CSE 158: Rating Prediction of Clothing Fit Dataset

Sijie Liu, Yingjing Wen, Mingjia Zhu

1. Introduction

Clothing is an important part of our daily life. The demand for clothing varies among different groups of people. Thus, it is important for clothing companies to understand their customer's needs through analyzing the customer's feedback on their products in order to better serve their customers. In this project, we try to assess what kind of model will be the most sufficient in illustrating the relationship between customer review data and rating and if the similarity between items purchased and customers plays a role in the model. We seek to understand what is the most contributing factor in determining the rating of an item and what model is the best at demonstrating the relationship between the factor and customer satisfaction in the clothing industry.

Throughout our study, we compare and contrast 5 models:

1. Logistic regression (baseline model)
2. Jaccard similarity-based model
3. Item2vec similarity-based model
4. Random forest (original training set)
5. Random forest (oversampled training set)

We concluded the following with our study:1) only review_text and review_summary has correlation with rating 2) similarity-based techniques do not work well for this dataset 3) sentiment analysis works better for this dataset 4) random forest (original training set) is our proposed model in which it obtained the highest overall efficiency.

2. Data

2.1 Overview of the dataset

The dataset we used is from kaggle's Clothing Fit Dataset for Size Recommendation (https://www.kaggle.com/rmisra/clothing-fit-dataset-for-size-recommendation?select=renttherunway_final_data.json), namely from the renttherunway_final_data.json file. It contains information about customers' reviews for items they purchased/rented from the website www.renttherunway.com. The file consists of 192,544 reviews. Each review has 15 variables and the descriptions are as below:

1. user ID and item ID: The IDs of the customer and item in each rental record.

2. rating and review: the customers rate their rental experience from 0 to 10, where 10 means the best; they also write reviews in text.
3. fit feedback: contains 3 categories: fit, small, large. This variable indicates whether the item fits, runs small or large for the customer.
4. customer measurements: include the weights, heights, ages, body type, and bust size of each customer in each rental record.
5. product measurement: includes the size of item in each rental record.
6. category information: contains the information about what the clothes were rented for (vacation, party, formal affair, etc.) and the categories of the clothes (shirt, dress, etc.)

2.2 Data preprocessing and EDA

We first checked if all variables have proper data types. Table 1 shows the first observation of the original dataset and the other rows follow the same format. We could see that the values in the "height" column are strings and in feet. We converted the values to inches in type float. For the "weight" column, we stripped "lbs" at the end of the value and converted them to floats. The values in "age" columns were strings, so we converted them to floats. In addition, we split "bust size" into two parts, bust measure and cup size (eg. 34d → 34 [bust measure] and d [cup size]). Other columns were already in appropriate types, so we didn't change them in this step. The first observation of the resulting dataframe was shown in Table 2.

	fit	user_id	bust size	item_id	weight	rating	rented for	review_text	body type	review_summary	category	height	size	age	review_date
0	fit	420272	34d	2260466	137lbs	10	vacation	An adorable romper! Belt and zipper were a lit...	hourglass	So many compliments!	romper	5' 8"	14	28	April 20, 2016

Table 1. The first observation of the original dataset

	fit	user_id	item_id	weight	rating	rented for	review_text	body type	review_summary	category	height	size	age	review_date	bust measure	cup size
0	fit	420272	2260466	137.0	10	vacation	An adorable romper! Belt and zipper were a lit...	hourglass	So many compliments!	romper	68.0	14	28.0	April 20, 2016	34.0	d

Table 2. The first observation of the dataset after converting data types

2.2.1 Uni-variable analysis

We looked into the distributions of each numerical variable by generating histograms and box plots. Figure 1 and Figure 2 show the distribution of height. We could see that height has a normal distribution. We also calculated the mean value, which was 65.31. The median is 65 inches, and the 1st quartile and 3rd quartile are 63 and 67, respectively. There are 7 outliers in this column; their values are either smaller than 57 or greater t

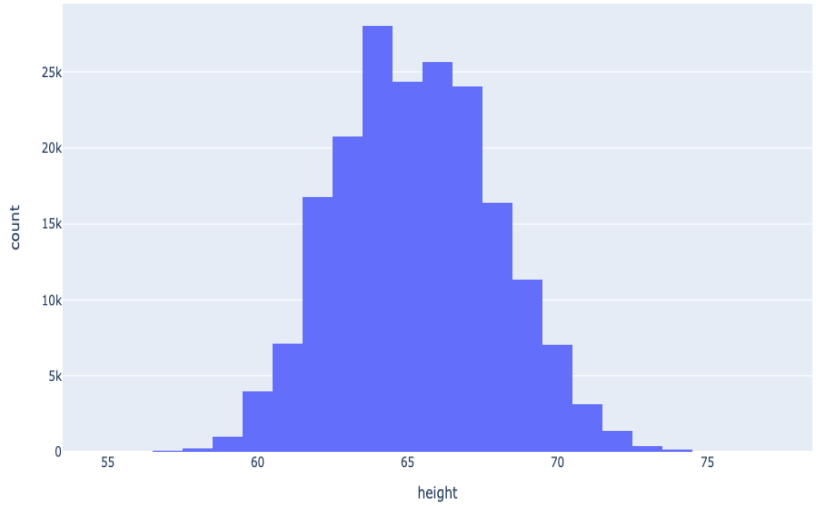

Figure 1. Histogram of height

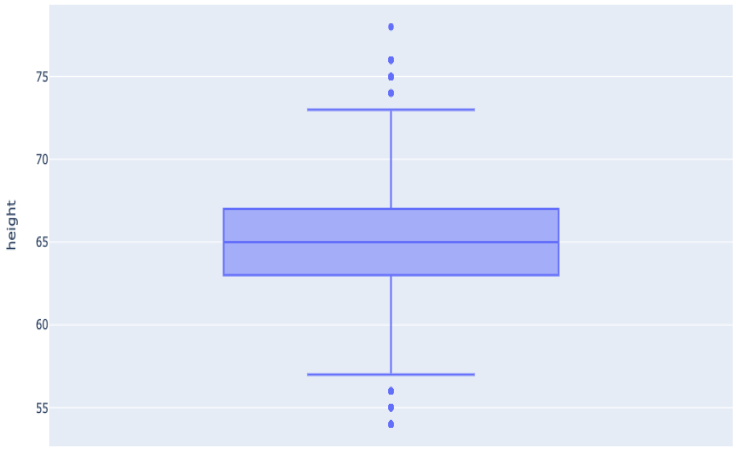

Figure 2. Box plot of height

We generated the same plots for the "age" column. From Figure 3 and Figure 4, we could see that the distribution of "age" is slightly skewed to the right. The mean value was 33.87 .The median value is 32, and the 1st and 3rd quartiles are 29 and 37, respectively. There are many outliers in this column, with values larger than 49 or lower than 17. We could also see some abnormal values in the box plot: some ages close to 120 or 0. We further handle these abnormal values in the data preprocessing section (2.2.2).

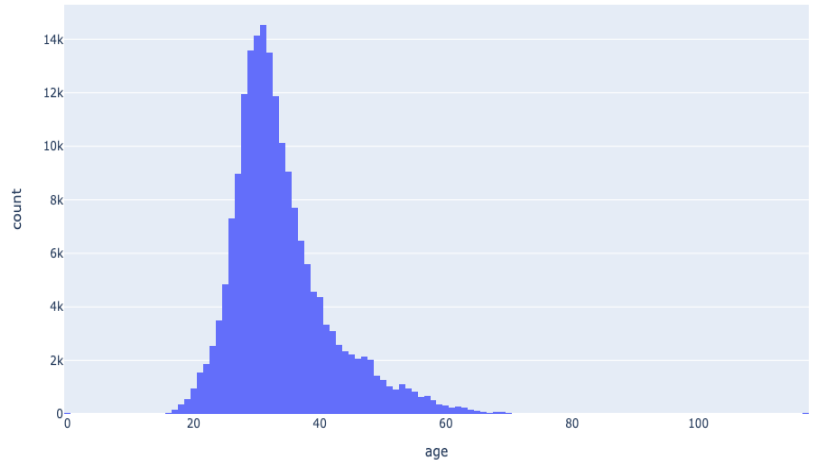

Figure 3. Histogram of age

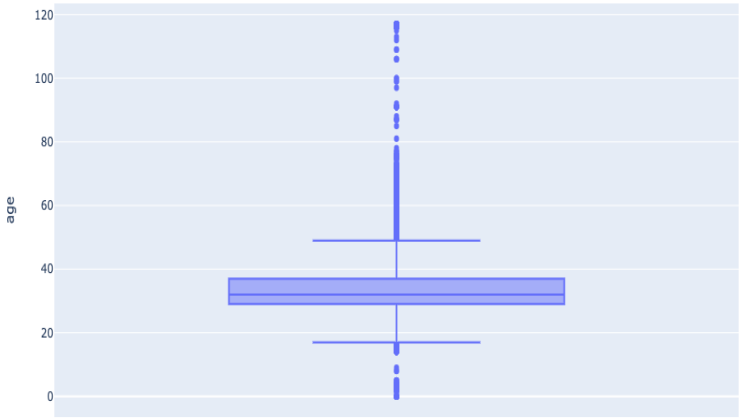

Figure 4. Box plot of age

For the "bust measure" column we created, from the histogram (Figure 5) below, we could see that the distribution looks like a bell shape, with a mean value equal to 34.2. According to the box plot (Figure 6), the median is 34, and most values are between 32 and 38. There are seven outliers.

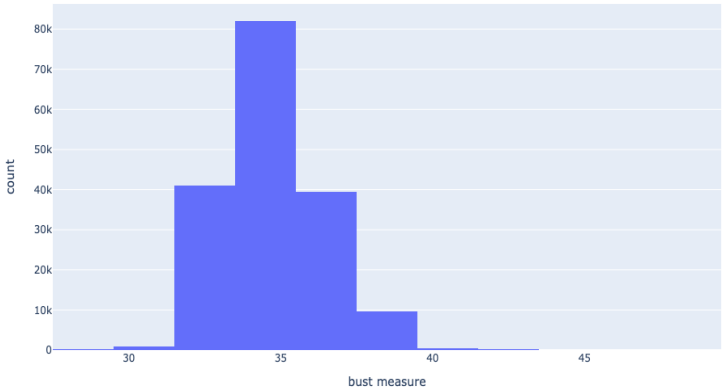

Figure 5. Histogram of bust measure

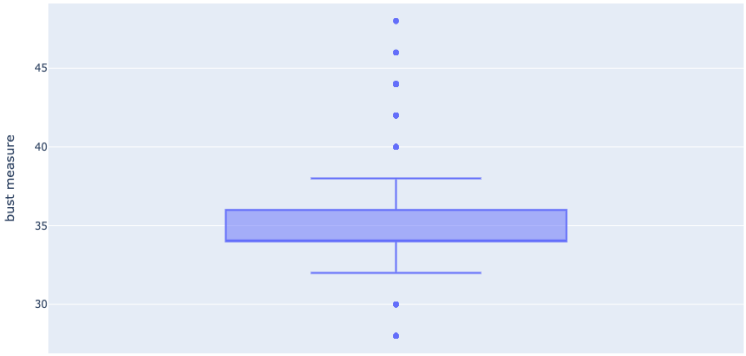

Figure 6. Box plot of bust measure

We also looked into the "weight" column from the two figures below. Figure 7 shows that the distribution of "weight" is right-skewed. We calculated the mean value, and it was 137.4. From Figure 8, we can see that the median is 135. There are many outliers in this column, whose values are either larger than 86 or smaller than 135.

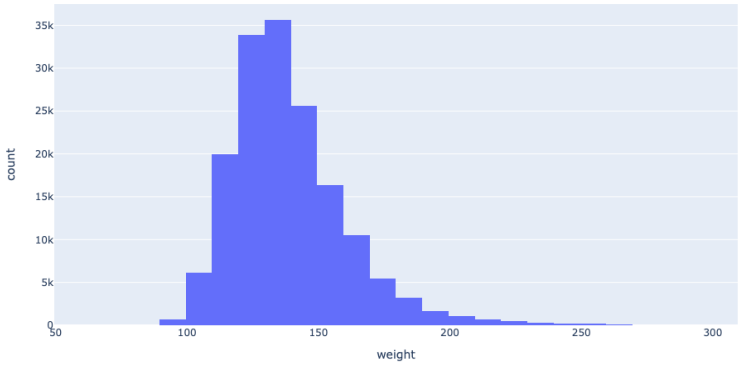

Figure 7. Histogram of weight

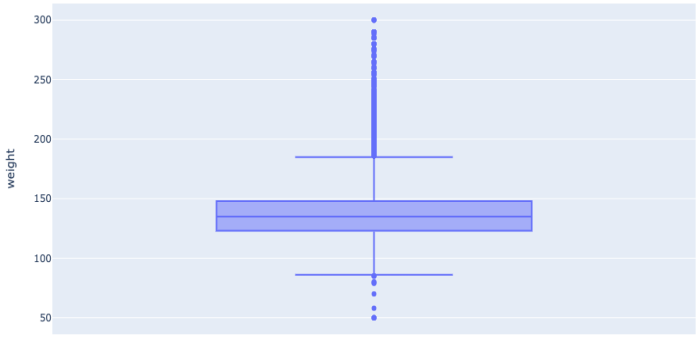

Figure 8. Box Plot of weight

After investigating the "size" column, we found that the values in this column vary for different brands or companies, so we can't standardize. Thus we decided to ignore this column.

We also looked into positive and negative phrases in the "review_text" and "review_summary" columns. We have plotted 60 most popular positive and negative words, which are figures 9 and 10. The output of the positive and negative word lists does align with our common knowledge of what would be included in a high/low rating respectively. Thus, we believe it would be a good choice to use these 2 word lists to predict ratings.

Figure 9. Positive words in reviews

Figure 10. Negative words in reviews

2.2.2 Data Preprocessing

In addition to converting data type, as we mentioned in the first paragraph of this section, we preprocessed the data, including finding appropriate values to deal with nulls and handling categorical variables. For the “height” column, from Figures 2 and 3, we could see that it has a normal distribution, and the outliers don’t contain extreme values, so we filled the null values with the average of all values. For the “age” column, as we can see in figure 5, there are some unreasonable outliers, so we dropped the rows with age larger than 90 or lower than five and filled the nulls with the median of all values. For the “bust measure,” we didn’t find any extreme outliers from Figures 6 and 7, so we filled the nulls with the average of all values. For the “weight” column, from figure 9, we could see many outliers, so we filled the nulls with the median of all values.

We have also preprocessed categorical columns, including “rented for,” “body type,” and “category.” In the “category” column, we found that there are different words with similar meanings, such as “pant” and “pants,” “skirt,” and “skirts,” so we unitized these categories. “Rented for” and “body type” columns don’t need to be changed. Then, we performed one-hot encoding on these three categorical columns; we let the feature vector be all zero for null values.

2.2.3 Bi-variable analysis

We looked into the count of each rating and the proportions each category in “fit” takes. From Figure 11, we can see that the “rating” column only contains five unique values: 2, 4, 6, 8, ad 10. Thus this problem is more suitable to be classified as a classification problem than regression. We could also see that the data is highly imbalanced. Most of the ratings are 10 and 8, and only a small part of the ratings are 2, 4, or 6. This tells us that we need to take the imbalanced data into account when generating the models.

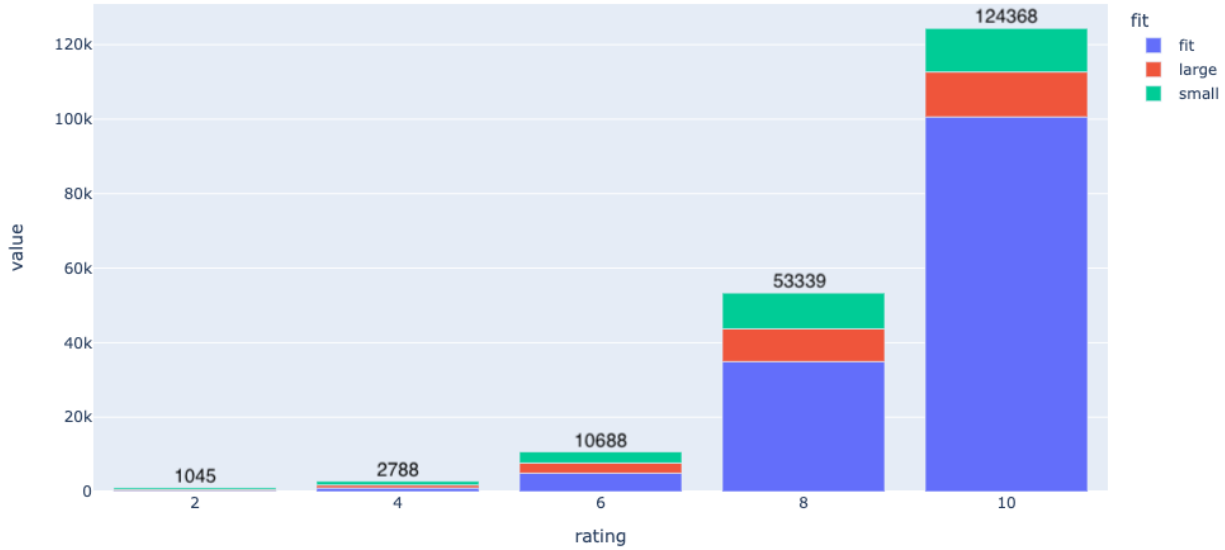

Figure 11. Stacked bar chart of rating with fit

2.2.4 Multivariable analysis

We plotted a pairplot to figure out if there are correlations between numerical variables. From the graph below, we didn't find any correlation between the factors. Thus we might not consider using those variables for rating prediction.

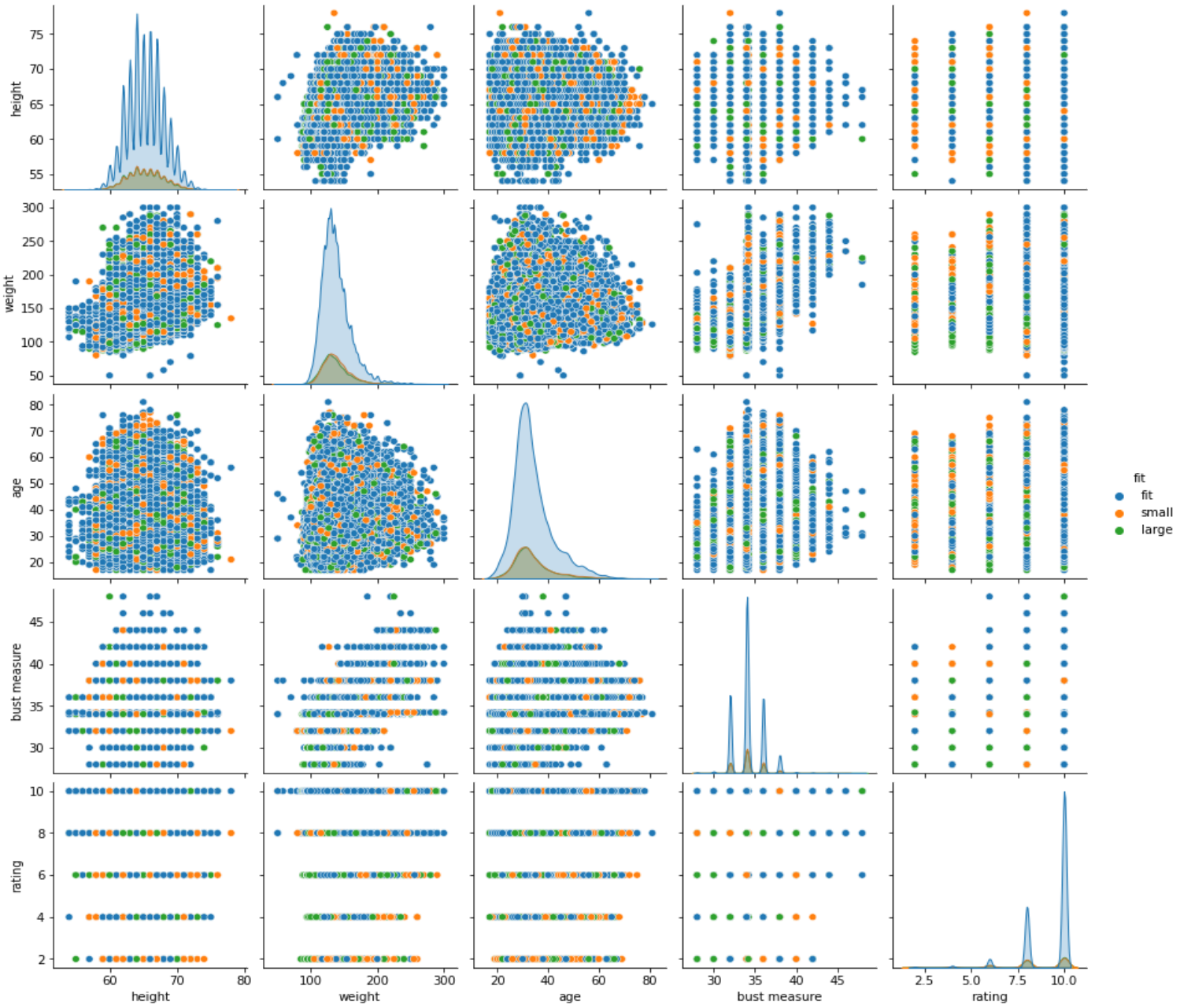

Figure 12. Pairplot of numerical variables

3. Predictive Task

Our predictive task for this project was to predict the rating for each review through classification. Since the rating column only contains 2, 4, 6, 8, and 10, this is a discrete variable rather than a continuous one. Thus, we believe classification models will be more suitable for this task. We used 80% of our data as the training set while the other 20% as the testing set.

3.1 Features

According to the EDA section, there's no correlations between the "rating" and all other numerical variables. Thus we decided to develop different models by using review_text, review_summary, item_id, and user_id. For review_text and review_summary, we preprocessed the variables through removing punctuation and lower-casing and stemming each word. Afterwards, we used SentimentIntensityAnalyzer in the nltk package to extract the positive and negative words/phrases. We extracted the unigrams and a combination of unigrams and bigrams. For unigram only, we extracted 241 positive words and 175 negative words. For the combination of unigrams and bigrams, we extracted 2727 positive word/phrases and 170 negative word/phrases. When creating features, we calculate the number of times that each positive and negative word (phrase) appears in each review_text/review_summary and append those numbers into feature vectors.

For item_id and user_id, we used them to calculate the similarities and predict the ratings by calculating the summation of weighted ratings of items similar to them. There's no need to preprocess item_id and user_id. A detailed description of the features each model used is shown in table 6.1.

Name of the Model	**Features**
Logistic Regression	1. Positive word list of review_text (unigram) 2. Negative word list of review_text (unigram)
Jaccard Similarity-based model	1. Item similarities according to unique user_id who purchased the item
Item2vec Similarity-based model	1. Item similarities according to unique user_id who purchased the item
Random Forest (original dataset)	1. Positive word list of review_text and review_summary (top 50,000 unigram + bi-gram) 2. Negative word list of review_text and review_summary (top 50,000 unigram + bi-gram)
Random Forest (oversampled dataset)	1. Positive word list of review_text and review_summary (top 50,000 unigram + bi-gram) 2. Negative word list of review_text and review_summary (top 50,000 unigram + bi-gram)

Table 3. Features used for each model

3.2 Baseline: Logistic regression

We used the Logistic Regression model as our baseline. Our features are the number of times that each positive and negative unigram appears in the review_text. For example, if a review text is "love love good i love it", and

the positive unigram list is [‘love’, ‘good’, ‘beautiful’, ‘pretty’], the feature vector should be [3, 1, 0, 0], each number corresponds to the number of times that the word appears. We trained a logistic regression and get the accuracy below (Table 4):

	precision	recall	f1-score	support
2	0.00	0.00	0.00	0
4	0.00	0.00	0.00	1
6	0.00	0.31	0.01	16
8	0.04	0.42	0.08	919
10	0.98	0.65	0.78	31292
accuracy			0.65	32228
macro avg	0.21	0.28	0.17	32228
weighted avg	0.96	0.65	0.76	32228

Table 4. Results of the baseline model

3.3 Validity of Model Prediction

To assess the validity of our model, we used micro-average and macro-average as our main metrics for model comparison throughout our experiment. **Micro-average** (aka **accuracy**, we will be using it interchangeably throughout the paper) is calculated by counting the total true positives, false negatives, and false positives for all samples. **Macro-average**, on the other hand, computes the metric independently for each class and then takes the average of all metrics (from each class), hence treating all classes equally. The reason for choosing micro and macro averages is because we want to understand the overall accuracy of the model (micro-average) while looking at the performance of the model on a class-wide basis since we are aware of the fact that we have an imbalanced dataset. On top of that, we will also be looking at the precision, recall, and f-1 score for each class throughout the report to present a more thorough evaluation of each model’s performances for each class.

4. Predictive Model

After trying several different models, we decided to propose the random forest model with the counts of positive and negative words in review_text and review_summary (unigrams and bigrams) as features. This model will be introduced in detail in section 4.3.1

4.1 Jaccard similarity of items

The Jaccard similarity is a measure of similarity for two sets of data. It compares the items in the two sets to check which items are shared and which are distinct. The formula is: $J(X,Y) = |X \cap Y| / |X \cup Y|$. In this model, we could calculate the Jaccard similarity between two items A and B by applying the formula above on the set of users who have bought item A and the set of users who have bought item B. Then we calculate the weighted rating based on those similarity scores.

Here’s how it works: given a specific user-item pair (u, i),

1) Loop through all of the items (*i2*) the user has bought
    a) For each *i2*, extract all of the users who have bought it as a set, calculate the Jaccard similarity with the set of users who have bought *i*. Append all values to a list called "similarities"
    b) Calculate the difference between the rating that *u* gives to *i2* and the average rating of *i2*. Append all differences to list called "ratings"
2) If the sum of the "similarities" list is larger than 0, we calculate a list of weighted ratings, in which the *i-th* item in the list is the product of the *i-th* item in "similarities" and *i-th* item in "ratings." Then we use the sum of values in the weighted rating list divided by the sum of values in "similarities", and add this division result with the average rating given by this user; else gives the average of all items.
3) Returns a value among 2, 4, 6, 8, 10 which is closest to the result we got in (2).

When making predictions based on an item-user pair, this model helps us to look into whether the user has rated similar items and calculate the weighted average according to the users who have bought similar items. For this model, we didn't run into any issues due to scalability and overfitting. Unlike some of the other models we tried, which didn't use the item_id and user_id, this model considered the similarities between users and items. However, this model is relatively simple since it only uses Jaccard similarity. After checking the results, we found that this model didn't work well. We will further discuss the results of this model in the "results" section.

4.2 Item2vec

We have made more attempts on similarity-based prediction. We utilized Item2vec, an adaptive version of Word2vec, to calculate the similarity between items. In Word2vec, we need to tokenize the documents to make them a list of words and fit them into the Word2vec model. The Word2vec algorithm uses a neural network model to learn the associations between words from the text. For Item2vec, we replace the "documents" in Word2vec with sequences of item_ids. After fitting the items into the model, the model will give us the similarity between items. Similar to the model in 4.1, this model also made predictions by using ids, trying to find the similarities between items based on the sets of users who have bought them. This model uses a more advanced method to find similarities compared to the model in 4.1. Although we didn't run into any issues due to scalability or overfitting, this model gave us bad results. The results of this model will be further discussed in the "results" section. From 4.1 and 4.2, we found that the similarity-based model might not be able to give us good results, so we decided to move on to models using reviews as features.

4.3 Random forest

Random forest is an ensemble learning method for classification/regression problems. It consists of many decision trees. We chose this model as our next attempt because the similarity approach is not a good fit for our dataset. Therefore, we decided to use a more complex classification model to model the pattern of our dataset. Random forest is also better for a multi-label dataset in comparison to our baseline.

We performed our experiment on 2 datasets: 1) the original training dataset (4.3.1) 2) oversampled training dataset (4.3.2) to compare if solving the imbalanced dataset problem with oversampling will benefit our model or not. The same features are used for 4.3.1 and 4.3.2.

4.3.1 Random Forest with Original Dataset (unigram + bi-gram)

There are two purposes of this model. First, we want to use this model to directly compare the performance of random forest to other methods we used so far. Second, we hope to utilize this model as a benchmark for testing if an oversampled dataset can solve the problem of an imbalanced dataset.

4.3.2 Random Forest with Oversample (unigram + bi-gram)

We used the SMOTE method to create an oversampled dataset for training. As shown in section 2.2.3, our dataset is extremely unbalanced i.e. more than 50% of data belong to the class rating 10 while less than 1% of data belongs to the class rating 2. The model is incapable of learning the decision boundary in this case. To counter this problem, we utilized the SMOTE method. The SMOTE algorithm can be summarized as below:

1. For each minority class (classes that are not the class with the most samples):
    a. Randomly select an instance (x)
    b. Find the k (k typically = 5) nearest minority neighbors of x
    c. Randomly select a neighbor from those k neighbors (y)
    d. Generate a synthetic example by randomly selecting a point between x and y in feature space
    e. Repeat this a-d n times, where n = # of samples of the most populated class - # of samples of the current minority class

4.4 Scalability

We will address scalability in 2 aspects: 1) runtime 2) performance on future data. Overall, all of the models we tested are not scalable due to either runtime issues or both runtime and poor performance on possible future data.

4.4.1 Runtime

The overall runtime for all models, except for 4.3.1 and 4.3.2, is around 30 seconds to 1 minute. These models are highly scalable from a runtime perspective. For 4.3.1, the runtime is ~10 minutes for a total of 160,000 samples. This model should still be scalable in terms of runtime if we are able to implement a multiprocessing method to train the model. Lastly, 4.3.2 took ~2 hours to train 517,245 samples (103,449/class). This model is not as scalable as other models in terms of runtime since the training time has significantly increased when the number of training samples increased.

4.4.2 Performance on Future Data

In terms of performance on future data, all of the models we tested are not scalable. None of them are complex enough to model the pattern of our dataset. All of the models would perform poorly if we are given new test data from all classes besides the class of rating 10.

4.5 Overfitting

We did not encounter any problem with overfitting. For all of the models we tested, the training set's micro-average is ~8% better than the test set; and, the macro-average is ~10%. There is no significant difference between the micro- and macro- average of the training set and the test set.

5. Literature review

As introduced in the data session of this paper, the dataset we used is collected from a cloth renting website. It was used to model the semantics of the fit feedback of the customers and develop a related classifier for the fit prediction based on the available transaction data in the store [1]. In Misra et al.'s paper, researchers tried to address the problem of imbalance labeling by using a metric learning approach; and to deal with the semantics of the customer review, they developed latent factor formulation models [1]. They find out that models with K dimensional latent variables (K-LV) might be more suitable for the task here, but they also find that K-LV might not work well with the metric learning approach as it does not provide a suitable representation to be easily separated for the metric.

Another similar dataset being studied can be the amazon review dataset. The research goal is the same as ours: use the reviews from users to predict the final rating; also, the researcher is dealing with unbalanced data. The researcher investigated the binary classification and multi-class classification using different classification algorithms on imbalance, undersampled, and oversampled data; the author then found that Naïve Bayes and SVM are better for both binary and multi-class cases compared to other algorithms chosen and pointed out that combining the text review with its summary better represent the reviews [2]. The author also said that when there are mostly good reviews, the undersampled data is preferred.

In Wang et al. 's work, researchers define a text mining problem named Latent Aspect Rating Analysis (LARA), which takes reviews and a certain aspect as input, and investigates each reviewer's latent ratings on the given aspects and their emphasis on the different aspects [3]. They proposed a state-of-the-art Latent Rating Regression (LRR) model and experimented on the hotel reviews data, which greatly represent the opinions and reviewing behaviors of customers. The functionality of the LRR is very expressive as it can not only solve reviews from different perspectives (business service, location, cleanliness, etc.) but can also output a ranking or aspect opinion summary. Besides, LRR can solve the orientation of opinions with respect to the aspect of the study. In our study, we might have the situation that "no problem" might be regarded as negative wording but in fact, it is showing a neutral or positive opinion. Our negative words and positive words interpretation can not deal with words for specific situations (i.e. our clothing review). On the other hand, LRR is able to access the opinion of words reliably as they can connect the meaning of words for the specific context and evaluation settings (i.e. "linen" can refer to good "cleanliness" based on the study of LRR but in most of the cases "linen" is just a noun that does not show people's opinion). That can be a reason for the difference between the performance of our predictions.

6. Results

In this section, we will go over and compare the results of the model we used. Due to the fact that we have an extremely unbalanced dataset, in which the accuracy is dictated by the class of rating 10, we believe that macro-average is a more desirable metric to focus on when we are choosing our proposed model because we want our model to mimic the general pattern across all classes. Therefore, our proposed model is the random forest model trained with the original dataset (6.3.1). Its micro-average is not as good as some of the other models (including the baseline), but the macro-average has the second-best score, just 0.01 away from the best. Moreover, we did not choose the random forest model with the oversampled dataset, even though it has the highest macro-average. This is because the random forest model with an oversampled dataset is not scalable. The training time is six times more than the model trained on the original dataset, but the macro-average only increased by 0.01 while the accuracy dropped by 0.14. Overall, we do not think the trade-off is worthwhile.

Over the course of the experiment, the features we tried can be classified into 2 large categories: 1) sentiment analysis of review_text and/or review_summary 2) similarity analysis. For logistic regression and random forest, we used sentiment analysis of review_text and/or review_summary. We concluded that 1) using positive and negative wordlists of both unigram and bi-gram is better than having just unigram 2) including wordlists for both review_text and review_summary is better than having review_text alone. For Jaccard Similarity of Items and Item2Vec, we used Jaccard similarity of items and similarity model using neural network. We concluded that 1) Jaccard similarity is better than the similarity model using neural networks. 2) Both similarities are not as sufficient as sentiment analysis.

6.1 Jaccard Similarity of Items

Table 5 shows the results of the model in 4.1. We could see that the model gains the same micro-accuracy (0.65) compared to the baseline, while the macro-average (0.16) of it is even below the baseline. Moreover, we got nearly 0 f1-score in class 2, 4, 6, and 8, which means that the model only works well on predicting rating 10.

This is one of our unsuccessful attempts. This model shows that the Jaccard similarity is not a proper approach for this question. This is likely because whether a clothes can fit a person largely depends on the clothes and person themselves, and it's hard to infer whether a person is satisfied with a clothes based on other customers' opinions. Thus this dataset might not be suitable for using Jaccard similarity.

	precision	recall	f1-score	support
2	0.00	0.00	0.00	2
4	0.00	0.12	0.00	8
6	0.00	0.12	0.01	40
8	0.01	0.34	0.02	218
10	0.99	0.65	0.79	31960
accuracy			0.65	32228
macro avg	0.20	0.25	0.16	32228
weighted avg	0.99	0.65	0.78	32228

Table 5. Results of the model use Jaccard similarity

6.2 Item2vec

The results of this model are shown in the table below. We could see that compared to the model above, the predictions of classes with lower ratings are slightly better. However, the model doesn't work well overall, with a micro average of 0.62 and a macro average of 0.14. The reason for the failure might be similar to what we mentioned in 4.1; the ratings of each person on each item are personalized. Thus we decided to stop using similarity-based models and start to explore models that use reviews.

	precision	recall	f1-score	support
2	0.00	0.00	0.00	1
4	0.00	0.11	0.00	9
6	0.00	0.10	0.01	48
8	0.02	0.35	0.04	554
10	0.95	0.65	0.77	30510
accuracy			0.62	32228
macro avg	0.16	0.20	0.14	32228
weighted avg	0.90	0.62	0.73	32228

Table 6. Results of the model use Item2vec

6.3 Random forest

We are trying to solve a multi-class classification problem using random forest. Here we separated our task into fitting the random forest classifier with the original dataset and oversampled dataset.

6.3.1 Random Forest with Original Dataset

	precision	recall	f1-score	support
2	0.02	0.31	0.04	13
4	0.01	0.12	0.03	56
6	0.04	0.20	0.07	388
8	0.26	0.38	0.31	6076
10	0.86	0.70	0.77	25695
accuracy			0.63	32228
macro avg	0.24	0.34	0.24	32228
weighted avg	0.74	0.63	0.67	32228

Table 7. Results of the Random Forest Model with Original Data

The result of our random forest with the original dataset is shown above. Although the overall accuracy (micro-average) of the model compared to our baseline decreased by 0.02, the macro-average increased by 0.08. There are improvements of f-1 scores (looking at each class individually) for all classes of rating 2, 4, 6, and 8 while maintaining a relatively stable f-1 score for the class of rating 10. Like what we addressed at the beginning of this section, we want our model to have an overall better performance for all classes so that it is able to capture a more general pattern of the dataset while being the most sufficient (resource-wise). Therefore, this is our proposed model.

Admittedly, this model might still be improved. When we look into the dataset, we found that many of the reviews with low ratings contain complex sentiments i.e. they might comment about liking the dress but think the dress is small or does not look good on them. We should look into more advanced NLP models such as LSTM which is capable of conducting time-series analysis to explore the mixture of sentiments in reviews.

6.3.2 Random Forest with Oversampled Dataset

	precision	recall	f1-score	support
2	0.54	0.04	0.07	2636
4	0.13	0.05	0.08	1142
6	0.19	0.10	0.13	3269
8	0.30	0.34	0.32	8021
10	0.61	0.74	0.67	17160
accuracy			0.49	32228
macro avg	0.35	0.25	0.25	32228
weighted avg	0.47	0.49	0.46	32228

Table 8. Results of the Random Forest Model with Oversampled Data

For the random forest classification with an oversampled dataset, the overall accuracy is 0.14 less than the accuracy from the original dataset, while the macro average increases by only 0.01 (similar result). There are improvements of f-1 scores (looking at each class individually) for all classes of rating 2, 4, 6, and 8. However, the f-1 score for the class of rating 10 decreased significantly. Although this model better illustrates the pattern of the classes of rating 2, 4, 6, and 8 (but still relatively trivial with an average increase of 0.2), the f-1 score of the class of rating 10 decreased by 0.1. More importantly, the runtime of this model is 6 times more than the random forest using the original dataset. This shows that the runtime skyrocketed due to the fact that we need to train 5x more training samples than before, but the oversampled data provided trivial additional information about the minority classes. Therefore, we do not think the improvement is sufficient when considering the tradeoff between improvement of macro-average and accuracy and computing resources.

7. Conclusion

We analyzed the clothing size and fit dataset of the clothing company Rent the Runway for the purposes of assessing the relationship between rating and other variables and what kind of model can best illustrate the pattern of the dataset. Our conclusion is that: 1) no numerical variable in the dataset has correlation with the rating 2) similarity-based model is insufficient for this dataset 3) review_text and review_summary should both be used as features and both are important 4) it is better to use a combination of unigram and bi-gram for our task.

Our proposed model is the random forest model using the original training set because it is the most efficient model considering both scalability and macro-average. Although our proposed model did not improve much from the baseline model, in fact the accuracy was lower than baseline by 0.02; we believe the model captures a more accurate pattern for all classes unlike the baseline. The baseline has such a high accuracy is probably because the sum of numbers of all samples from the minority classes is roughly equal to the number of samples of the class rating 10. This means that if the logistic regression model performs like a binary classifier, which is what it is good for, it should get a high accuracy. Our hypothesis is supported by the output of the logistic

regression model as well in which the f-1 score of the class of rating 10 is as high as 0.78 while the f-1 score for all other minority classes are <= 0.08. Same goes for the similarity-based models, which are even worse at capturing the pattern of other classes other than the majority class.

Some of the future improvements we hope to achieve is to try an NLP model such as LSTM in which it is able to capture the sentiment in review_text from a time-series perspective. This should help us get a more accurate and detailed sentiment analysis of the review_text and review_summary in which we can classify the sentiments into 5 or more categories rather than just positive and negative. It will also be beneficial if we are able to obtain more data for the minority classes (classes of rating 2, 4, 6, 8) to solve the fundamental problem of imbalanced datasets.

Works Cited

[1] Misra, R., Wan, M., & McAuley, J. (2018, September). Decomposing fit semantics for product size recommendation in metric spaces. In *Proceedings of the 12th ACM Conference on Recommender Systems* (pp. 422-426).

[2] Martin, M. (2017). Predicting ratings of amazon reviews-techniques for imbalanced datasets.

[3] Wang, H., Lu, Y., & Zhai, C. (2010, July). Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 783-792).