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. Insight-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 in the industry 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 daily. 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 as 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 (i.e. marketing team) to input a customer's review and immediately receive the answer of whether or not they will recommend the airline. The purpose of this app is to eliminate the need for statistical knowledge in the user as well as mask the mathematical mechanisms of the models 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's textual review regarding their 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 that represents the customer's textual review is the main independent 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. By 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 a linear kernel and allowed for shrinkage. Random Forest is an ensemble tree-based learning method that constructes multiple decision trees which can collectively used for classification tasks (see Citation). For all four of the models, we added two adjusting variables: US Citizen (1 = yes, 0 = no) and Flight Class (First, Business, and Economy). The discussion of how these algorithms work from a mathematical perspective is beyond the scope of this project report, however, we provide references at the end of this document. To assess and compare these models, we take a training-testing approach with recall, precision, specificty, and F1 scores as our evaluation metrics. The statistical computing was performed using `R` (packages: `caret`, `e1071`, `randomForest`, `glmnet`). We also compared the run time for each of the model's training using the `system.time()` functionality. As for the interactive dashboard, we use the `shiny` package in `R` to develop a Shiny application.

Challenges

The data processing aspect of this project was very smooth and we were able to do all of it on our local machines (Mac & P.C.). However, we ran into computational issues when it came to training the statistical models. We first attempted to train the models in `R` on our local machines. All of the models were either taking a significant amount of time to run or breaking due to timeout and memory errors. Therefore, we used the `doParallel` package to run our `R` scripts in parallel time. By doing this, we were able to run some of our scripts only on a subset of our full training set (n = 2,000). However, the computational time was still very demaning: SVM took more than 20 hours, Logistic Regression never converged, Naive Bayes ?, and RandomForest did not finish.

Therefore, we opted to use the University of Michigan Biostatistics Cluster for our model training. For all of the models, we allocated 10 GigaBytes (GB) of data with one core on the cluster. For Logistic Regression, we allocated a total of 45 mins on the cluster and both scripts used about 7 GB of memory (peak usage). For SVM and Naive Bayes, we also allocated 45 mins and found that they used both 10 GB and 8.7 GB respectively. We tried training the Random Forest on the cluster as well with combinations of 10GB, 15 GB, 45 minute, 2 hours, and 12 hour allocations. However, we were unsuccesful in training the model due to timeout and memory errors. Since we were on a time constraint and the other models were showing satisfactory results, we opted to stop training the Random Forest. It is important to mention that the `R` scripts submitted to the clusters incorporated the `doParallel` package's functionality. Each model had its own script which were run individually so that they could be run around the same time on the cluster. The cluster significantly improved our work flow and we were able to achieve satisfactory results.

Results

Statistical Results

Based on the predictive analysis on the test set (n =), we found that Naive Bayes performed the best in terms of Recall and Logistic Regression with Intercept performed the best in terms of Precision. However, since F1 Score is the harmonic mean between recall and precision, we use it to identify SVM as our "best"" performing model. SVM had the highest accuracy scores in terms of the F1 Score. Logitic Regression with and without Intercept had very similar scores. The Naive Bayes Classifier performed the worst, however, it is our baseline model and we expected it to perform poorly due to its underlying assumptions about independence of its covariates (see Citation). As for computational time, the Logistic Regression models (`glmnet`) were trained the fastest whereas Naive Bayes and SVM took a much longer time (`e1071`). These times were calculated from the cluster jobs. The results are summarized in the following table.

Model	Time (sec)	Recall (%)	Precision (%)	F1 Score (%)
Naive Bayes	1842.39	99.91	53.79	69.93
Logistic Regression w/ Lasso	34.752	72.41	88.63	79.7
Logistic Reg. w/ Lasso (No. Intercept)	40.403	75.83	87.8	81.38
Support Vector Machines	1565.65	87.34	85.21	86.26

Shiny Application

We built our Shiny application using the SVM model as our classifier embedded behind the dashboard. This means that the dashboard makes predictions using the SVM model. The Shiny application can be viewed by running this command, `shiny::runGitHub('airplanes','benbren', subdir = 'shiny')`, on the `R` console. It is important to note that the `shiny` package must first be installed before this app can be run. We provide an illustration of what the dashboard looks like below.

Discussion

There are several ways we could improve our statstical analyses in the future. In this first iteration of the project, we did not attempt hyperparameter tuning for our models. For example, we did not use cross-validation to identify the "best" tuning parameter, λ, that minimizes the loss function for Lasso Logistic Regression. Hyperparameter tuning could potentially improve the prediction scores we see especially for Logistic Regression and Support Vector Machine. We could greatly improved our predictive model by using Deep Learning algorithms such as Feed Forward Neural Networks which are very popular in the field of computational NLP. As for data processing, we only explored the DTM approach when feature engineering our covariates. In the future, we would also like to explore other feature engineering approaches such as TF-IDF (term frequency-inverse document frequency) and Word Embeddings. This project was a challenging project that we formulated and enjoyed it very much. We hope to continue working on it after the course so that we can publish the Shiny application.

Contributions

- Ben Brennan : Helped formulate the research question as well as collect, manipulate and analyze data in both the preprocessing and model development stages. Actively contributed to the writing of the training scripts, as well as the intial attribute modeling. Helped with the writing of the final project report, as well as maintenance of the GitHub repository and associated aspects (READMEs, etc.). Collaborated with teammates through meetings, GitHub collaboration and GroupMe conversations.
- Mandy Meng Ni Ho :
- Tahmeed Tureen : Helped formulated the research problem and motivation behind project, identified statistical methodology 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 messeging

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**