Fit Predictions on Clothing Data

Adalina Ma	Jenna Canicosa	Anastasiya Markova	Mohit Sridhar
UC San Diego	UC San Diego	UC San Diego	UC San Diego
axma@ucsd.edu	jcanicosa@ucsd.edu	anmarkova@ucsd.edu	msridhar@ucsd.edu

ABSTRACT

In order to enhance customers' purchasing experiences and lower the number of returned products, product fit predictions are essential. Fit input from customers is difficult to model because of many external factors such as the manufacturer, body type, subjective assessment of products, and uneven label distribution. This study presents a prediction framework to address the product fit problem. It uses different learning techniques to try and address label imbalance concerns and hopes to aid with recommended products that will fit “true to size” for the consumer. We also used two publicly available datasets that we were given to conduct this experiment from two internet clothing stores.

1 INTRODUCTION

Given the expansion of the e-commerce fashion sector and the unpredictable size discrepancies across apparel items, it is vital to investigate the possibility of automatically generating precise and customized fit recommendations. Predictive models have recently been established based on this type of data because retailers frequently allow consumers to submit fit comments (e.g., "small," "fit," "large") during the product return process or while posting reviews. Recent approaches to this problem use two sets of latent variables to recover products’ and customers' predicted fit. We further use a sigmoid function to categorize ratings outputted by the latent model. We were curious to see if we could predict the fit of products for a customer based on either information we know about them or their previous purchases and reviews. In this research, we treat the fit prediction to address the aforementioned difficulties in the following ways: First, we create three Latent Factor Models that use user, item interactions to predict a rating between 0 and 1 for each label. We then use a Logistic Regression to classify a vector of these ratings into a particular category. Using this approach we were able to achieve improved results of True Positives for each category although there was still a lot of misclassification present.

2 RELATED WORK/LITERATURE

With the increase of popularity in e-commerce, we see the rise of the product size recommendation problem and increase in research for this particular question. In this paper we will discuss the similarities and differences in approach for this research topic [1, 2, 3, 4, 5]. The data we used within this work comes from two datasets, *ModCloth* and *RentTheRunWay*, which are online women clothing websites. ModCloth specializes in vintage ecommerce divided into three categories: dress, tops and bottoms. RentTheRunWay is an online renting service that allows users to rent pieces for events. Both datasets contain self-reported fit feedback from users along with categories such as: rating, sizes, user measurements, etc. Similar to our approach, [1, 2, 4] conducts their experiment on the same datasets that we have provided.[3] dataset of online fashion from over 14 countries not specified within the paper and [5] utilizes data from ASOS, a popular online retail clothing store. [1] focuses on recovering products’ and customers’ ‘true’ size and uses these as features in a standard classifier for fit prediction, going in a similar fashion by using latent factor model. [2] differs from our task as it focuses on

capturing fit semantics and handles label imbalances using metric learning approaches with prototyping, taking a very specific route of analysis from reviews. [3] introduced a personalized size and fit prediction method employing a split-input neural network architecture with global and entity-specific parameters. This method incorporates a Bayesian model for size recommendations across genders, alongside the SFnet, a deep learning approach that combines collaborative and content-based modeling to learn size and fit predictions' representations, providing flexibility and accuracy. [4] introduces SizeFlags, a potent approach leveraging Bayesian methods to model uncertainty and utilize priors effectively. By delving into customer reviews to scrutinize potential discrepancies in "true" sizes, this method focuses on minimizing size-related returns. [5] utilized the Product Size Embedding (PSE) model which offers a fresh solution for fashion e-commerce's size recommendation challenge. It excels by using only customer-product interactions and brand information, showcasing superior accuracy in mapping product sizes to a unified space compared to other methods. These existing methodologies collectively showcase a landscape of varied approaches, incorporating Bayesian modeling, latent factor models, deep learning, and review analysis to tackle the challenge of fit prediction in e-commerce. While some align closely with our aims, others take unique paths, emphasizing different facets of the problem, providing a diverse range of insights and methodologies to consider in our pursuit of an efficient recommender system for fit prediction.

3 METHODOLOGY/ PREDICTIVE TASK

We decided to narrow down our predictive task to predicting ‘small’, ‘fit’, or ‘large’ labels for a user, item pairs. Our data had more than 200,000 interactions so we could use a recommender model, such as Latent Factor Model since we had enough interactions. Our dataset also contained additional features such a height and weight, which are useful in classification models. Since our prediction is a categorical output we used two measurement techniques to assess how accurate our models are. The first one is accuracy, simply checking the percent of correct predictions. However, this wouldn’t be enough to assess the performance of our model. Our data is heavily skewed since around 70% of our data represent ‘fit’ and the rest 30% is an equal split between ‘small’ and ‘large’. This makes it easy to have high accuracy if we predict ‘fit’ for all values. This is the reason we chose to consider F1-score as another statistic by which we assessed our models.

Before we created the models, we prepared the necessary features. We began by removing all of the rows with NaN values in the height and weight columns. We also needed to clean the data by converting height into inches only so it could be represented as one number and we did the same for the weight converting it to pounds. We also decided not to keep the rest of the columns. This left us with two datasets: one on RentTheRunway and the second on Modcloth. Since Modcloth did not have the weight data we decided to keep it out of our classification model and only run the model on rent the runway data. For the Latent Factor Model approach we decided to keep all of the data in order to maximize user interactions. Since the latent factor model. Since the latent factor model predicts the ratings we decided to convert the rating into binary values creating a column for each label and using 0, and 1 to signify where that label was chosen. We used the data we cleaned to test out our predictions with the Classification and the Latent Factor Models.

4 DATASETS

Our data is composed primarily of two datasets obtained from the websites *ModCloth* and *RentTheRunWay*. ModCloth sells women's vintage clothing and accessories, from which data belongs to three categories: dresses, tops, and bottoms. RentTheRunWay is a platform that allows women to rent clothes for various occasions. These datasets contain self-reported fit feedback from customers as well as other metadata like reviews, ratings, product categories, catalog sizes, customers' measurements (etc.).

Based on the information we received from the dataset, we decided to begin with a latent-factor model to consider the customers' input, focusing more on recovering "true" sizes. This approach allows us to learn more about size preferences.

We initiated the data cleaning process by eliminating columns irrelevant to our research goals. Specifically, in the Modcloth dataset, we retained only the columns item_id, size, height, fit, and user_id. Similarly, for the RentTheRunway dataset, we preserved the columns: item_id, size, height, fit, and user_id. Our next step involved standardizing the height measurements by converting them into inches. Following this, we merged the datasets, paving the way for us to begin our experimental work and modeling. Our goal with this merged dataset is to explore the correlations between user measurement and fit feedback.

Statistics	ModCloth	RentTheRunway
# of Transactions	82,790	192,544
# of Users	47,958	105,571
# of Items	5,005	30,815
Percentage of Small	15.7%	13.4%
Percentage of Fit	68.6%	73.8%
Percentage of Large	15.8%	12.8%
Percentage of Customers with 1 Transaction	38.5%	37.3%
Percentage of Item with 1 Transaction	40.5%	26%

General statistics about both datasets can be found in the table above. It is important to note that the number of items count every item in different sizes as its own unique product to consider how each item's clothing size can differ. Note that the data is scattered considering almost 40% of the data in both datasets only have a single transaction from a customer.

5 MODEL/ EXPERIMENTS -

(Note: APIs are from scikit-learn)

For our baseline model we decided to experiment with two different types of classifiers. First, we used a OneVsRestClassifier (OvR) with SVC, or C-Support Vector Classification, as our estimator. Since we are performing multiclass classification, this seemed like a good start since OvR is the most commonly used strategy and a fairly default choice. In our case, this meant that the algorithm would fit a classifier for each label and choose the one with the highest confidence compared to the others.

Although the model achieved around **69%** accuracy, the training time took much longer than anticipated (see Table below). Since the fit time scales at least quadratically with the number of samples, SVC may be impractical beyond tens of thousands of samples. For reference, the *RentTheRunWay* dataset itself has over 190,000 reviews.

In order to combat this issue, we decided to switch to a linear classifier with SGD training. Our final baseline model consisted of a Pipeline composed of a StandardScaler, to transform the input features, and a SGDClassifier with logistic regression to classify the fit. SGDClassifier is an estimator that implements regularized linear models with stochastic gradient descent (SGD) learning. The algorithm is beneficial for handling large-scale datasets, as it updates the model parameters on samples of observations rather than using the entire dataset.

Additionally, we trained the baseline model solely on the *RentTheRunWay* data because it consisted of two important features, 'Weight' and 'Height', which we wanted to use to test the model prior to using the larger, merged dataset.

After dropping missing values from the dataset, we created a training and testing split with roughly 25% of the data being used for testing. In terms of the hyperparameters for the model, we use Logistic Regression, or probabilistic classification, as the loss function. We used 1000 epochs for training and decided to allow early stopping in order to prevent overfitting of the data. After fitting the model to the training data, the accuracy of the baseline model came out to **73.5%**. This is relatively good performance for a baseline model from which we would try to improve upon.

However, there was one major drawback to this model - it only predicted 'fit' for all of the testing data. In other words, it never predicted 'small' or 'large'. This can be attributed to bias in the data since over 75% of the *RentTheRunWay* data was composed of 'fit' reviews/labels. This means that the model learned to predict 'fit' for the testing data since it was able to produce an optimal accuracy doing so. The model itself didn't seem to have any issues since it understood there were 3 possible classes to predict, but rather became biased as a result of the data it was trained on. A possible fix for this issue would be to resample the original data in order to artificially increase the number of 'small' and 'large' labels. This would adjust for any biases within the training data and thus remove any bias from the model itself.

	Model	
Step	**OvR with SVC()**	**SGD**
Training	3m 51s	~5µs
Inference	4m 49s	~3µs

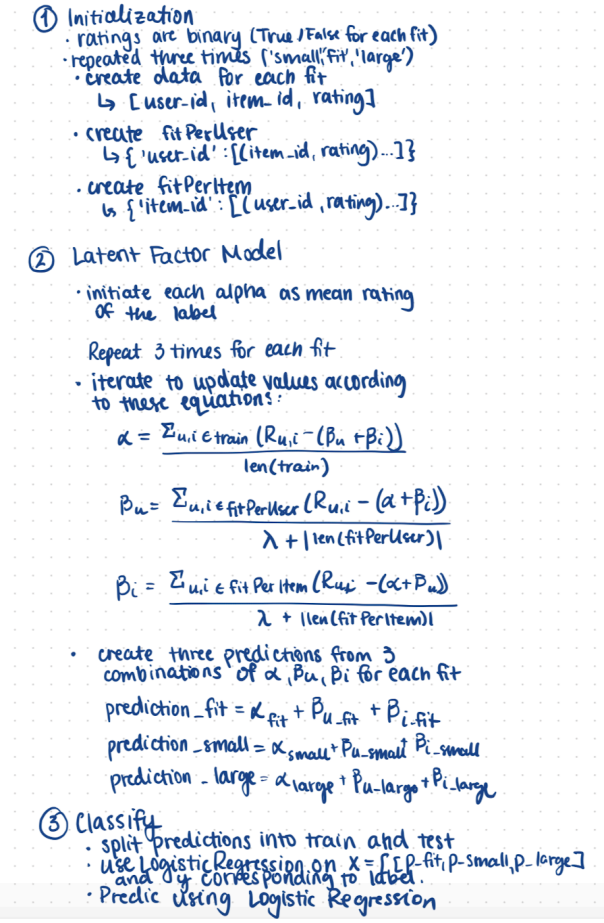
After attempting a classification, we decided to use a latent-factor model. We decided to use small, fit, and large as ratings and translate them into a numeric scale. As mentioned above we created a binary vector of three values and represented which fit applied with zeros and ones. We used the index values as the ratings for this model (0, 1, 2) corresponding to ('small', 'fit', 'large') and alpha values started as the mean of our data. Using the latent-factor model we were able to predict the approximate rating. Since the ratings returned by the model were continuous rather than categorical we decided to round the values so that they correspond to our rating scale. The initial model produced an accuracy score of 73.35%. Although the model achieved a pretty good accuracy because our dataset is very unbalanced we wanted to further explore how well our model is performing. For this we used an F1-scores for each class which were 0.16, 0.84, and 0.16 for small, fit, and large respectively. Even though this model performed better than our classifier, it is still bad at predicting large and small fit based on our F1-score. This would be a challenge to fix since the issue likely lies in the fact that our data is so

biased towards ‘fit’. However, we attempted to improve this.

The improved version of our original Latent Factor Model was to use three latent factor models. Each model predicted the rating between 0 and 1 for each type of fit. We had one model for ‘small’, one for ‘fit’, and one for ‘large’. Then we used a portion of our predictions to train a logistic regression to classify the vector of ratings. The rest of our data was test data we assessed our predictions on. Tuning the hyperparameters didn’t yield any improvements of significance thus we decided to keep the model with its initial parameters.

We were hoping that by predicting each rating individually and using Logistic Regression to classify these ratings into a particular category we would be able to counteract some of the disbalance caused by the dataset.

Algorithm:



Although this model didn’t reach a higher accuracy, the F1-score did in fact improve. Accuracy rate for this model was 72.61%, which is comparable to the results of the Latent Factor Model we ran previously. The F1-scores for ‘fit’, ‘small’, and ‘large’ are 0.83, 0.18, 0.24 respectively. You can clearly see the misclassifications in our confusion matrix.

Confusion Matrix:

	Predicted fit	Predicted small	Predicted large
True fit	18792	517	614
True small	3350	458	60
True large	3095	27	621

Although the accuracy decreased the error F1-scores show that we are actually better at identifying the true values for each category than originally. Nevertheless, the biggest downside is the fact that our dataset is heavily skewed towards fit, so it is hard to predict all three labels correctly, even though we made improvements.

6 RESULTS AND CONCLUSIONS

Predicting ‘small’, ‘fit’, and ‘large’ on our dataset was fairly difficult due to the fact that it was unbalanced. Our dataset consisted mostly of the ‘fit’ labels which biased the prediction of any model to be ‘fit’ over other categories. A regular classification model tended to classify all labels as ‘fit’ despite the use of additional features. The Latent Factor Model did perform better but was still quite biased towards ‘fit’ labels. The decomposed Latent Factor Model for each category had less bias than previous models

but was still heavily biased overall. Although we saw our accuracy decrease from 73.5% in the classification model and 73.35% in the Latent Factor Model to 72.61% in the decomposed Latent Factor Model. We also saw that our F1-score improved from the original Latent Factor Model. Original model has the F1-scores for ‘small’ and ‘large fit' of 0.16 while the decomposed model improved the scores to 0.18 and 0.24 respectively. Our results show that perhaps our predictive task was not the best fit for this dataset as a whole. The results of two greatly varying models were very similar and didn’t yield very accurate results for all three labels. Perhaps it would be more beneficial to select a more specific predictive task such as predicting whether the item that didn’t fit was too large or too small. Alternatively, we could potentially manually balance our dataset for more accurate predictions for all three categories. Unfortunately, working with the data only sets up our model for failure, unless there is a different predictive technique that we weren’t able to think of in the process. Nevertheless, this was a great learning opportunity to consider the distribution of the values in the dataset when deciding on a predictive task such that it would yield good results.

REFERENCES

[1] Vivek Sembium, Rajeev Rastogi, Atul Saroop, and Srujana Merugu. 2017. Recommending Product Sizes to Customers. In *RecSys*

[2] Rishabh Misra, Mengting Wan, and Julian McAuley. 2018. Decomposing Fit Semantics for Product Size Recommendation in Metric Spaces In *RecSys*

[3] Abdul-Saboor Sheikh, Romain Guigoures, Evgenii Koriagin, Yuen King Ho, Reza Shirvany, Roland Vollgraf, and Urs Bergmann. 2019. A Deep Learning System for Predicting Size and Fit in Fashion E-Commerce. *In Thirteenth ACM Conference on Recommender Systems (RecSys ’19)*.

[4] Andrea Nestler, Nour Karessli, Karl Hajjar, Rodrigo Weffer, and Reza Shirvany. 2021. SizeFlags: Reducing Size and Fit Related Returns in Fashion E-Commerce. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[5] Kallirroi Dogani, Matteo Tomassetti, Sofie De Cnudde, Saúl Vargas, and Ben Chamberlain. 2019. Learning Embeddings for Product Size Recommendations. In *Proceedings of the SIGIR 2019 Workshop on eCommerce.*