BSTAT 625 Final Project Report

Ben Brennan, Mandy Meng Ni Ho, & Tahmeed Tureen

HuiJiang625, LLC

Introduction

Background

The integration of the fields of Machine Learning and Natural Language Processing (NLP) has been growing tremendously in the modern era. Powerful and flexible computers have allowed researchers in the field of NLP to apply statisical methodologies to large, unstructured data in the form of texts to create software applications or draw meaningful insights. Successful NLP applications such as the autocorrect feature on messaging apps, search engine suggestions, and question answering machines (i.e. `Siri`, `Cortana` etc.) have increased the field's popularity. Insights-driven NLP applications such as topic labeling (i.e. classifiying a text to a specific domain), spam and fraud detection, and sentiment analysis (i.e. classifying the tone of the text's author) have also been growning in popularity. As a result, many companies have started to invest in NLP to analyze the large amounts of textual data which they own at their disposal.

Motivation

In this project, we spin a scenario where we are data science consultants at a world renowed consultancy firm, **HuiJiang625 LLC**. The primary clients of this consultancy group are airline agencies (i.e. Delta, American Airlines, & JetBlue etc.) who are looking to gather some insights from the flight reviews they receive from their customers everyday. The marketing teams at these agencies would like to identify customers who are most likely to recommend their airline to friend(s) or family member(s) based on their review of the flight. This would allow the teams to optimize their marketing strategy by making recommend-friendly customers their priority when sending out incentive-based offers such as gift cards, sky miles, or vouchers etc. in return for a recommendation. For example, if the marketing team knows that Tom is more likely to recommend than Sarah then the marketing team could first reach out to Tom.

With this objective in mind, our team is tasked with building a NLP-driven statstical model that takes in input of a customer's textual review (in English) and outputs whether or not that customer will recommend the airline. A secondary objective of this project is to provide an user-friendly application that allows the user to input a customer's review and immediately receive the answer of whether or not they will recommend. The purpose of this app is to eliminate the need for statistical knowledge in the user as well as mask the mathematical mechanisms behind an attractive dashboard.

Dataset

Raw Data

The dataset for this project is provided by Skytrax, a United Kingdom based consulting group that specializes in airline and airport reviews. This dataset consists of 41,396 observations with attributes such as customer review regarding flight, customer's country of origin, customer's ratings (i.e. overall, seat comfort, cabin staff rating etc.), customer's seat type (i.e. Business, Economy etc.) and finally a variable that represents whether or not a customer had recommended the airline after their review. The customer recommendation (variable name : `recommended`) is a binary variable and is the primary outcome of interest in this project. As discussed earlier, the objective of this goal is to create a model that can predict whether a customer will or will not recommend an airline. Therefore, the attribute holding the customer reviews is the main dependent variable in our analyses (variable name : `content`).

Data Processing

Data preprocessing for this project required two steps. The first was to include meaningful covariates that were not included in content. In essence, we needed to answer the question: Is there anything outside of the review (`content`) that is significantly associated with whether or not an individual is going to recommend the airline? After answering this question, we are then able to understand how to process the non-review part of our data to make it efficient in our alogrithms.

To answer this question, we used simple logistic regression to assess significance of certain covariates on the reccomendation. First, we excluded all *rating* variables (i.e. overall rating. wi-fi rating, food rating) as we felt the project would be too easy if we included those variables. We mainly focused on country of interest, type of traveller and what class the traveller was traveling in. We found, in our data set, that solo travellers were more likely to recommend the airline - but there was also about 39,000 missing values for this covariate, so it was not included. We found that a person from the US, as opposed to someone not from the US, is much less likely to recommend an airline. Furthermore, we found that if a person was flying first class or business class, they were more likely to recommend the airline compared to someone flying economy. Thus, we extracted two bits of information from the attributes of the customer - a binary indicator of whether they were in business class or first class vs. not and a binary indicator of whether a passenger was from the US vs. not.

The second question we sought to structure our data in order to answer was: how does a customers' review affect their recommendation? This question requires us to process the `content` column of our dataset in a reasonable way. We did this using the `tidytext` package in `R`. Essentially, we split the content of each review up into singular words, removed words that were not associated with sentiments from the *bing* list of words (`tidytext::get_sentiments('bing')`) and then constructured a document-term matrix (DTM). This matrix contains columns that are 1 if a word is present and 0 if that word is not present. For instance, a customer with a review such as "This was a great flight" would be split into words, and then "great" would be the only word that existed in this review. This customer would then have a row of data in the DTM that consisted of zero for every other sentimental word that existed in other reviews, and a 1 in the column correspoding to 'great'. In the end, our document term matrix is a 41,117 by 3,513 matrix, using about 1.1 GB of memory.

By joining our attribute predictors to our DTM, we obtained a dataset that was appropriately structured such that we were able to explore and attempt to answer the questions above.

Methodology

The dataset provides both the input of the model as well as the output, therefore, we consider four supervised learning algorithms to create our classification model: (i) Naive Bayes Classifier, (ii) Logistic Regression, (iii) Support Vector Machines, and (iv) Random Forest. The Naive Bayes Classifier is the most easy-to-use algorithm when it comes to textual data classification (citation 1). It is a probabilistic classifier based on the Bayes Theorem and we use this as our baseline classifier. Logistic Regression is a regression-based model that allows us to model the probability of a customer's recommendation (citation 2). For the Logistic Regression, we consider a Lasso Logistic Regression without an intercept and one with an intercept. Applying the lasso shrinkage, we get to shrink the effects of low-importance words in our regression model to exactly zero. Support Vector Machine (SVM) is a discriminative classifier that uses hyperplanes in Euclidean spaces to separate datapoints into separate classes (citation 3). For our SVM, we used **hyperparameters**. Random Forest is an ensemble tree-based method. We separated the dataset as X% Training and Y% Test. The models were then trained on the training set using `R` (packages:). To evaluate and compare the models, we used Recall, Precision, and F1 Scores. As for the interactive dashboard, we planned to develop a Shiny Application using the `shiny` package in `R`.

Statistical Models

- **Naive Bayes**

Let R be the event that a customer recommends the airline.

$$P(\text{R} \mid \text{"}Great\ Time\text{"}) \propto P(\text{"}Great\ Time\text{"} \mid \text{R}) = P(\text{"}Great\text{"} \mid \text{R}) \times P(\text{"}Time\text{"} \mid \text{R})$$

- **Logistic Regression**
- **Logistic Regression (No Intercept)**
- **Support Vector Machine**
- **Random Forest**

This shit failed so we legit didn't use it.

Computational Challenges

We attempted to train our models on our personal laptops (both Mac and P.C.). However, due to the large amounts of data particulary the large number of dependent variables (X_{word_i}) we were either not able to succesfully train our models (timeout or memory errors). Therefore, we subset the dataset to a smaller number of reviews and re-trained our models to assess if our code was running appropriately.

Therefore, we opted to use the University of Michigan Biostatistics Cluster for our model training. For Logistic Regression

Results

Statistical Results

Based on the predictive analysis on the test set (n =), we found that Support Vector Machines had the highest accuracy scores in terms of all three evaluation metrics. Logitic Regression (with and without Intercept) had very similar scores. The Naive Bayes Classifier performed the worst, however, it is our baseline and we expected it to perform poorly due to its underlying modeling assumptions (see Citation). These results are summarized in the following table.

Model	Time (sec)	Recall (%)	Precision (%)	F1 Score (%)
Naive Bayes	0	0	0	0
Logistic Regression w/ Lasso	0	0	0	0
Logistic Reg. w/ Lasso (No. Intercept)	0	0	0	0
Support Vector Machines	0	0	0	0

Shiny Application

The Shiny application can be viewed by running this command on the `R` console: . It is important to note that, the use of this app first need to install the `shiny` package on their personal machine.

Discussion

Contributions

- Ben Brennan :
- Mandy Meng Ni Ho :
- Tahmeed Tureen : Helped formulated the research problem, identified statistical methods and evaluation metrics appropriate for text classification, helped find NLP tutorials for `R`, helped write `R` scripts, wrote job scripts (`slurm` files) for the cluster, helped write the final report, and collaborated with teammates via in-person meetings, `GitHub`, and online messenging

Repository

The work directory of this entire project has been published on a GitHub repository, which can be accessed via the following link: **CLICK HERE**