall feedback of their services. In terms of application development, Dash is the clear choice here, as it is a framework built off of JavaScript in order to develop web applications using Python. The visualization figures themselves are actually coded and rendered with Plotly.

Overall, the Dash framework provides an elegant solution for the development of a visualization app. The framework will use JavaScript to render Plotly figures in order to summarize the data set all from Python.

3. Methodology

The methodology of the model, evaluation, and visualization components of the project will be discussed throughout this section.

3.1. Classification Model

The ultimate goal of training the classification model, namely the SVM, was to fit a function upon the Hotel Review data set, in order for it to make predictions on unseen data. In order for the model to be able to fit the data properly, the data first had to be preprocessed. Once the data was preprocessed, the model had to be trained, and its various hyperparameters needed to be tuned. Only then was the model prepared for evaluation on a subset of the data used for testing. The data was imported using Dask [16], preprocessed using NLTK [17], and trained and tuned with Scikit-Learn [18]; these processes will be discussed.

3.1.1. Importing the Data Set. Dask is a Python library which emulates data processing libraries such as Pandas and Scikit-Learn, with the added feature of processing big data in a parallel method. When a data set is imported with Dask, it is through a lazy loading process, which will read the overall structure and data types of the data set, but not actually the data. This is in order to import data sets that may be too large to hold on a machine's memory alone. The data sets themselves are then structured in partitions, which are held in Pandas data frames. These partitions allow for more efficient processing of data as they can be called and processed on separate CPU or GPU cores than other partitions. This allows for data processing to be much more efficient compared to running it all on one core, while also allowing for the processing and training of much larger data sets than a machines memory may load.

In the implementation of this project, the Hotel Review data set was imported and stored with Dask. The data set itself was quite large, so this feature saved a lot of time and space. This was because the lazy loading nature of Dask did not load any data into memory that was not actively being used. Dask also provided efficient methods for parallel preprocessing of such a large data set.

3.1.2. Text Preprocessing. The input data was originally imported as raw text data, while the label data was reviewer rating scores from 0 to 10. Thus, the data was in need of quite a bit of preprocessing before any model was ready to train with it. The process of cleaning the text data and preparing the labels will be briefly described.

From earlier experimentation, there was found to be a problematic unbalance in the labels of the data set. Out of the 500000 data instances, only around 20000 of them were actually negative reviews. The effects on the model due to this unbalanced data is shown in figure 3. Even if the model predicts only positive, it will still have an accuracy of around 96%, as nearly 96% of the data was positive reviews. In order to balance this data, only the first 20000 positive reviews were processed, along with the other 20000 negative reviews. The first 20000 positive reviews were taken by filtering out data by date, as Dask does not keep track on data instances since it is not actively loading them into memory. Overall, this reduced the size of the data set to combat time and space constraints presented while training the model. As shown in the Background section, the reviews had statistical significance while being represented as a population, and sampling 20000 entries from a data set with no defined order should hold that significance. This method was more robust than any up-sampling methods, as the data remained original and unique across training and testing data sets. This would allow the model to avoid potentially becoming overfit from duplicate data or cross contamination.

```
Accuracy Score : 0.9625846999901797
Precision Score : 0.9625846999901797
Recall Score : 1.0
F1 Score : 0.9809357017763323
Confusion Matrix :
[[   0  381]
 [   0 9802]]
```

Figure 3. The trained SVM classifier on a data set of 500000 text reviews, which only had 20000 negative reviews.

As mentioned, the text data itself was partitioned into positive and negative reviews from the same reviewer, as separate data frame columns. If no negative review was given, in its place was "No Negative", as was the case for positive reviews. This text was combined into one column, filtering out any "No Negative" or "No Positive" strings. These strings were filtered out as they provided unwanted noise to the data as very positive reviews would tend to have many "No Negative" strings, and vice versa. This would cause the word "Negative" to accumulate within very positive reviews, and vice versa. This was an issue which could throw off the model as it was fitting the data based on text in reference to the polarity of the review.

This combined text would then be cleaned with a few NLTK functions. The data was all set to lowercase, where it was lemmatized and checked for stop words. Lowercase conversion and lemmatization were implemented in order to standardize words with the same meanings. This is because in vector space, words with the same meaning should be represented with the same vector. For example, "running" and "Run" have the same meaning in the context of the model, but may have different vector representations. Lowercase conversion and lemmatization would convert those words to a string which resembles "run", thus setting their vector representation equal. The stop word removal is required as

it removes words without meaning to the sentiment of the review, only its structure. This further removes pointless words from the data set. For example, in terms of sentiment analysis "I had a great time" and "great time" mean essentially the same thing, but the latter is quicker to process and has less noise overall.

The text data was finally projected onto vector space, as word vector representations. This was in order for the model to be able to interact with the data in some dimensional space. This word vectorization was undertaken by Scikit-Learn's TfidfVectorizer, in order to further filter out noise (through idf) while converting the text data to numeric vector representations.

The intended label data was imported as a review score which ranged from 0 to 10. In order to refine this data to be trained using a classification model, a score of 5 to 10 was converted to a 1, representing a positive review. All other review scores were converted to a 0, representing a negative review.

3.1.3. Hyperparameter Tuning. In order to properly optimize the label prediction of the classification model, it is important to test a wide range of hyperparameters. In terms of the SVM classification model, these hyperparameters included the C value and choice of data shrinking. Table 4.1 shows an example of the various hyperparameters that were tested during implementation of the model.

The hyperparameters were tested using a cross validation grid search method, known as Scikit-Learn's GridSearchCV. This function will executed an exhaustive search in parameter space, with the given hyperparameters. For example, if 0.01, 0.1, and 1 were passed in as input parameters for the C hyperparameter, a model will be trained with cross validation over each set of parameters. This function allows researchers to search a wide range of hyperparameters, with the ultimate goal of optimizing the given model.

In order to further increase the efficiency of the grid search model, another parallel computing library known as Scikit-Learn's joblib was implemented. This works with Dask in order to train a model in parallel, using multiple cores on a machine. This effectively increased the efficiency of the training, which was extremely valuable for a model fitting on very large data sets.

3.2. Model Evaluation

As mentioned, after training the model was evaluated in terms of its accuracy, precision, recall, and f1 score. These metrics were implemented using Scikit-Learn's various metric libraries. Tables 4.2 and 4.3 in the Results section describes this evaluation further.

3.3. Visualization

Using Dash [19], two map visualizations were implemented. The data was imported in a separate work space using Pandas [20], and figures were rendered using Plotly [21].

The first map visualized the nationality of the reviewers, using the data set's "Nationality" column. Instances were sorted by nationality, counted, and their review score's were averaged. This data was then shown on a map of the world, using a color scale based on the average review score from that nation. A user is able to select which hotel they are viewing, and the month and year of the review data. Lastly, there is a tool which displays additional information about the reviewers when a user hovers over an eligible country. This map, as well as its features, is shown in figure 4.

The second component visualized each individual hotel's review score, in context of its location on a map. As the hotels from this data set are all European, the figure displays only a map of Europe. Each point on the map represented an individual hotel, whose colours were based on an average review scale. This data can be selected in the context of the month and the year. This map, as well as its features, is shown in figure 5.

3.4. Example Data Instance

In order to further clarify, this section will describe the process that single instance of data goes through in the context of the above descriptions. Figure 6 also shows this process visually. The example data instance will be said to have the following values. Review date = 03/15/2017, hotel name = "Prince George", average score = 7.7, reviewer score = 8.9, positive review = "We really enjoyed our stay, will come back next year!", negative review = "No negative", reviewer nationality = "Canada", lat = 75.4, lng = -9.2.

3.4.1. Model Example. In order to be trained or tested with the model, the data must be preprocessed. First, as the reviewer score is greater than 5, the instance's polarity value will be set to 1, for positive. If the text is chosen to be used by the model during data balancing, its full text value will be set to a combination of the positive review and negative review. In this instance, negative review is discarded as it is set to "No negative". Stop word removal and lemmatization then occur on the full text, resulting the its value being set to "really enjoyed stay, come back next year". The last bit of preprocessing involves converting this full text to a word vector. The resulting word vector (rounded to the second decimal) is [0.44, 0.27, 0.30, 0.34, 0.51, 0.40, 0.35]. This vector and the polarity value will be used with the model.

If the data instance is used for training the model, it will be processed by a fresh SVM classification model. This model will fit to the vectors in their dimensional space, in order to distinguish between positive and negative reviews. The label which the model will check throughout training is the polarity value. If the data instance is used for testing the model, the model will simply predict its polarity value, based on its trained function. This prediction can then be related back to the instance's true polarity value in order to measure the performance of the model.

3.4.2. Visualization Example. During visualization of the reviewer data map, the data instance will be grouped with

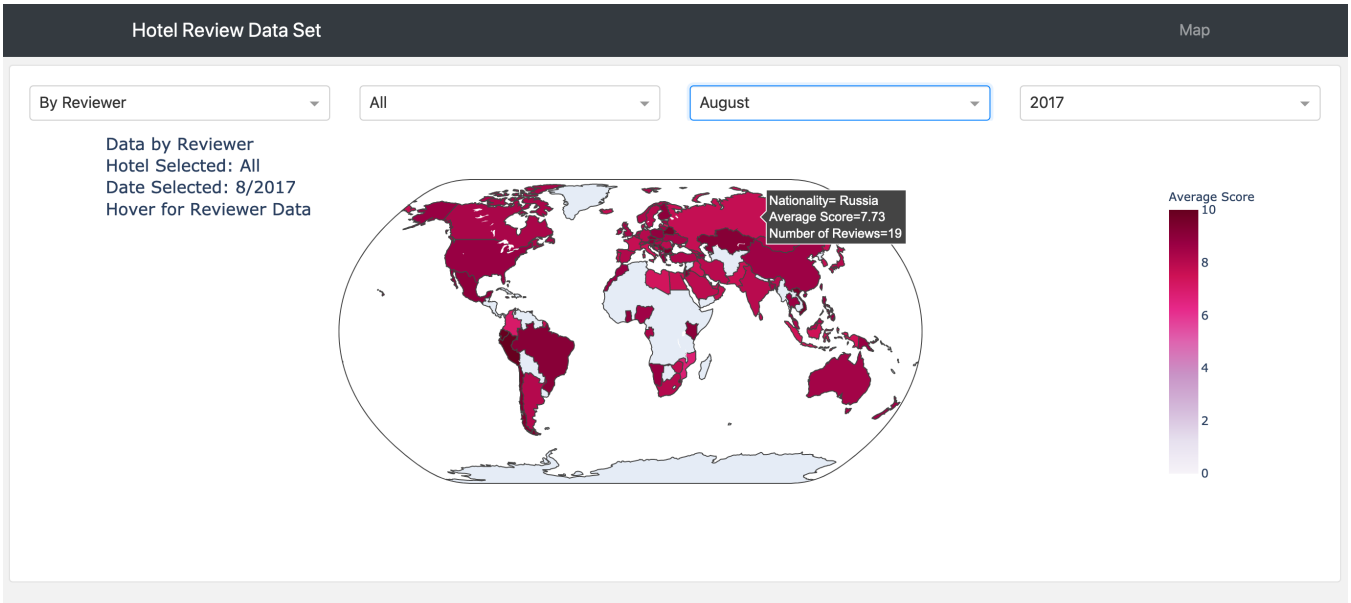

Figure 4. First map component, data by reviewer, from Dash visualization web application. The hover tool is displaying the selected country (Russia), the average review score from reviewers (7.73), and the amount of reviewers (19). This data is from August, 2017 for all hotels in the Hotel Review data set.

other data instances of the same nationality, in this case Canada. The count and average reviewer score of these instances will be calculated. These values, along with the nationality itself, will be displayed on a map similar to figure 4. During visualization of data by hotel, the data instance will be grouped by other instances of reviews on the same hotel, in this case "Prince George". The hotel will be displayed on a map based on its lat and lng values, in this case 75.4 and -9.2 respectively. The colour on the map of this data instance will be determined by the hotel's average review value.

It should be noted that both visualization can filter the date of the data shown on the map. For this, the instances only displays when the user selects the month March, and the year 2017; as these were the review year and month values. The reviewer data map can also select which hotels to display; if the hotel name of the data instance exists within the month and year selected, it will be displayed on the map.

4. Results

This section will briefly discuss the results of the project in terms of the discussed metrics.

4.1. Classification Model Results

As previously discussed, hyperparameters were tuned with the intention of optimizing the SVM classification model using Scikit-Learn's GridSearchCV. The results from this tuning are shown in table 4.1. As shown here, the hyperparameters which resulted in the most robust classification model was a C value of 1, with a linear kernel and parameter shrinking.

Table 4.1: A glimpse of the GridSearchCV test results for each hyperparameter tested. This instance of the search tested different C values and parameter shrinking with the classification model. All instances of the model used a linear kernel function.

C Value	Shrinking	Mean Score	Rank
0.01	True	0.753	7
0.01	False	0.753	8
0.1	True	0.837	3
0.1	False	0.837	4
1	**True**	**0.839**	**1**
1	False	0.839	2
5	True	0.823	5
5	False	0.823	6

The selected model was tested using the test data from the data set. This data included 30% of the overall preprocessed data set. The results from these tests on the tuned and trained model are shown in table 4.2. A confusion matrix of the prediction results is shown in table 4.3, in regards to the models true and false positives and negatives. As there are no random elements in the SVM classification model, nor in the data sampling, statistical significance is not required to be shown for the results. This is because the same results are yielded every time due to this lack of randomness.

Table 4.2: Testing results for SVM classification model chosen by GridSearchCV on Hotel Review task. The model had a C value of 1, with a linear kernel function and parameter shrinking.

Accuracy	Precision	Recall	F1 Score
0.835	0.845	0.794	0.819

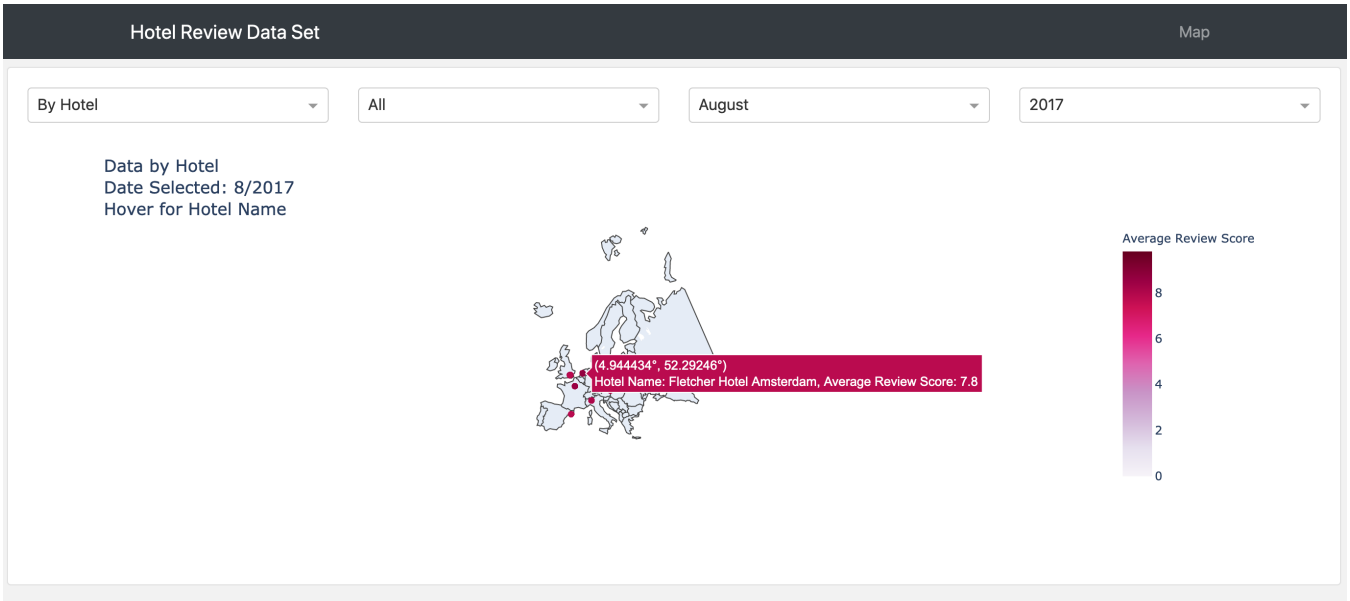

Figure 5. Second map component, data by hotel, from Dash visualization web application. The hover tool is displaying the selected hotel's coordinates (lat: 4.94, long: 52.29), its name (Fletcher Hotel Amsterdam), and its average review score (7.8). This data is from August, 2017.

Table 4.3: Confusion matrix of prediction results by SVM classification model chosen by GridSearchCV on Hotel Review task. The rows represent the ground truth values, and the columns represent the predicted values.

	"Negative"	**"Positive"**
"Negative"	5855	854
"Positive"	1205	4636

4.2. Visualization Results

Although no metrics are used to grade the visualization, its development did result in a working web application. This web application can be downloaded and run through the file Index.py in a PyCharm environment according to the requirements.txt file provided in the dash-app folder of the project. The dash-app folder can be found in the final_code.zip submission. The app will only run on Dash version 1.16.3, so if all else fails, a demonstration of the visualization web application can be found here: https://youtu.be/b7yuEy1-4hs.

5. Discussion

This section will analyze the results found throughout the project, as well as its limitations. This discussion will be in reference to the model and its evaluation, as well as the visual component of the project.

5.1. Model and Evaluation

This section will provide a reflection on the strengths and weaknesses of the SVM classification model, and its evaluation discussed throughout this paper. According to the results, the model performed with an accuracy of around 83.5%. This means that the model is predicting the correct label on the data 84% of the time. This can be believed as an accurate measure of the models intentional performance, due to the results shown through the other metrics used. These metrics were the precision, recall, and f1 score; all of which are used to measure the quality of the accuracy itself.

The precision, at 0.845, can be interpreted as measuring how many instances of a class according to the model are truly of that class according to the ground truth values. Having a relatively high precision indicates that the model is not frequently falsely assigning data to the wrong class. The recall, at 0.794, can be interpreted as measuring how many instances of each class are actually being correctly classified by the model, according to the ground truth values. Having a relatively high recall indicates that each class is frequently being accurately predicted. The f1 measure, at 0.819, can be interpreted as a weighted mean between precision and recall. It can be thought of a standard score to represent the model which considers both precision and recall. This standard provides an important tool which can be used to measure the model against others implemented for the same task. Having a high f1 measure, indicates that both precision and accuracy are relatively high.

Overall, it can be confidently said that the implemented SVM turned out to be a robust model given the project. This is considered robust despite not being able to work at 100% accuracy. Although a human could probably classify the text with close to a 100% accuracy, it would take them a substantial amount of time compared to the model. Given that the goal of the model is to be able to classify the polarity of hotel reviews efficiently, a high precision value is ideal, as observed. This is because in order for the model to be

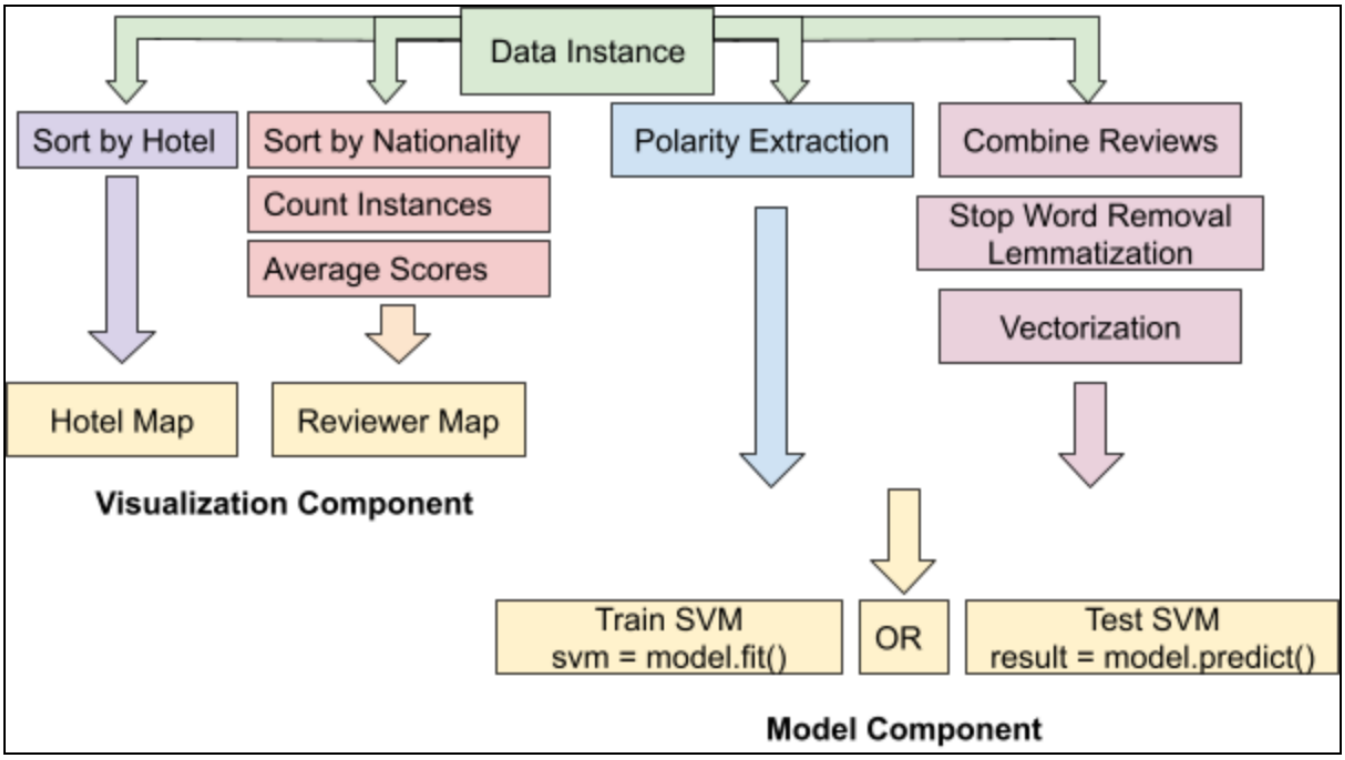

Figure 6. Work flow diagram, following a single data instance through the model and visualization component of the project.

effective, it needs to consistently predict each class correctly. As mentioned, a higher precision indicates how accurately the model predicts each class. As hotel management will not be looking at these reviews, they will have to be able to trust the results. Additionally, recall is less important for this solution as false negatives are not crucial to avoid. In other words, there is enough text data that there can be a few reviews classified as the wrong polarity; it is not life or death.

There are two pressing implementation limitations which improvement could substantially increase its robustness for future use. These limitations are the lack of range in labels in the data and the way the model learns from the language.

A disadvantage of the style of learning with this specific data set is the current binary representation of polarity. The implemented SVM acted as a binary classifier as polarity was simply represented as ”Positive” or ”Negative”. An issue with this binary representation is that it must sacrifice insight into the range of polarity. The polarity of each review is originally presented as a review score, ranging from 0 to 10. This indicates that there is an alternative polarity representation which provides more depth to each review. For example, this allows for the difference of a very positive review, with only a partially positive review. The current implementation of the model does not possess the ability to classify these labels other than just ”Positive” and ”Negative”. This is a limitation to the overall goal of the project; with a deeper range of polarity, a hotel management may be able to identify more important reviews while processing this data.

While the first limitation related to the data representation, the second limitation relates to the model’s learning itself. Supervised learning involves fitting data based on labels. Thus, a model learns representations of each label based on the data provided. The supervised learning model is not learning actual coherent human language, rather fitting a function around a subset of strings within the human language. Although it was shown that the reviews were likely to not be contradictory of their scores, this does not apply to several patterns in the human language. An example of this is shown in figure 7, when a negative review is predicted to be positive. It is hypothesized that this is because the model has weighted the word ”great” towards being labelled as ”Positive”. Natural language is very complex; based on the training data provided, a limitation of the model is that it is not capable of learning such complexity throughout training. This leaves the model with a very loose understanding of the language, and may be detrimental to the robustness of the model, depending on the contents of unseen input.

5.2. Visualization

As discussed in the Introduction and Background section, the goal of the visualization component was to visualize the Hotel Review data on maps. This presentation would grant hotel management a fuller insight on the demographics, cultures, and nationalities of the guests visiting and reviewing their hotels.

```
input_text = 'The service was not great'
pre_process = remove_stop_lemmatize(input_text)
process = vectorizer.transform([pre_process])

result = grid_search.predict(process)

if result == 1:
    print('Input: ' + input_text)
    print('Polarity: Positive')
else:
    print('Input: ' + input_text)
    print('Polarity: Negative')
```

```
Input: The service was not great
Polarity: Positive
```

Figure 7. Potential limitation of model depending on input. The model is not learning actual coherent human language, rather fitting a function around a subset of strings within the human language.

Overall, the first map visualization can be very valuable in the right hands. As mentioned, showing this data on a map allows hotel management to understand the data in a much more compelling way compared to reading a .csv file. They are able to view where their reviewers are coming from, and how those reviews change depending on where that is. This may allow them to gain further insight into accommodating for different cultures and nationalities. The hotel selection tool will benefit hotel management as they can view the review data of their own hotel, other hotels, and all hotels at once. The month and year selection was implemented for the efficiency of the web application. Otherwise, Plotly takes a very long time to load and render each 500000 data instances.

The second map visualization provides another valuable tool for hotel management to analyze while viewing their own hotel scores. These hotel scores can now be analyzed with direct reference to the location of the hotel, and can be compared to other hotels near and far. This data may be used in order to find competition within a district of a city, and may lead to discussion of how to improve hotel performance compared to that competition. As with the first map, the month and year selection was chosen as it would make rendering the data on the map much faster.

The most pressing limitation of the visualization component of the project is the Plotly rendering speeds. Without the month and year filtering, the maps take far too long to load for a web application. This is problematic as more insights may be gained by viewing general reviews as a whole in the context of just location.

Time constraints on the project was also a limitation to the dashboard. If more time was available to further develop this component, another page would be added to the web application. This page would allow for hotel management using the application to enter in reviews of their own, and have them classified as positive or negative. This would allow for users to test out the model in order to gain a better understanding of what may determine a reviews polarity according to the model.

6. Future Work

With reference to the limitations described in the Discussion section, future implementations to improve the overall project will be listed throughout this section.

6.1. Classification Model

The goal of the model was to help hotel management teams in efficiently analyzing review data in order to further improve their services. This includes gaining insights on which reviews are the most pressing. For example, out of all of the negative review, the most extreme negative reviews may deal with issues such as harassment, abuse, or racism coming from staff; while partially negative reviews may just complain about the swimming pool being too crowded. It is important that a hotel management is able to differentiate between reviews at this level, in order to identify their most pressing issues. As discussed, the current implementation of the model lacks a depth from the ranges of reviews, as the only labels are ”Positive” and ”Negative”. Future work of this project may include adjusting these goals to a range of polarity. For example, this range may be ”Very Negative”, ”Partially Negative”, ”Mixed Review”, ”Somewhat Positive”, and ”Very Positive”. Providing this is important in order to further refine the review analysis, but a binary classification model will not suffice for learning with that many labels. The Background section discusses an extension of the SVM, a multi-class SVM. As shown in the Results section, the SVM proved to be a robust model given this data set, thus it is hypothesized that a multi-class SVM will be an effective model to implement a model that can handle this deeper range of polarity.

The model itself was shown to have issues related to understanding a deeper level of natural language given the training data provided. An obvious solution to this is to simply provide a larger data set. The issue with this solution will increase training times significantly as well as not guarantee any further understanding of language. A solution to this issue is to investigate potential pretrained classification models. Many large API development companies have released models which have a broader understanding of the human language. This may prove useful for this project, and if the investigation yields useful results, the direction of the future work will shift to performance testing of such pretrained models based on the data set.

The final issue which was discussed in the Preprocessing subsection of the Methodology section was the unbalanced nature of the data. Ultimately, this prevented the use of all of the data as a mere fraction were negative reviews. Investigating methods of balancing this data is considered as future work. Potential solutions include changing the threshold for what is considers negative, in order to increase the amount of negative reviews. This will include a deeper investigation into where this threshold occurs in reviews naturally, in order to further tune this hyperparameter. Of course, another solution is to simply generate more negative reviews. If this route is taken, reviews may be generated

either through data collection of other related reviews, or by implementing a probabilistic model to generate additional review documents.

6.2. Visualization

The final suggestion of future work to note relates to the visualization web application of the project. The Discussion section mentions the original intention to provide some sort of model interaction component as a page on the web application. This feature will simply include a text box, which a user can enter in a review. Upon submission of this review, the classification model will be called to predict the polarity of the text. This polarity will be displayed next to the input text after it is returned. This feature of the web application will provide a hotel management with the ability to assess reviews, in order to gain a deeper understanding of how polarity is assigned to a review. This will also be beneficial to the model, as it will be able to continue to learn with the users input; potentially reducing the risk of data drift effecting its performance.

7. Conclusion

The goal of this project was to develop tools for a hotel management to efficiently analyze hotel review data. This would allow them to gain insights on these reviews instead of having to sort through thousands of text documents and review scores. A supervised classification model, known as a support vector machine, was implemented in order to perform sentiment analysis on a data set of hotel text reviews. The polarity (”Positive” or ”Negative”) was extrapolated in each review based from a corresponding review score. The model was evaluated using accuracy, precision, recall, and f1 measure to which the model scored 0.835, 0.845, 0.794, and 0.819 respectively. These scores deemed the model as a robust method of efficiently processing thousands of review texts.

A visualization component was also implemented to the project, which allowed users to view hotel reviews, and reviewer data in the context of time and space. This visualization's was able to show off data in ways not possible by simply parsing through a .csv file.

Based on the limitations of this project, future work includes implementing a multi-class SVM to broaden the range of polarity, finding a solution to the model's lack of understanding of natural language patterns, further balancing the data set, and add an interactive model feature to the visualization web application.

References

[1] “515k hotel reviews data in europe,” 2017. [Online]. Available: https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe

[2] S. Matwin, “c1_v2,” *Presentation Slides for CSCI 6515 Lecture*, 2020.

[3] “Booking.com.” [Online]. Available: https://www.booking.com/

[4] B. Illowsky and S. Dean, *De Anza College: Introductory Statistics*, 1st ed. Open Education Resource (OER) LibreTexts Project, 2020.

[5] B. Liu, “Sentiment analysis and opinion mining,” *Presentation Slides for Tutorial at Univesity of Illinois*, 2011.

[6] M. K. Jiawei Han, Jian Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2011.

[7] F. Kapadia, “Sentiment analysis: Android applications,” *B. Thomas Golisano College of Computing and Information Sciences*, 2017.

[8] H. Hamdan, P. Bellot, and F. Bechet, “Supervised methods for aspect-based sentiment analysis,” *SemEval2014*, 2014.

[9] D. Berger, “Introduction to binary logistic regression,” *http://wise.cgu.edu/*, 2016.

[10] N. Zainuddin and A. Selamat, “Sentiment analysis using support vector machine,” *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, 2014.

[11] S. Ji and Y. Xie, “Logistic regression: From binary to multi-class,” *Texas A&M University*.

[12] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, 1995.

[13] T. Evgeniou and M. Pontil, “Support vector machines: Theory and applications,” *Machine Learning and Its Applications, Advanced Lectures*, 2001.

[14] baeldung, “Multiclass classification using support vector machines,” 2020. [Online]. Available: https://www.baeldung.com/cs/svm-multiclass-classification

[15] P. L. Lanzi, “c2_v2,” *Presentation Slides for CSCI 6515 Lecture*, 2020.

[16] “Dask.” [Online]. Available: https://dask.org/

[17] “Natural language toolkit.” [Online]. Available: https://www.nltk.org/

[18] “Scikit-learn.” [Online]. Available: https://scikit-learn.org/stable/

[19] “Dash.” [Online]. Available: https://dash.plotly.com/

[20] “Pandas.” [Online]. Available: https://pandas.pydata.org/

[21] “Plotly.” [Online]. Available: https://plotly.com/