Recommender System for Rent the Runway

Jianzhi Li (jl9862), Anshan He (ah4734), Meiyi He (mh5275), Shuting Gu (sg5686)

JAMS, Center for Data Science, New York University

I. Business Understanding

Fashion changes all the time. Bell-bottomed jeans were fashionable in the 1960s, but now we hardly can find a similar item in most clothing stores. People always want to follow the latest trend in fashion, especially the young women. However, they only have limited budget and wardrobe space at home. Rent the Runway, as an E-commerce company that provides designer dresses and accessory rentals, comes to rescue modern women in the U.S since 2008. What makes this company outstanding in the garment industry is its concept of access economy that favors renting rather than buying, which helps people save money while having access to more variety of clothes. However, unlike those successful designers clothing stores such as Saks Fifth Avenue and Barneys New York, who make good personal recommendations to their customers in order to attract them buying more clothes, Rent the Runway did not make much progress in improving the personalized shopping experience for customers. That is why we want to tackle the question of increase customers satisfaction score during online renting and also prevent them from churning, which are essential for the continuous profit and development of the company.

Our project aims to build a recommender system for Rent the Runway to better match customers with the most appropriate items. Specifically, we will first try to predict whether a user would rent an item. Moreover, we will predict the rating to present a ranked list of items given a specific user. And lastly, we will try to classify the item by the occasion customers rent those items for so that we can prompt filtered items to satisfy customers need.

In sum, we are developing models for three tasks:

1) **Rental prediction model**
   Given a pair of (User ID, Item ID), predict whether the user will rent the item.
2) **Rating prediction model**
   Given a user's profile and an item's information, predict the rating that user gives to the item.
   Given a utility matrix filled with ratings for each user-item pair, predict "unknown" ratings.
3) **Category prediction model**
   Given a user's profile and an item's information, predict the category of the item.

The three models will work complementarily with each other and together provide a good prediction for a ranked list of recommended items.

II. Data Understanding

A. Data Collection

We collect the data from an open data source posted on Julian McAuleys personal website, a professor at the University of California, San Diego. The dataset has 192544 rows and 15 columns. Each row is a single transaction of a Rent the Runway user's rental history. For each transaction, the data contains "fit", "user_id", "bust size", "item_id", "weight", "rating", "rented for", "review_text", "body type", "review_summary", "category", "height", "size", "age", and "review_date".

B. Extracted Features

For recommendation task, we solve this problem in two ways. One is to consider the model as a binary problem, whether this user would rent this item or not. The other way is to predict the rating of an item, so that we can later rank the ratings of different items for a specific user and recommend them top-ranked items.

In the binary approach, we applied the collaborative filtering method. We only need to use "user_id" and "item_id" for each transaction to predict whether a user may want to rent another item in the future.

In the rating prediction approach, we further split the model into rating problem based on content-based filtering method and collaborative filtering method. For content-based filtering, we select user's profile, which includes user's age, size, height, weight, bust size, and body type, as well as item's profile, which includes items' category and their rent purpose to build feature vector. The information about whether the item is fit or not given by a user is not included in the group of features used for rating prediction since for an item that has not been rented, it is impossible for us to know users evaluation about its fit. For collaborating filtering, we built a utility matrix with users as rows, items as columns, and ratings as the elements of the matrix.

For the category prediction task, we also tried two different ways, one is based on text mining and another is using the classical content filtering method. In the text mining model, we use the review text and review summary from each transaction. While in the content filtering method, we use the features of each item and user, such as age, height, weight, body type, category, etc. to build different machine learning models.

C. Sparsity and Imbalance

In this dataset, 68% of users only have one record in their renting history, and around 80% users rent fewer than two items. The sparsity of our data is 0.03%, which means only 0.03% entries in the utility matrix have values. As for entries without value, we have no explicit information about the user's preference for the item.

The imbalance dataset is another problem. In our dataset, 64% of the ratings are 10, and the low ratings (2, 4) are very rare. In this situation, the predictive model we developed could be biased and inaccurate. In order to reduce the effect of imbalanced data, we tried the approach of over-sampling, which increases the number of instances in minority classes by randomly replicating them. However, the model we rebuilt using this new approach performed similarly as the original model. Therefore, the imbalance problem remains unsolved and need further discussion.

III. Data preparation

A. Rental prediction model

For the collaborative filtering method of (binary) rental prediction problem, only “user_id” and “item_id” were used. We dropped rows that contain null values. For the target variable here, we only know whether a user has rented an item or not by the transaction history. Hence, we have to manually construct the second half of testing sets by randomly pick a user and then randomly pick an item until this item has not rented by this user before. Thus, the target variable we have for this problem has 50% zero (not rented before, so we assume users wouldnt rent) and 50% one (rented before, so we know it should predict would rent). The test data we made is shown in the Fig.1:

	User ID : Item ID	Rent or not
0	[151496, 466944]	1
1	[820669, 607186]	0
2	[710021, 2493982]	0
3	[770605, 1356952]	0
4	[413753, 858304]	1
5	[613379, 126335]	1
6	[526803, 714374]	1
7	[366539, 1251617]	0

Fig. 1. Part of the data that can be used to test the accuracy of rental prediction model.

B. Rating prediction model

In this section, the target variable is ’rating’,which is numeric and discrete positive values: 2, 4, 6, 8, 10. The distribution of the target variable is shown in in the Fig.2:

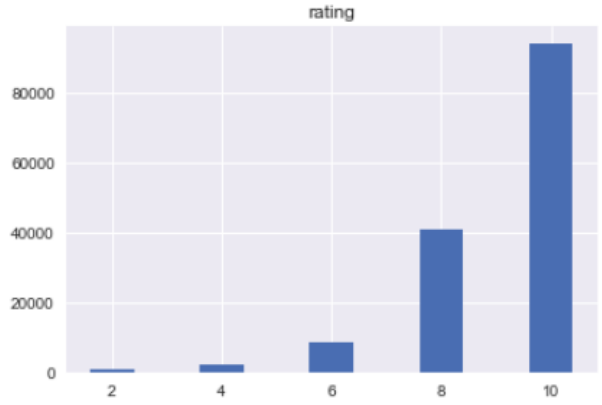

Fig. 2. Distribution of target variable: rating

1) Content filtering approach: We conducted data transformation for the content filtering approach as follows:

1. Dropped null values, and then 146381 transactions remained in the dataset. Variables like “user_id”, “item_id”, “review_date”, “review_text” would be useless in regression methods of this project, so we dropped them as well. Now, we only keep “age”, “bust size”, “body type”, “height”, “weight”, “rented for”, and “size” as our feature variables.

2. The variable “category” is a categorical variable having 68 categories. Some categories may refer to the same kind of items. For example, “leggings” and “legging”, “skirts” and “skirt”, etc. We re-divided the 68 categories into 12 categories according to their similarities. We also put “party: cocktail” in “rented for” to “others” because it has only one count.

3. For all the categorical variables, we did one-hot

encoding so that we can fit it into regression models.
4. For numeric features like “bust size”, “height” and “weight”, we need to transform them as well. We parsed height to usable numerical format and removed ‘lbs’ after weight. We also turned ‘bust size’ into a numeric variable only, and created a new variable named “cups” with values like ‘a’, ‘b’, ‘c’. “cups” is a categorical variable, to be used in regression models, we mapped its values to numerical values.

2) Collaborative filtering approach: For the collaborative filtering approach in the rating prediction part, except the target variable “rating”, only two more variables are needed: “user_id”, “item_id”. The data transformation step conducted here is merely dropping null values.

C. Category prediction model

In this section, the target variable is “rented for”, which represent the rental purpose of a item. The distribution of the target variable is shown in the Fig.3:

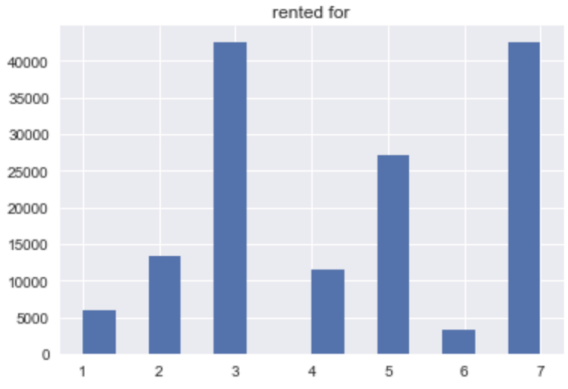

Fig. 3. Distribution of target variable: rented for

1) Content filtering approach: For the content filtering method of the category prediction task, we use “fit”, “bust size”, “weight”, “rating”, “body type”, “category”, “height”, “size”, and “age”as our feature variables. The original dataset has the target variable as categorical data, so we also encoded it into numerical data using one-hot encoding. In total we have seven “rented for” value, which are “date:1”, “every day:2”, “formal affair:3”, “party:5”, “vacation:6”, “wedding:7” and “other:4”.

2) Text Mining approach: For the text mining method of the category prediction task, “review_text” and “review_summary” are the main features. To be more specific, in order to use text as data, we have to somehow represent words as numerical data. First we made all text into lowercase characters, remove stopwords and also perform stemming method to each word. Then we extract the top 5000 frequent words among all reviews, and represent every review as a row in the feature matrix. Each row in the matrix is a single text and each column is one of the most frequent words. We apply the TF-IDF (Term Frequency - Inverse Document Frequency) transformer to this matrix and then this final matrix will be used as our features in this method.

IV. Modeling Evaluation

A. Rental prediction model

Recommending appropriate items to users is the primary goal of our recommender system. We first try to see if we should recommend an item to a user based on the similarity of users and items using the Jaccard Similarity score:

1) Jaccard Similarity: Jaccard similarity is one of the measurement people use in Collaborative Filtering

System. In our recommender system, we measure the similarity between the user and the item to determine if we should recommend this item to a certain user. In our practice, we looked at the transaction history of the item when we determined whether we should recommend an item, finding a list of users who have rented this item before. Then we calculated the cardinality of the intersection of two sets of users' item list divided by the cardinality of the union of two sets. To evaluate the performance of this model, we set a threshold: if the score is above the threshold, we predict that this user would rent this item and vice versa. We test several different thresholds and find out the best threshold is at 0.001, with an accuracy rate is about 0.8209.

B. Rating prediction model

However, since Jaccard Similarity ignores the value of the ratings, it might wrongly regard two users as similar although one of them rated 10 to one item and the other only gave 2 to the same item. Therefore, another possible method is to rank the items by their predicted ratings. Based on the ranked list of items, we can recommend high ranked ones given a user's data. As a regression problem, we choose the Mean Squared Error (MSE) as the metric. Instead of focusing on test MSE only, we looked into both train and test MSE combined with the bias-variance tradeoff, trying to find the model with the best performance.

We firstly predicted ratings by analyzing the features in user profiles and item profiles. We use **linear regression, decision trees, random forests, and gradient boosting**. Furthermore, because the number of features included in our dataset is limited, we alternatively applied collaborative filtering. More precisely, we tried **latent factor models** to predict the unknown ratings based on users previous renting history and item rating. When training models, we utilized 5-fold cross-validation and in Matrix factorization model we also used L2 regularization to prevent overfitting problems.

*1) **Linear Regression**:* Linear regression is the most basic machine learning algorithm to solve regression problems. It is applied as the baseline model to predict ratings in our project. Linear regression tries to find the best fit line and minimize the error of all the points. It performances well when the relationship between dependent variable and independent variables is linear, but it can also be applied when the relationship is not linear. The linearity between target variable and features is not clear in our project, so linear regression is worth a try. We used the default parameters. The best result came when the train MSE is 2.0247 and test MSE is 2.0666.

*2) **Decision Tree**:* Decision Tree learning is another method commonly used in machine learning. The goal is to create a model that predicts the value of a target variable based on several input variables. This model can be used for both classification and regression. Since we are dealing with numeric target variable (rating from 0 to 10), we can apply regression trees here. Decision tree usually performs well on large datasets, so it fits our case well. We first tried the default Decision Tree Regressor with min_sample_split=2 and min_sample_leaf=1 in sklearn to construct the model. It gave us the test MSE of 4.09, which is not good compared to the previous Linear Regression

model. Overfitting of the model might exist with the default setting since the values of min_sample_split and min_sample_leaf are small. In order to solve the overfitting problem and achieve the best performance, we then tried different values for the parameters of max_depth and min_sample_leaf. It turned out that and min_sample_leaf=500 and min_sample_split=5000 returned the minimum train MSE is 2.0213, test MSE is 2.0682. The learning curve is shown in Fig.4 and Fig.5.

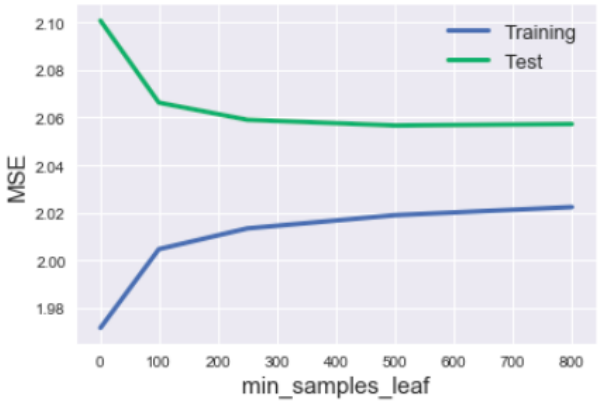

Fig. 4. Finding the best min_sample_leaf for decision trees

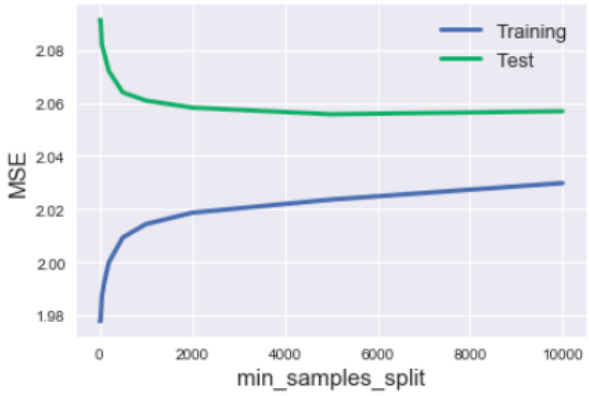

Fig. 5. Finding the best min_sample_split for decision trees

3) ***Random Forests****:* Random Forests is a model built by a multitude of decision trees. For this project, we applied Random Forests here since it can do a better bias-variance tradeoff job via training on different samples of the data and using random subsets of features. For our project, we used RandomForestRegressor in the sklearn package. We tested different numbers (10, 25, 50, 100, 200, 500) for the parameter n_estimators while setting everything else the same as the Decision Tree model. We reached the optimized state when n_estimators=100. At this stage, we got train MSE=1.9530 and test MSE=2.0595.

4) ***Gradient Boosting Classification****:* The main idea of boosting is to modify a weak learner like decision trees and gradient boosting minimizes the loss function while adding trees. There are a lot of high ratings while only a few low ratings in our dataset, so Gradient Boosting might be a good choice to resolve such issue. We introduced GradientBoostingRegressor from sklearn, and we tried different max_depth values like 1,5,10,15,20,30 as well as other parameters, such as n_estimators, min_samples_split, and min_samples_leaf to generate different trees. The best result appeared when train MSE is 2.0145 and test MSE is 2.0653.

5) ***Latent Factor Model****:* Besides factors directly related to features of users and items, the explicit feedback, which contains users' ratings for different items, includes additional information about users' interest and preference to various items. It is reasonable to assume that the rating of an item given by a specific user can be explained by some factors, which can be used to describe the item as well as to measure the user's affinity. The latent factor model can help to relate user and items with those concealed factors.

In this part, we will apply latent factor model to predict "unknown" ratings by using the utility matrix filled with ratings for each user-item pair.

a) *Matrix factorization method:*

Matrix factorization method is a computerized way to extract latent factor without human selection. The inner products of the user-factor matrix and item-factor matrix can explain the interaction between users and items and can be used to approximate the rating that the user would give to that item in Rent the Runway. We establish a small grid search and tune the number of latent factors. The number of latent factors with the best performance is 50.

b) *Modeling bias:*

Biases are effects associated with either users or items. Some users tend to give higher ratings than others. Similarly, some items receive higher ratings than other items. Therefore, we include bias to predict ratings more wisely for different items given by different users. The bias term is $b_{ui} = \mu + b_u + b_i$, where μ is the average rating for all items. b_u and b_i are the observed deviations of user u and item i. The overall average rating is around 9. The maximum value of user bias is 1.77, and the maximum value of item bias is 1.53, which are significant regarding their orders of magnitude, so it is necessary to add biases in our basic matrix factorization model.

c) *Regularization:*

To avoid overfitting, we include L2 regularization. Moreover, the regularized model provided results with smaller differences between training MSE and test MSE. We set up a grid search and tune the regularization parameters. The optimal regularization parameters for user latent factors, item latent factors, user bias, and item bias are 0.75, 0.5, 0.75, and 0.5 respectively.

d) *Stochastic gradient descent:*

Stochastic gradient descent is a fast method to minimize our MSE. Learning rate is an essential parameter in this learning algorithm because it controls the step size. The accuracy would fluctuate sharply if it is too large, while the learning speed will be much lower if it is too small. After grid searching, we finalize the learning rate with value 0.001.

e) *MSE of the latent factor:*

We drew the learning curve of the matrix factorization model in Fig. 6. Given the best group of parameters, the best test error achieves for 100 iterations. When the number of iterations increases to 200, the model starts to overfit. The best performance appears when training MSE is 1.5484 and test MSE is 2.0697.

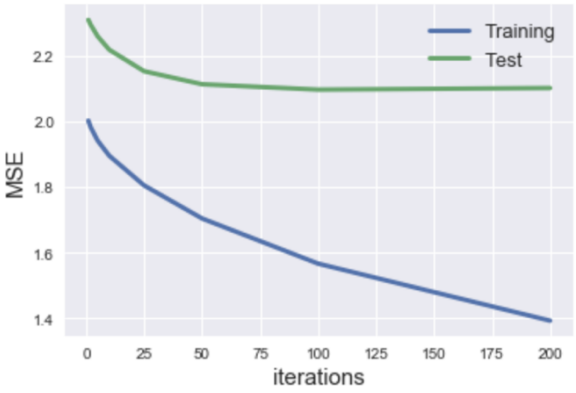

Fig. 6. Finding the best iterations for latent factor model.

C. Category prediction model

1) ***Logistic Regression****:* Similar to Linear Regression, Logistic Regression is also the most basic machine learning algorithm to solve classification problems. Therefore, we choose this model as our baseline model to predict the rental purpose of a specific item. Since we have multiple classes, we use one-vs-rest scheme instead of the regular binary scheme. We use cross-validation method to train models and then use grid search algorithm to tune the regularization parameter (C value) in order to have the best accuracy score for the testing set. When use the features in user profiles and item profiles as features, the best result is at c=0.1, with training accuracy 0.6142 and testing accuracy 0.6145. When use review text and review summary as features, the best result is at c=5, with training accuracy 0.7225 and testing accuracy 0.6972.

2) ***Decision Tree****:* We also tried Decision Tree here since Decision Tree can deal with noisy and incomplete data also it has the ability to select the most significant features. When use the features in user profiles and item profiles as features, the best result is at max_depth = 10, with training accuracy 0.6137 and testing accuracy 0.6140. When use review text and review summary as features, the best result is at max_depth = 15, with training accuracy 0.7118 and testing accuracy 0.6164.

3) ***Random Forest****:* Since Decision Tree often prone to overfitting, we also choose Random Forest model. Also, Random Forest can deal with multiple correlated variables, which very suitable in our case since height and weight for a person might be highly correlated. For this model, when use the user profiles and item profiles as features, the best result is at max_depth = 15, with training accuracy 0.6510 and testing accuracy 0.6148. When use review text and review summary as features, the best result is at max_depth = 40, with training accuracy 0.8357 and testing accuracy 0.6431.

D. Conclusion of model selection

1) Rental prediction model: Collaborative filtering with the Jaccard score as the measurement gave a high and impressive accuracy score, which is 0.8209.

2) Rating prediction model: Table I compares the best train and test MSE of all four regression models and the matrix factorization model. The baseline model, linear regression, is a suitable and traditional model for this regression problem, it might bring some insights into the relationship between the features and rating. Tree models and matrix factorization model also achieved satisfying results, although there are no vast improvements compared with the baseline model regarding the test MSE. Random Forests and Gradient Boosting as the extension of Decision Tree excelled at the performance slightly. Matrix Factorization has the lowest train MSE among all models, which may indicate its potential for further decreasing in test MSE.

TABLE I
RATING PREDICTION MSE TABLE

Model	Train MSE	Test MSE
Linear Regression	2.0247	2.0666
Decision Tree	2.0213	2.0682
Random Forest	1.9530	2.0595
Gradient	2.0145	2.0653
Matrix Factorization	1.5484	2.0697

3) Category prediction model: Table II compares the best train and test accuracy of all three classification

models, and with both content-based features and text mining features. In the table, F1 stands for content based filtering and F2 for text mining. As we can see, text mining do provides a higher accuracy rate in predicting the purpose of rental, since review text and review summary often shows people's feeling towards the item, thus gives more information then just use the user profile and item profile as features.

TABLE II
CATEGORY PREDICTION ACCURACY TABLE

Model	Train Acc.	Test Acc.
Logistic Regression + F1	0.6142	0.6145
Decision Tree + F1	0.6137	0.6140
Random Forests + F1	0.6510	0.6148
Logistic Regression + F2	0.7225	0.6972
Decision Tree + F2	0.7118	0.6164
Random Forest + F2	0.8357	0.6431

V. DEPLOYMENT

A. Concrete deployment for Rent the Runway

Our first step would be suggesting a possible solution to the cold-start and first-rater problems. Since Rent the Runway currently do not have a process of collecting new customers personal information, we propose, when a new user registers an account, Rent the Runway could ask her to fill out a questionnaire to obtain her information about age, size, height, the purpose of rent, etc. In addition, when there are new arrivals, we could also record features of new items. Based on these data, the company could use the rating prediction model we developed to have a rough estimate of which item has higher score given a user and then summarize a list of recommended clothing. Or, even if they do not provide specific data about themselves, we still can recommend them items based on their rental purpose. Later, after the customer has made a few transactions, Rent the Runway can make estimation by using our rental prediction model. This model is faster and more lightweight, while providing better predictions.

However, it is still challenging for Rent the Runway to have enough information on users and items since often time users might skip a few section when filling the review of an item they rented and it is time-consuming to summarize the information for all items in stock. Moreover, there might be other useful features about users and items beyond our imagination. In those case, collaborative filtering could be helpful for Rent the Runway in their recommendation. This model works for all kinds of customers and items and avoids the subjectivity when the company selecting different features to do recommendation.

Our efficient recommender system can be helpful to provide a personalized shopping experience for customers and save their time from searching and viewing all kinds of clothing and accessory.

B. Risk associated

Although this solution is good in improving users' shopping experience, it has the privacy problem of personal/confidential data. Rent the Runway should be really careful in using and protecting customer's personal information.

Besides, inappropriate recommendation could be another risk. Bad prediction happens some times, especially for those new customers who haven't had enough data in the system. Once the company

recommend inappropriate clothes to new users, they are very likely to churn because they are not happy with their very first rental experience. Therefore, when generating recommendation list for new people, we suggest that Rent the Runway make decision more cautiously. For example, selecting items from most popular ones or best rating ones and then run our models.

VI. Code

Our code is available open-source on Github: https://github.com/MeiyiHe/JAMS

VII. Contribution

After we understood the business problem and our data, we separated the work into two groups, each group took on different prediction problem. To be more specific:

- **Meiyi and Jianzhi:**

  Defined rental prediction and category prediction problem, approached these two problem using content based filtering and collaborative filtering method. Then cleaned the data, extracted features for machine learning models, model selection and justification, hyper-parameter tuning (cross-validation) and text mining (TF-IDF).

- **Anshan and Shuting:**

  Defined rating prediction problem, approached the problem using content filtering and collaborative filtering models. Then, did research on recommender system (especially latent factor model), cleaned the data, extracted and adjusted features, selected regression models and latent factor model (matrix factorization), tuned parameters (cross-validation), and evaluated models based on the pre-determined metric.

After that, we put our models together to make comparisons, did model evaluation and discussed the deployment.

References

[1] M. Rishabh, W. Mengting and M. Julian, Decomposing Fit Semantics for Product Size Recommendation in Metric Spaces, RecSys, 2018.

[2] L. Jure, R. Anand and U. Jeff, Chapter 9, Mining of Massive Datasets, Cambridge University Press, 2014, http://www.mmds.org/.

[3] L. Jure, Lecture Notes of CS246: Mining Massive Datasets, https://web.stanford.edu/class/cs246/handouts.html.

[4] B. Jason, A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning, XGBoost, 2016, https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/.

[5] K. Yehuda, B. Robert and V. Chris, Matrix Factorization Techniques for Recommendation Systems, IEEE Computer Society, 2009, https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf/.

[6] Explicit Matrix Factorization: ALS, SGD, and All That Jazz, Insight Data, 2016, https://blog.insightdatascience.com/explicit-matrix-factorization-als-sgd-and-all-that-jazz-b00e4d9b21ea.