CSE 158 Assignment 2 Report

Yuxiao Ran, Haoyu Huang, Ka Ming Chan

December 3, 2019

1 Dataset

In this assignment, our group decides to examine upon the dataset of clothing fit measurements from *Rent the Runway*[1], a website that provides rentals for designer dresses and accessories. The dataset can be found on the website named *Recommender Systems Datasets*[2] provided by professor Julian McAuley from UCSD, with the label "Clothing Fit Data". This dataset has grabbed our interest for examination because in the present days people frequently make purchases of clothing online, and the unfitting of the purchased clothing is a common tragic experience for many people (including us). So, out of curiosity, we would like to investigate to see how do people review their products rented online, and what kinds of reviews do they generally write. In terms of dataset quality, this dataset's wide varieties of possible features based on its column categories, and sufficient rooms for manipulations and transformations to be made on its data also contribute as reasons for our selection.

The dataset consists of 192,462 rows and 15 columns in total. The column names include `fit`, `user_id`, `bust_size`, `item_id`, `weight`, `rating`, `rented_for`, `review_text`, `body_type`, `review_summary`, `category`, `height`, `size`, `age`, and `review_date`. Judging from these categories, it can interpreted that each row of the dataset represents a user's information and reviews on a particular item they had rented. `user_id` represents each unique user, while `item_id` represents the particular item the user reviews.

Below are the plots for the distributions of `rating` (**Figure 1**) and `fit` (**Figure 2**):

Figure 1: distribution of ratings

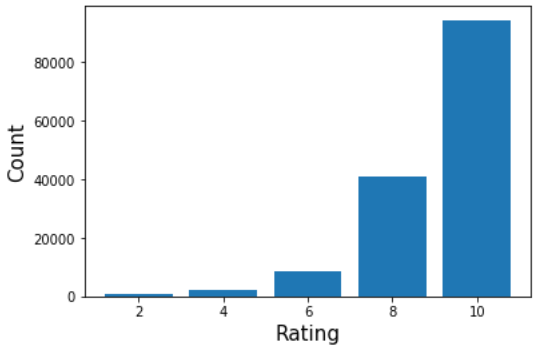
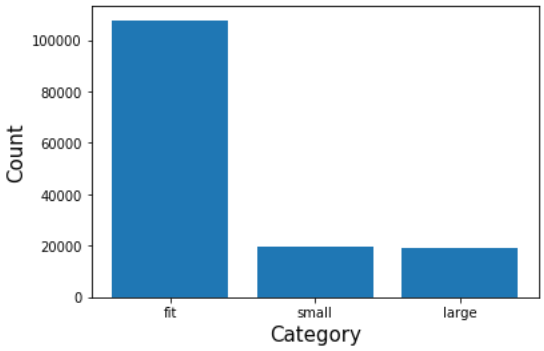


Figure 2: distribution of 'fit' categories



We observe that customers of *Rent the Runway* can only rate their rentals with even number values from 2 to 10, and tend to give high ratings (around 92.3% of ratings are 8 or higher). However, a lower percentage (about 73.6%) of rentals actually "fit". Such a discrepancy make us wonder about the association between the customers' ratings and measured clothing fit based on their body measurements. Perhaps customers tend to give high ratings even though the clothing does not fit on them, due to reasons such as people tend to be kinder on the website reviews, or reviews with lower ratings have not been released by the website. Anyhow, we have reasons to suspect that the "rating" column may somehow be biased.

One way we come up with for examining the review texts is by using a Word Cloud. We implement the `WordCloud` package and apply it on the

[1] `https://renttherunway.com`

[2] Dataset can be found at `https://cseweb.ucsd.edu/~jmcauley/datasets.html#clothing_fit`

`review_text` column of the dataset, and we are able to generate a Word Cloud chart of popular words out of these reviews (**Figure 3**):

Figure 3: wordcloud of reviews

This Word Cloud indicates that the most of the biggest words in the chart, or the most frequently used words in the reviews, are complimentary words such as *fit*, *comfortable*, *compliment*, *perfect*, *great*, *loved*, *beautiful*, etc. This supports our previous finding that the majority of the dataset's ratings and reviews are positive. Meanwhile, words like *small*, *tight*, *large* also appear in the chart quite significantly, possibly presenting items that do not fit. This graph provides an insight into the correlation between `review_text` and the `fit` label, and motivates our baseline model design in the predictive task.

2 Predictive Task

After the exploratory analysis of the dataset, we have two options as the potential appropriate goals that we can perform our predictive tasks on: columns "fit" and "rating". The reason for choosing these two candidates is they both are based upon each individual user's preferences or body shapes, which are described by the other columns of the dataset that we can build our features out of. They can both be predicted as a classification problem, as the input options for "fit" are simple: "fit", "large", or "small". "Rating" can also be converted into simple ternary options, like "high rating", "medium rating", and "low rating". But as mentioned above in the Exploratory Analysis of Dataset section, "rating" is more biased and unbalanced than "fit", and therefore we settled to perform our predictive task on "fit" by using a classifier.

Predicting the fit value for the users in the datasets has values in real-world applications because predicting whether the clothing would fit the a certain user depending on the user's measurements (height, weight, etc.) and the size of the clothing can allow the *Rent the Runway* website to give the users feedback before purchasing, thus reducing product return rate in the process.

Before building features upon the rest of the columns and predicting "fit", the first thing we need to do is to clean the dataset. We drop the rows that originally have missing values in them, because missing values and incomplete data will distract the building of our models, and results will be altered in negative ways. After the manual `.dropna()` process, we are left with 146,381 rows of the dataset to deal with. It is still a large enough dataset (more than 50,000 samples).

The baseline models we implement are to check if `review_text` contains any related words associated with the fit. The baselines are the following:

- **B1**: If the text has (*small*, *tight*, etc.) in it, then the model predicts *small*. If the text has (*fit*, *perfect*, etc.) in it, the model predicts *fit*. Otherwise, the model predicts *large*.

- **B2**: Taking advantage of the fact that our dataset has much more *fit* data than *small* and *large*, the model always predicts *fit*. This is a good baseline because we need to aim for higher accuracy than this.

Because texts from the column `review_text` are really indicative of whether the clothing fits or not, we can use Bag of Words or tf-idf to extract features out of it. `review_text` works better than `review_summary` because it generally has richer information.

User information columns such as body measurements along with clothing size give insights for us on how likely the clothing of that size is going to fit the user of a certain body type. Therefore, the set of specific non-text columns including `bust_size`, `weight`, `height`, `rating`, `body_type`, `size`, and `age` are also used as features. For `bust_size`, we divide it into *bra size* and *cup size*. For `height`, we convert the "feet-inch" measurements into *inch*.

After deciding our features, we are going to first try predicting on the model using text features and

the model using non-text features separately. Then, we will try combinations of the features to compare and determine which set of features works the best. Then we will feed to the features into a multi-class classifier. From class, we know that Naive Bayes, Logistic Regression and SVM are good candidates. We will compare the accuracy rate and choose the best one.

To evaluate our model, we divide the dataset into training, validation, and test sets using an 8:1:1 split. We use the simple accuracy rate (the number of true positives / the number of predictions) to access the prediction validity. We train the model on the training set, tune the hyperparameters on the validation set by picking the set of parameters that yields the highest validation set accuracy rate, and test the models on the test set.

3 Model

3.1 Feature-building Options

First, because we have different options to build features, we first test these options that get fed into a simple Naive Bayes classification model with hyperparameters tuned by the validation set to pick the best option (highest testing set accuracy). In this way, we can pick the best feature-building option without worrying about determining the classification model.

Feature-building options are as follows:

- **F1**: Use only non-text features like `height` as features
- **F2**: Use BoW (Bag of Words) representation to extract 1000 unigrams from `review_text`
- **F3**: Use tf-idf to extract 1000 unigram features from `review_text`
- **F4**: Concatenate both non-text and text features (pick the best between BoW and tf-idf)

Below is the accuracy rate on the test set for each of the the model including the baselines (**Table 1&2**).

Model	B1	B2
Accuracy	0.483	0.736

Table 1: Accuracy of baseline models.

Model	F1	F2	F3	F4
Accuracy	0.675	0.664	0.708	0.717

Table 2: Accuracy of Naive Bayes classification on various feature-building options.

All the models using Naive Bayes perform better than B1 because B1 only looks at a selection of words that indicate the clothing fittingness. However, they are worse B2 because of the simplicity of the model which we will improve upon later. The accuracy of F3 is higher than F2 because tf-idf puts more weight to the words that appear less in the whole corpus but appear in certain texts, picking out the most important words in the texts. This result aligns with what we learned in the class.

Aside from this, we can see that F4 is the best predictor, which aligns with our hypothesis that combining non-text and text features will increase accuracy. However, it does not improve much from F2 and F3. The problem is that we simply concatenated 8 dense numeric features to a sparse array of text features (a lot of 0s). The huge number of text features mask the non-text features, so adding them does not make huge impact on the predictive score. Therefore, our solution is to make these dense arrays sparse by splitting numeric features into categories and use one-hot encoding to encode these categorical features to add on to the sparse array of the text features (**F5**). And the accuracy raised to 0.731 which is better than B2 even with a Naive Bayes model.

3.2 Classification Model

After using F5 as the feature-extraction model, we need to pick the best classification models from the following.

- **NB**: Naive Bayes classification
- **LR**: Logistic Regression classification
- **LSVM**: Linear Support Vector Machine classification

Below is the accuracy rate on the test set for each of the the model (**Table 3**).

From **Table 3**, we can see that NB is worse compared to the other two models. This is because NB wrongly assumes that the entries of the dataset are independent from each other. We can see from

Model	NB	LR	LSVM
Accuracy	0.731	0.804	0.791

Table 3: Accuracy of various classification models on feature-building option F5.

the dataset that a single user can have multiple entries and a single product can be purchased multiple times. LR and LSVM don't have this limitation, thus doing better than NB.

LR and LSVM have similiar scores, with LR doing a little better. This is not surprising because in our previous assignments when trying out these models, the same thing happened.

Thus, we propose to use F5 and LR as our final model (**F5-LR**).

3.3 Optimization

There are several ways to optimize our model.

We can use **stemming** or lemmatization for preprocessing in the tf-idf feature-building stage. Grouping words that have the same stem is a good idea for natural language processing.

We can extract both unigrams and **bigrams** in tf-idf.

We can increase the **max number of features** in tf-idf from 1000.

We can use **grid-search** to find the optimal set of parameters of LR.

4 Literature

4.1 Description of Dataset for Own Study

The experiment dataset for our assignment comes from professor Julian McAuley's research lab in University of California, San Diego, so as it is described on his *Recommender Systems Datasets* website. The data was gathered through collecting users' feedbacks on items from the *Rent the Runway* database. The time range lasts from late 2010 until early 2018. These data were meant for *Rent the Runway* to receive information for individual customer's preferences, as well as reviews on items that might help *Rent the Runway* to improve their item qualities.

4.2 Study Example on Another Similar Dataset

Studies similar to ours had been done in the past, and we have used some of them for references. In an article titled "Analyzing Customer Reviews Using Text Mining to Predict Their Behaviour" [1], written by Sowmya Vivek on Medium, the author tries to analyze a dataset of customer reviews to predict if the customers would recommend a product, mainly using the method of text mining. Her dataset comes from Kaggle and consists of reviews written by customers on a Women's Clothing E-commerce site. As her dataset for examination is similar to ours by that both are collectons of customer reviews, so does her plans and steps in her analysis.

The author splits her analysis into two main components: exploratory analysis of text data, and classification models. The exploratory analysis of text data portion is applied on the review text category of her data, and the classification models for predicting the customer recommendations are built upon her features based on the review text. In terms of her text data analysis, she uses text mining approaches which include two methods - semantic parsing and Bag of Words. She pre-processes the text by converting it into lower-case, removing punctuation, and removing stopwords. Stemming is also applied.

After the pre-processing, the author mentions that the main component of her study is to analyze the differences of key words between positive reviews and negative reviews. She observes the most frequently appearing words in both types of reviews by using bar graphs, word clouds, and polarized tag plot. Through using these three types of graphs, it is clear to visually see the differences of key words in the users' reviews for a recommended product and for a non-recommended product. Other methods are also performed in the article, including simple word clustering, word associations, and the use of n-grams, but as they are quite complicated to implement and irrelevant to our own study, it is not necessary to go deeper in them.

Then, as the exploratory analysis of text data is done, her study moves on to the next stage of building classification models. In order to predict whether an item would be recommended by an user, the author introduces three classification algorithms:

[1] https://medium.com/analytics-vidhya/customer-review-analytics-using-text-mining-cd1e17d6ee4e

CART (Classification and Regression Trees), Random Forest, and Lasso Logistic Regression. Though, because the time cost for our group to try to implement these three types of algorithms is expected to be high as we are all not familiar with them, we decided not to include these classifiers in our study. But besides using different types of classifiers, we have definitely gained great insights on our own assignment from this article after all.

4.3 Another Study Example on Another Similar Dataset

Another Medium article that also does a similar study to ours we found is titled "Detecting bad customer reviews with NLP" [1], written by Jonathan Oheix. In it, the author also uses a dataset which comes form Kaggle and consists of customer reviews that are composed of textual feedbacks of their experiences at hotels, and an overall rating. For the purpose of the analysis, the author splits the rating into "good" reviews and "bad" reviews by a specific threshold, turning it into a classification problem. Just like the last article, the author cleans the data and pre-processes it (by lowering, removing punctuation and stop words, lemmatizing, etc.), splitting each review text into tokens of words.

The features that author builds based on his tokenization of words involve using Vader from the NLTK module for distinguishing neurtal, positive, or negative words. He also uses Word2Vec from the Gensim module, creating numerical vector representations of every word in the corpus by using the contexts in which they appear. Lastly, he uses the tf-idf method as well by using TfidfVectorizer from the sklearn module. He combines these three features together. For the classifier for his predictions, he chooses Random Forest.

His dataset turns out to be highly unbalanced - only less than 5% of the reviews are evaluated as negative. Knowing this exploratory result is essential for the author to set up his modeling. The author mentions that his original plan is to use the AUC (Area Under the Curve) ROC (Receiver Operating Characteristic) metric to evaluate the quality of his predictions, but the unbalanced data causes a problem. False positive predictions would not be significant, while true positives greatly affect the metric.

[1] https://towardsdatascience.com/detecting-bad-customer-reviews-with-nlp-d8b36134dc7e

To solve this problem, the author uses another metric instead - in which he claims to be better - the AUC PR (Area Under the Curve Precision Recall), or the AP (Average Precision). By comparing this AP metric of the built model with a baseline model, he is able to determine whether his model has good predictive power or not. The result for his model receives a AP score of 0.35.

4.4 Conclusion on Both Study Examples

In the two articles described above, both of them try to predict a classification problem using text mining features. They both involve the processes of pre-processing the review texts, tokenizing or separating the words, and converting those words into forms of quantification like vector matrices by methods such as bag of words, tf-idf, Vader, etc. Then, different types of classification algorithms are applied to predict on those transformed features of review text words. Examples of algorithms include Random Forest and Lasso Logistic Regression.

In conclusion, just like how it is summarized in the first article, the state-of-the-art methods currently employed to study this type of data which revolves around user reviews and ratings on products are composed of two main steps: building features on text data using text mining, and creating models by choosing appropriate classifiers. Building features by using methods like Bag of Words and tf-idf just like what is taught in the course of CSE 158 are currently popular choices in other real world data studies,as well as Logistic Regression being the classifier.

Learning from other existing exploratory studies on data similar to ours strengthens our confidence in our approach. Although the results of these articles are dissimilar to ours because of difference in datasets, our steps of processes for the findings and the conclusion are proven to be similar to them. Therefore, our approach for the predictive task is reassured.

5 Results and Conclusions

As shown in **Table 1, 2, & 3**, we can achieve the best accuracy by performing Logistic Regression classification with feature **F5**, a combination of text features and features of user measurements.

After various tests on this model (**F5-LR**), it can constantly achieve an accuracy of $(80.4 \pm 0.1)\%$ on

the test set, which is significantly higher than the accuracy of the baseline models, and outperforms Naive Bayes classification and Support Vector Machine classification under the same feature. Such an high accuracy is significant, because given that we are not taking advantage of the distribution of *fit* data in the dataset, we can predict a large part of customers' `fit` measurements correctly based on their personal statistics and reviews. Therefore, the clothing rental company can give better suggestions and feedback to a customer before the rental is complete, and better respond to potential return requests in advance.

We also test Linear Regression classification with other feature-building options (**F1-F4**), and record their performance as below (**Table 4**).

Model	F1-LR	F2-LR	F3-LR	F4-LR	F5-LR
Accuracy	0.734	0.732	0.798	0.802	0.804

Table 4: Accuracy of LR under different feature-building options.

This result justifies that combining one-hot encoded features of customers' personal measurements with text features (tf-idf) performs better than other feature representations.

The other feature representations under-perform because they consider only part of the features that the model is dependent on. **F1** only focuses on personal statistics, while **F2** only focuses on customers' subjective feelings. Although **F4** gives importance to both parts, it masks some predictive power of the non-text features.

In addition, Naive Bayes model performs much worse than Linear Regression and Support Vector Machine model because it makes wrong assumptions on the correlation among various entries of the dataset. Support Vector Machine model returns slightly worse accuracy, because the feature arrays become large when text features are incorporated, and the subjective nature of the "fit" measurement might input much noise into the dataset. SVM generally under-performs under these conditions.

The parameters of the model include `ngram_range=(1,2)`, `max_features=3000` for the TfidfVectorizer, and `C=10` for the LogisticRegression. Including bigrams to the features and increasing the number of features make the prediction more accurate as we hypothesized. Also, `C=10` means we need wekaer regularization on our classification madel to get optimal accuracy.

As a result, we conclude that the model **F5-LR** succeeds, because Linear Regression best suits the nature of the dataset, and because this model takes into consideration not only a customer's subjective feeling (`review_text`), but also her objective statistics (`height`, `weight`, etc.) when predicting on the customer's "fit" measurement. Also, this model makes these numeric features of objective statistics into one-hot encoded sparse arrays and concatenates them with text features, and thus enables the objective statistics to show their true predictive power in the feature representation.