Rating and Fit Prediction on RentTheRunWay Dataset

Wang Pi
University of California, San Diego
La Jolla, California, United States

Xiaohang Zeng
University of California, San Diego
La Jolla, California, United States

Ming Cheng
University of California, San Diego
La Jolla, California, United States

ABSTRACT

In clothing recommendation, predicting customers' potential rating on different clothes and whether the size of cloth would be appropriate are most common and essential tasks. The accuracy of rating and fit predictions both have significant impacts on the effectiveness of recommendation. Essentially, rating and fit prediction are classical regression and classification problem in machine learning. Quite many models in the literature are proposed to tackle them. To have a clear vision of the performance of different models on clothing recommendation, we compared the mean squared error of rating prediction and accuracy of fit prediction of different models on RentTheRunWay, a clothing transactions and reviews dataset collected by Rishabh Misra *et al.* in [7], in this paper.

KEYWORDS

rating prediction, cloth fit prediction, latent factor, neuron network, bag-of-words, boosting

ACM Reference Format:
Wang Pi, Xiaohang Zeng, and Ming Cheng. 2019. Rating and Fit Prediction on RentTheRunWay Dataset. In *CSE 258: Web Mining and Recommender Systems*. ACM, New York, NY, USA, 6 pages.

1 INTRODUCTION

With the significant growing of online shopping, products recommendation based on previous transaction records and user reviews are becoming more and more important for retailers to improve customers' online shopping experience and their sale avenues. In this paper, we focus on two specific tasks:

(1) Predicting the potential rating a user may give to a cloth.
(2) Predicting whether a cloth is fit for a user.

In practice, an item is eventually recommended to a user if the rating prediction of this user-item pair is high and this item is fit for a user. However, we don't discuss the combination approaches of two prediction results, but rather, we only evaluate the performance of rating and fit prediction individually in the scope of this paper.

In both tasks, we implement various models by combining many learning algorithms with different features. First, we adopt different learning algorithms in both tasks. For rating prediction, simple linear regression and more complicated latent factor model and feed-forward neuron network are implemented. For fit prediction, logistic regression, support-vector machine, random forest, boosting method are used. Second, different features are selected, from simple explicit features, e.g. customers' weight, height and size of clothes, user-item interaction records, to more complicated extracted bag-of-words feature vectors.

CSE 258: Web Mining and Recommender Systems, 2019, La Jolla, CA

Table 1: Number of customers, clothes and records in RentTheRunWay dataset

	Customers	Clothes	Records
Before Filtering	105,508	5,850	192,544
After Filtering	77,347	5,736	146,381

2 DATASET ANALYSIS

Before getting down to choosing and implementing different regression and classification models, we collected some important statistics of our dataset. The original RentTheRunWay dataset has some records that don't have all terms, so we filter out those records. The number of customers, clothes and records before and after filtering are shown in Table 1.

After filtering, each record contains 15 terms and the form of a single record is:

```
{
  "fit": "fit",
  "user_id": "420272",
  "bust size": "34d",
  "item_id": "2260466",
  "weight": "137lbs",
  "rating": "10",
  "rented for": "vacation",
  "review_text": "An adorable romper! ...",
  "body type": "hourglass",
  "review_summary": "So many compliments!",
  "category": "romper",
  "height": "5' 8\"",
  "size": 14,
  "age": "28",
  "review_date": "April 20, 2016"
}
```

Then we collected and plotted the distribution of 10 terms: "fit", "bust size", "weight", "rating", "rented for", "body type", "category", "height", "size", "age", to have a better understanding of statistics of this dataset.

As we can see from Fig. 1, the distribution of fit, large and small are quite imbalanced. The ratio between them are approximately 6:1:1. This imbalance property actually caused some problems for the fit prediction, which will be analyzed later. In Fig. 2 we can see

that bust size of most records fall into the range of 34-43 inches. The envelop of the distribution of weight seems like Gaussian curve. Another very imbalanced term is the rating, which can be seen in Fig. 4. Most ratings are 10 or 8. Top 3 rent purposes are wedding, formal affair and party. Top 2 body type of customers are athletic and hourglass. Top 3 categories are dress, gown and sheath. The size of different clothes varies from 0 to 58, which actually causes some problems for the fit prediction. And eventually, the distribution of customers' age and height are also Gaussian like, which is as expected.

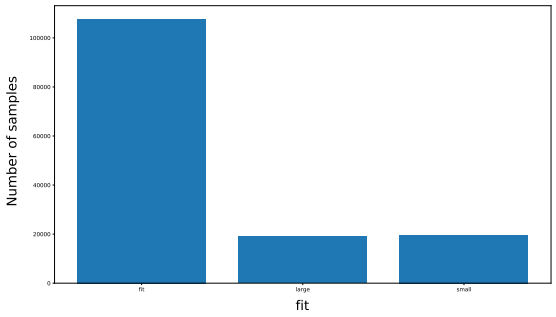

Figure 1: Distribution of fit

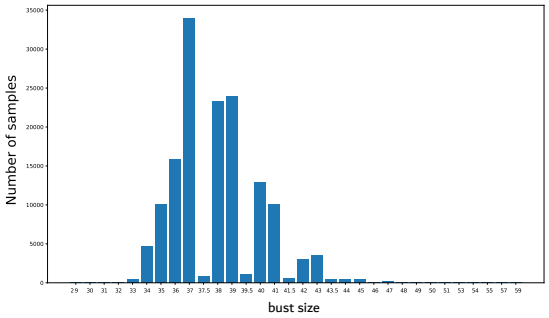

Figure 2: Distribution of bust size

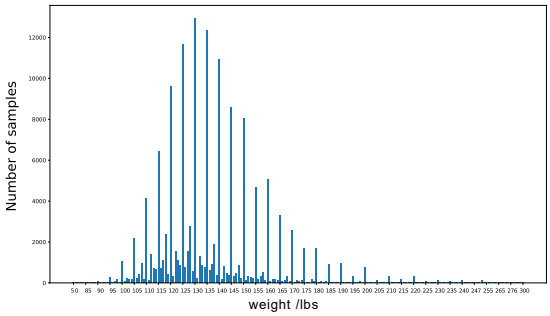

Figure 3: Distribution of weight

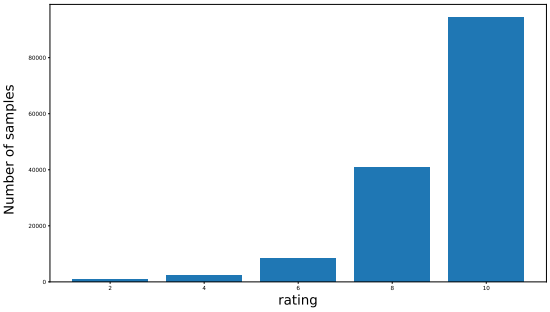

Figure 4: Distribution of rating

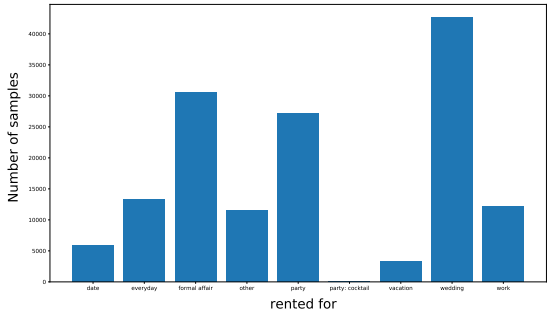

Figure 5: Distribution of rented for

3 RELATED WORK

After exploring the statistics of the dataset, literature investigation is conducted for choosing models to be used for comparison.

There has been substantial academic researches related with rating prediction in the field of recommender systems. Based on published work so far, most recommender systems use collaborative filtering (CF), content-based filtering (CB) and hybrid models to predict review's rating.

The concept of collaborative filtering was introduced by Resnick *et al.*[8]. Their theory was that users like what like-minded users like, where two users were considered like-minded when they rated items alike. When like-minded users were identified, items that one user rated positively were recommended to the other user, and vice versa. And content-based methods provide recommendations by comparing representations of content contained in an item to representations of content that interests the user [6].

Content-based methods can uniquely characterize each user and therefore, are more intuitive, but collaborative filtering still has some key advantages over content-based methods [2].

Firstly, collaborative filtering can perform in domains where there is not much content associated with items, or where the content is difficult for a computer to analyze (e.g. ideas, opinions etc.).

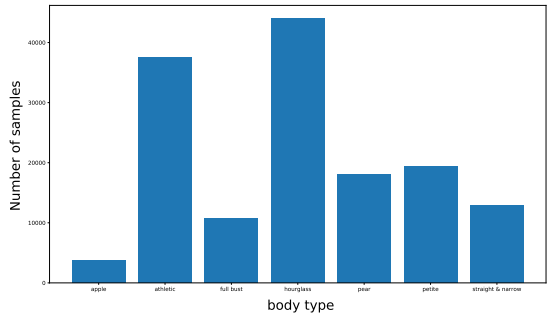

Figure 6: Distribution of body type

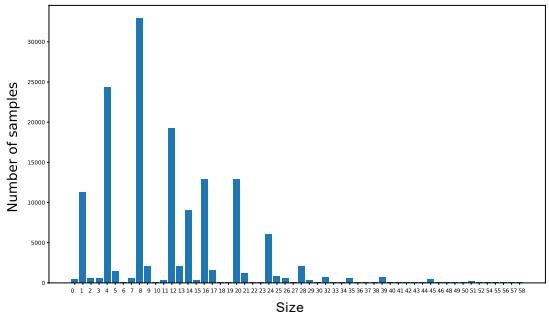

Figure 9: Distribution of size

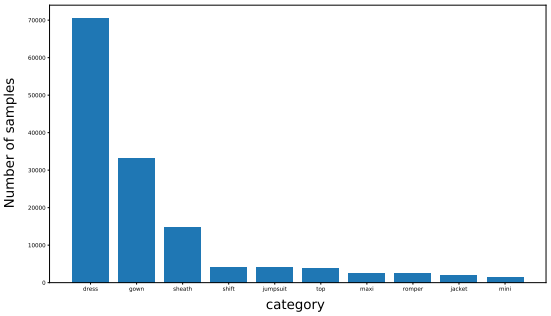

Figure 7: Distribution of category

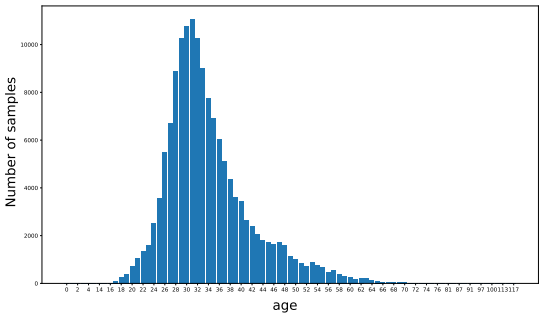

Figure 10: Distribution of age

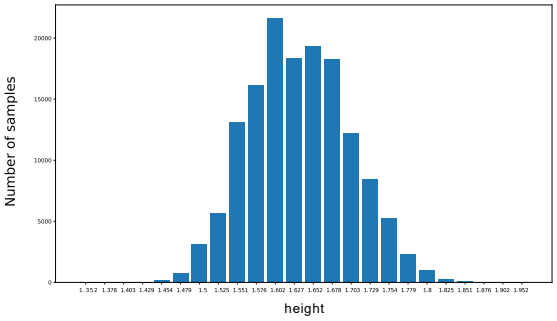

Figure 8: Distribution of height

Secondly a collaborative filtering system has the ability to provide serendipitous recommendations, i.e. it can recommend items that are relevant to the user, but do not contain content from the user's profile. Because of these, collaborative filtering systems have been used fairly successfully to build recommender systems in various domains [1, 8]. However, they suffer from two fundamental problems: sparsity and first rater problem [6].

The collaborative filtering approach can be further divided into two categories: neighborhood-based collaborative filtering and model-based collaborative filtering. Model-based collaborative filtering methods fit parameterized statistical models based on known ratings and make recommendations on the basis of predictions by fitted models [5]. Latent factor models based on singular value decomposition (SVD) are one of the most popular model-based techniques in collaborative filtering [3]. They are also known as matrix factorization models, are one of the feature extractor models used to capture meaningful latent connections between users and items inferred from rating patterns [4]. Latent-factor-based approaches relate users and items in a low-dimensional latent feature space $\mathcal{F} \in \mathbb{R}^d$, where d is the dimension of the latent feature space. $\mathcal{F}$ describes the hidden characteristics of items and users. Furthermore, those interactions between users and items explain the patterns of the ratings since intuitively speaking, if some certain characteristics of an item match a user's interests on those item characteristics, the user tends to rate this item high [11].

While both methods have their own advantages, individually, they fail to provide good recommendations in many situations. Thus, many hybrid models are proposed to ensemble collaborative filtering method and content-based method to overcome their shortcomings and obtain better performance.

In terms of fit predictions, less work are reported in the literature. Some previous work [9, 10] use two latent vector to represent the true size of customers and items, then fit prediction is made based on the similarity of latent vectors. Recently, Rishabh Misra *et al.* [7]

proposed a predictive framework to tackle the product fit problem, which captures the semantics behind customers' feedback, and employs a metric learning technique to resolve label imbalance issues.

4 MODEL SELECTION

In this section, models used for comparison in both tasks are introduced. Different models adopt different learning algorithms or different feature inputs.

4.1 Rating Prediction

Models implemented in the rating prediction task fall into three categories mentioned in the related work above. Linear regression model is one of the most simple and intuitive content-based methods, so we implemented it here as the representative of contented-based method. Then we used latent factor model as the representative of collaborative filtering method. We also implemented a feed-forward neuron network as an alternative collaborative filtering method.

In terms of input, we firstly only use the fit as the feature input of linear regression and user-item interactions as the input of collaborative filtering methods and form three basic models. However, the performance of those models are not quite satisfied. Thus secondly, we use both user-item interactions and fit information as the input of collaborative filtering and form two hybrid models. Although we used the explicit fit information as the content input, we are yet to use the review text which should empirically have high relevance with rating. Thus, we exploit the simple bag-of-words approach to extract implicit information and form the review feature vector. Eventually, we add review feature vector to the input of linear regression and two hybrid models and form the three models with better performance.

4.1.1 Naive Averaging. This can barely be regarded as a learning algorithm. However, we still present it as the most basic baseline. The prediction $\hat{r}$ is obtained through:

$$\hat{r} = \frac{\sum_{r_i \in \mathcal{T}} r_i}{N} \tag{1}$$

where $\mathcal{T}$ is the rating set for training and $N = |\mathcal{T}|$ is the size of rating set.

4.1.2 Linear Regression (LR) with Fit Feature. Since rating prediction is a very classical regression problem, so a reasonable baseline would be linear regression. The prediction $\hat{r}$ is obtained through:

$$\hat{r} = \theta \cdot x_{\text{Fit}} \tag{2}$$

where θ is the weights and x_{Fit} is the one-hot vector encoded with fit information. If the cloth is fit, then $x_{\text{Fit}} = 100$, if cloth is large then then $x_{\text{Fit}} = 010$, and if cloth is small then then $x_{\text{Fit}} = 001$.

4.1.3 Latent Factor Model (LFM). Since each record is a user-item interaction sample and latent factor model is a very classical and effective way to utilize the user-item interaction information. Thus latent model is also implemented here. The prediction $\hat{r}$ is obtained through:

$$\hat{r} = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i \tag{3}$$

where α is the global mean of rating on training set, β_u is the user bias term which represents how would the user tends to rate higher or lower than global mean, β_i is the item bias term which represents how would the item tends to be rated higher or lower than global mean. γ_u and γ_i are latent vectors of user and item.

4.1.4 Feed-forward Neuron Network (FNN). Neuron network is widely used in all subareas of machine learning nowadays. Thus we also try to use a simple feed-forward neuron network to exploit the user-item interactions. The input of the network is the concatenation of latent vectors of user and item. As a very simple one, there are only one or two hidden layers between input and output layers. And the connection between each layer is fully connection, i.e. it's a dense feed-forward neuron network. The prediction $\hat{r}$ is obtained through:

$$\hat{r} = f_{\text{Out}}(\theta_{\text{Out}} \cdot f_{\text{H}}(\theta_{\text{H}} \cdot (\gamma_u \oplus \gamma_i))) \tag{4}$$

where $\gamma_u \oplus \gamma_i$ is the concatenation of latent vectors of user and item, θ_{H} is the hidden layer weights matrix which contains each the weight vector of each hidden neuron. $f_{\text{H}}(\bullet)$ is the activation function of hidden layer. θ_{Out} is the output layer weights vector (since there is only one output i.e. the rating, thus it's vector not matrix), $f_{\text{Out}}(\bullet)$ is the activation function of output layer.

4.1.5 LFM with Fit Feature. In the test, we observe that the performance of latent factor model and FNN is poor if only user-item interactions are feed as input, thus, we come up with the idea to ensemble the latent factor model and linear regression, so that both user-item interactions and explicit or implicit features can be exploited. To compare with linear regression with fit feature vector, we only include the fit feature firstly. Thus the ensemble model is:

$$\hat{r} = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i + \theta \cdot x_{\text{Fit}} \tag{5}$$

All notations in this model are consistent with subsubsection 4.1.2,4.1.3.

4.1.6 FNN with Fit Feature. Adopting additional features are easier in FNN, we simply keep concatenating feature vector with previous input vector. So the model is:

$$\hat{r} = f_{\text{Out}}(\theta_{\text{Out}} \cdot f_{\text{H}}(\theta_{\text{H}} \cdot (\gamma_u \oplus \gamma_i \oplus x_{\text{Fit}}))) \tag{6}$$

4.1.7 LR with Fit and Review Feature. In the existing simple features, fit is the one with highest relevance, however, the prediction performance is not that good if only fit feature is adopted. Review of each record is definitely relevant with the rating, however, it can be used directly as the feature like other simple features. Thus review of each record is converted to the review feature vector through bag-of-words method. The prediction $\hat{r}$ is obtained through:

$$\hat{r} = \theta \cdot (x_{\text{Fit}} \oplus x_{\text{Review}}) \tag{7}$$

where $\oplus$ is the concatenation symbol and x_{Review} is the review feature vector.

4.1.8 LFM with Fit and Review Feature. Eventually, we use user-item interactions, fit and review vector as the input to obtain the best performance. So the prediction $\hat{r}$ is obtained through:

$$\hat{r} = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i + \theta \cdot (x_{\text{Fit}} \oplus x_{\text{Review}}) \tag{8}$$

4.1.9 FNN with Fit and Review Feature. Similarly, the final version of FNN model is:

$$\hat{r} = f_{\text{Out}}(\theta_{\text{Out}} \cdot f_{\text{H}}(\theta_{\text{H}} \cdot (\gamma_u \oplus \gamma_i \oplus x_{\text{Fit}} \oplus x_{\text{Review}}))) \tag{9}$$

4.2 Fit Prediction

For fit prediction, four learning algorithms are used: 1) Logistic Regression 2) Support-Vector Machine 3) Random Forest 4) Gradient Boosting. And two types of features are used: 1) explicit simple features: customers' height, weight, bust size, age, body type and size of cloth 2) implicit features: review feature vector constructed with bag-of-words approach.

Since all features are simply concatenated as a whole input feature vector in all four algorithms, thus, we don't separately introduce models with the same learning algorithm and different input features.

4.2.1 Logistic Regression. Even though we have multi-classes: fit, small and large, we can still use a logistic regression to model the relationship between the features and the labels. The prediction $\hat{r}$ is obtained through:

$$\hat{r} = \sigma(\theta \cdot f) \tag{10}$$

where θ is the weights vector and f is the features vector mentioned above.

4.2.2 Support-Vector Machine. We also tried Support-Vector Machine to find the best separating hyper-plane between the labels which should optimize the number of mislabeled examples.

4.2.3 Random Forest. Random Forest is another classical method for classification problems. It creates decision trees in training time and predicts the labels using those trees. One advantage of using Random Forest is that it avoids over-fitting problem on the training set.

4.2.4 Gradient Boosting. Gradient boosting is a popular classification algorithm that often uses decision trees as well. It creates a combination of weak prediction models. In other words, it trains many models and optimize them in each stage using gradient descent.

5 PERFORMANCE EVALUATION

For performance evaluation, the dataset is split into training set and validation set and the validation size is 1/5 of the whole dataset. Training is conducted on training set and predictions obtained on the validation set are used for performance evaluation.

5.1 Rating Prediction

For rating prediction, we employ the mean squared error (MSE) as the metrics to evaluate the performance of different models. The performance of all models on validation set are shown in Table 2.

The MSE of Naive Averaging is the worst one as expected. In terms of pure content-based methods (i.e. LR with Fit Feature and LR with Fit and Review Feature) and pure collaborative filtering methods (i.e. LFM and FNN), the former one has better performance on this dataset even in case where only fit feature is the input.

Intuitively, this result is not very surprising. Since pure collaborative filtering methods only utilize the user-item interaction, its performance has a heavy dependency on the sparsity of the dataset. From Table 1 we can see that there are about 77k customers and 6k items, while there are onlt 146k total records, which means the whole dataset is extremely sparse. Thus, pure LFM and FNN can hardly extract much information from user-item interactions, which leads to worse performance as compared with pure content-based methods.

Table 2: Performance of models in rating prediction

Models	Validation MSE
Naive Averaging	2.033
Linear Regression (LR) with Fit Feature	1.903
Latent Factor Model (LFM)	1.920
Feed-forward Neuron Network (FNN)	2.031
LFM with Fit Feature	1.805
FNN with Fit Feature	1.901
LR with Fit and Review Feature	1.607
LFM with Fit and Review Feature	1.581
FNN with Fit and Review Feature	1.464

However, after we exploit the fit feature to form the hybrid models (i.e. LFM with Fit Feature, FNN with Fit Feature), we can see the advantage of additional user-item interactions as compared with pure content-based method. Both hybrid models outperform LR with Fit Feature, especially LFM with Fit Feature.

Although hybrid models has better performance than LR, there is still a room for improvement. All user-item interactions and relevant explicit features are used, so we try to exploit the implicit review feature. And we can see that there is a substantial performance improvement. The intuitive explanation is that since the review feature vector we used has 1000 dimensions, it brought a lot of additional information. Moreover, empirically speaking, review feature vector, as the representation of review content, has strong relevance with rating. Thus, adding review feature to three models can bring substantial performance improvement.

Another interesting thing is that, when there are only user-item interactions or both user-item interactions and fit feature as input, LFM can easily outperform FNN with about 0.1 less MSE. However, when review feature is included in the input, FNN outperform LFM with 0.12 less MSE. As far as I'm concerned, this is because dense FNN has a better ability to extract more intricate information if the input is complex enough, since in our implementation, FNN has much more parameters than LFM (about 30x in our implementation). Therefore, when 1000 dimension review feature vector is added, the performance of FNN is much better than LFM.

5.2 Fit Prediction

For fit prediction, we employ the accuracy, weighted average precision, recall and F_1 score as the metrics to evaluate the performance of different models since the distribution of fit, large and small is quite imbalanced. The performance of all models on validation set are shown in Table 3. Note that, by definition weighted recall always equal accuracy, thus not listed in table.

For logistic regression, if uniform class weights are used, it tends to always predict fit. Though its accuracy is not bad, however that's because the dataset is quite imbalanced. And its weighted precision and F_1 score are quite poor. So we used a class weights of {fit = 0.3, small = 1, large = 1.2} to tackle the imbalanced problem in the dataset. Although the accuracy is actually slightly worse as

Table 3: Performance of models in fit prediction

Models	Accuracy	Weighted Precision	Weighted F_1 Score
Logistic Regression with Uniform Class Weights	0.735	0.580	0.625
Logistic Regression with Assigned Class Weights	0.660	0.625	0.640
Support-Vector Machine	0.736	0.541	0.623
Random Forest	0.736	0.541	0.623
Gradient Boosting	0.736	0.682	0.626
Logistic Regression with Review Feature	0.715	0.740	0.724
Support-Vector Machine with Review Feature	0.736	0.541	0.623
Random Forest with Review Feature	0.736	0.541	0.623
Gradient Boosting with Review Feature	0.768	0.746	0.717

compared with uniform class weights, its predictions are more balanced, which leads to higher weighted precision and F_1 score.

Predictions of support-vector machine and random forest looks quite odd on this dataset. If we didn't use class weights for support-vector machine and random forest, they will always try to predict fit. Even if we assigned the same class weights used in Logistic Regression, they still always predict fit. While if keep increasing the imbalance of class weights, it will always predict large or small. Hence, although two methods has 0.736 accuracy, they are not the best options for our dataset.

Among four implemented models, gradient boosting is the one with best accuracy and most balanced predictions.

Like what we did for rating prediction, we also included review feature to see if the performance can be improved. After 300 dimensions of review feature vector is included in input, the accuracy and balance of predictions of logistic regression and gradient boosting are indeed improved. However, the predictions of support-vector machine and random forest are still always fit.

6 CONCLUSION

In this paper, we tackled the rating prediction and fit prediction task on the RentTheRunWay dataset. For rating prediction, we compared the performance of pure content-based filtering methods, pure collaborative filtering methods and hybrid methods with different inputs. From the evaluation we can see that the best approach to do rating prediction on a dataset with this type of record format is using hybrid method that can utilize both content-based feature and user-item interactions. Moreover, other than explicit features, it's also crucial to extract more implicit features and exploit them in the model.

For fit prediction, we experimented with different classification models and features combinations. From the evaluation we can see that support-vector machine and random forest has poor performance on our dataset whereas logistic regression and gradient boost seem to be the better options for this task. Also, after we added the review feature constructed with bag-of-words approach, we had better performance on both logistic regression and gradient boosting. Hence, it's also crucial to exploit those implicit features in models as we concluded in rating prediction.

REFERENCES

[1] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35, 12 (1992), 61–71.

[2] Nathaniel Good, J Ben Schafer, Joseph A Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, John Riedl, et al. 1999. Combining collaborative filtering with personal agents for better recommendations. *AAAI/IAAI* 439 (1999).

[3] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.

[4] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[5] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. 2012. A comparative study of collaborative filtering algorithms. *arXiv preprint arXiv:1205.3193* (2012).

[6] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. 2002. Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai* 23 (2002), 187–192.

[7] Rishabh Misra, Mengting Wan, and Julian McAuley. 2018. Decomposing fit semantics for product size recommendation in metric spaces. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 422–426.

[8] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM, 175–186.

[9] Vivek Sembium, Rajeev Rastogi, Atul Saroop, and Srujana Merugu. 2017. Recommending product sizes to customers. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 243–250.

[10] Vivek Sembium, Rajeev Rastogi, Lavanya Tekumalla, and Atul Saroop. 2018. Bayesian models for product size recommendations. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 679–687.

[11] Jingying Zeng. 2017. *Latent Factor Models for Recommender Systems and Market Segmentation Through Clustering*. Ph.D. Dissertation. The Ohio State University.