An Evaluation Metric for Content Providing Models, Recommendation Systems, and Online Campaigns

Kourosh Modarresi(✉) and Jamie Diner(✉)

Adobe Inc., San Jose, USA
kouroshm@alumni.stanford.edu, jamdin@gmail.com

Abstract. Creating an optimal digital experience for users require providing users desirable content and also delivering these contents in optimal time as user's experience and interaction taking place. There are multiple metrics and variables that may determine the success of a "user digital experience". These metrics may include accuracy, computational cost and other variables. Many of these variables may be contradictory to one another (as explained later in this submission) and their importance may depend on the specific application the digital experience optimization may be pursuing. To deal with this intertwined, possibly contradicting and confusing set of metrics, this work introduces a generalized index entailing all possible metrics and variables – that may be significant in defining a successful "digital experience design model". Besides its generalizability, as it may include any metric the marketers or scientists consider to be important, this new index allows the marketers or the scientists to give different weights to the corresponding metrics as the significance of a specific metric may depends on the specific application. This index is very flexible and could be adjusted as the objective of" user digital experience optimization" may change.

Here, we use "recommendation" as equivalent to "content providing" throughout the submission. One well known usage of "recommender systems" is in providing contents such as products, ads, goods, network connections, services, and so on. Recommender systems have other wide and broad applications and – in general – many problems and applications in AI and machine learning could be converted easily to an equivalent "recommender system" one. This feature increases the significance of recommender systems as an important application of AI and machine learning.

The introduction of internet has brought a new dimension on the ways businesses sell their products and interact with their customers. Ubiquity of the web and consequently web applications are soaring and as a result much of the commerce and customer experience are taking place on line. Many companies offer their products exclusively or predominantly online. At the same time, many present and potential customers spend much time on line and thus businesses try to use efficient models to interact with online users and engage them in various desired initiatives. This interaction with online users is crucial for businesses that hope to see some desired outcome such as purchase, conversions of any types, simple page views, spending longer time on the business pages and so on.

J. M. F. Rodrigues et al. (Eds.): ICCS 2019, LNCS 11537, pp. 550–563, 2019.
https://doi.org/10.1007/978-3-030-22741-8_39

Recommendation system is one of the main tools to achieve these outcomes. The basic idea of recommender systems is to analyze what is the probability of a desires action by a specific user. Then, by knowing this probability, one can make decision of what initiatives to be taken to maximize the desirable outcomes of the online user's actions. The types of initiatives could include, promotional initiatives (sending coupons, cash, …) or communication with the customer using all available media venues such as mail, email, online ad, etc. the main goal of recommendation or targeting model is to increase some outcomes such as "conversion rate", "length of stay on sites", "number of views" and so on. There are many other direct or indirect metrics influenced by recommender systems. Examples of these could include an increase of the sale of other products which were not the direct goal of the recommendations, an increase the chance of customer coming back at the site, increase in brand awareness and the chance of retargeting the same user at a later time.

Keywords: Recommendation systems · Machine learning · Artificial intelligence

1 The Formulation of a New Metric for Recommendation Systems

1.1 An Overview of This Work

At first, we demonstrate the problem we want to address, and we do it by using many models, data sets and multiple metrics. Then, we propose our unified and generalized metric to address the problems we observe in using different multiple and separate metrics.

Thus, we use several models and multiple data sets to evaluate our approach. First, we use all these data sets to evaluate performances of the different models using different performance metrics which are "the state of the art". Then, we are observing the difficulties of any evaluation using these performance metrics. That is because dealing with different performance metrics, which often make contradictory conclusion, it'd be hard to decide which model has the best performance (so to use the model for the targeting campaign in mind). Therefore, we create our performance index which produces a single, unifying performance metric evaluation a targeting model.

1.2 The Data

Machine learning is the science of data driven modeling approach and as such, there is no one single model could work for all types of data [11–25]. As explained above, we use multiple of models, since there are models that may work the best only on some specific data sets. Also, to use as many diverse data sets, for this work, we use 15 publicly available data sets. The features of the datasets are described in Table 1.

To create a higher degree of variation in the choice of data, we created 100 bootstrap samples from each of the datasets by selecting a random number of rows, a random number of columns, and a random number of missing values. The number of

Table 1. The features of some of the different data sets used in this work

Name	numRows	numCols	numNumericalCol	numCategoricalCol
abalone	4176	9	8	1
air_quality	9357	13	13	0
batting	947	16	16	0
bike	731	14	9	5
boston	506	14	13	1

missing values for the rrecsys package was selected at random between 1% and 97%, while the number of missing values for mice, softImpute, missMDA, missForest, and impute was selected between 1% and 75%, as higher number of missing values caused runtime errors. The number of missing values for the Amelia package was selected between 1% and 20%, with only 20 bootstrap samples per dataset. The reason for this was because running with more sparse data or a larger number of bootstraps caused segmentation errors that crashed R. Before any analysis, we can see that for very sparse matrices, the all packages except rrecsys are not a good option.

1.3 The Models for Recommendation Systems

Several models in seven packages, rrecsys, mice, Amelia, softImpute, missMDA, missForest, and impute, all in R, have been used to compare the results. These models are [44, 46, 53, 55, 59–61]:

Rrecsys: The package rrecsys had the following models implemented:

1. **itemAverage**: impute the average rating of the item in the missing values
2. **userAverage**: impute the average rating of the user in the missing values
3. **globalAverage**: impute the overall average rating in the missing values
4. **IBKNN**: imputes the weighted average of the k-nearest neighbors of the item in the missing values

Mice: In mice, we varied the imputation method available for **numerical** variables with the following:

1. **pmm**: imputes the predictive mean matching in the missing values. One of the biggest advantages is that it maintains the distribution of the data, and one of the biggest disadvantages is that it does not handle well very sparse matrices.
2. **norm**: fits a bayesian linear regression model and imputes the data by predicting missing values.
3. **norm.nob**: fits a linear regression model that ignores the model error.
4. **mean**: imputes the mean of the column in the missing samples.
5. **sample**: finds a random item that is not missing in the column where there is a missing value and imputes it.

Amelia: The Amelia package works by performing multiple imputation on bootstrap samples based on the EMB algorithm (Expectation-Maximization with Bayesian Hierarchical classification). It allows for multiprocessing, making it fast and robust. It makes the following assumptions:

1. All the variables have Multivariate Normal Distribution.
2. Missing data is random (MAR).

softImpute [44, 46]: The package softImpute implements matrix completion using nuclear norm regularization. It offers two algorithms:

1. **SVD**: iteratively computes the soft-thresholded Singular Value Decomposition (SVD) of a filled in matrix
2. **ALS**: uses alternating ridge regression to fill in the missing entries.

missMDA: The package missMDA imputes the missing values of a mixed dataset (with continuous and categorical variables) using the principal component method "factorial analysis for mixed data" (FAMD). It implements 2 methods:

1. **EM**: Expectation Maximization algorithm where missing values are imputed with initial values such as the mean of the variable for the continuous variables and the proportion of the category for each category using the non-missing entries. Then, it performs the FAMD algorithm on the completed dataset until convergence.
2. **Regularized**: Similar to the EM but adds a regularization parameter to avoid overfitting.

missForest: The package missForest creates one Random Forest model for each of the columns of the dataset, where the training set is given by the non-missing rows. Then, it uses that model to impute the missing values. It iterates until convergence. Even though Random Forest may deal with missing values in theory, this implementation first imputes the mean of the column into the missing values.

Impute: The package impute only has one method that fills in missing values using the K-nearest neighbors. It only works on *numerical* datasets.

1.4 The Results

In this section, we use the performance (execution time and accuracy) of the models (Sect. 1.3) with respect to different variables such as size and sparsity of the data sets (Sect. 1.2).

Execution Time: We want to compare the computational complexity of the packages because it is of interest to find if a model will perform well when scaling.

We can see that two packages stand out from the rest, missMDA and missForest, followed by softImpute. This is an expected result, as both packages create one model for each column of the dataset and then iterates into convergence. Because we need to create $k \times m$ (*number of iterations* $\times$ *number of columns*), the execution time is greater for these models.

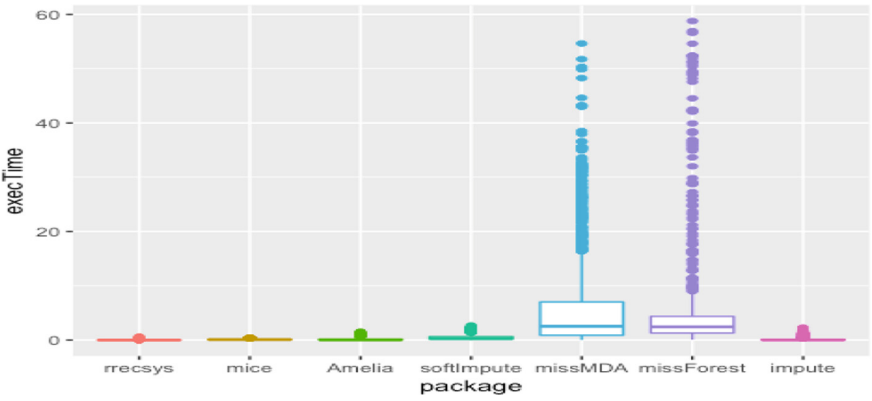

Fig. 1. We can see the comparison of the execution time (in seconds) of the packages.

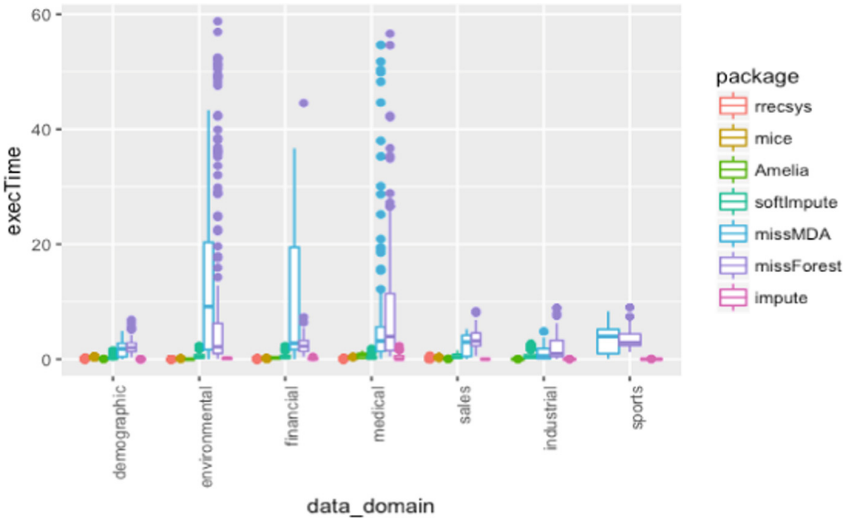

Fig. 2. This figure displays the execution time (in seconds) by domain and package.

We can see that missMDA and missForest stand out from the rest in all of the packages regardless of the domain. As expected, the domains with the largest datasets (environmental, financial, and medical) took the longest time to run. The high variance in some of the runs is because the execution time increased exponentially in some models, so small datasets got executed relatively fast while large datasets took an exponentially larger amount of time to execute.

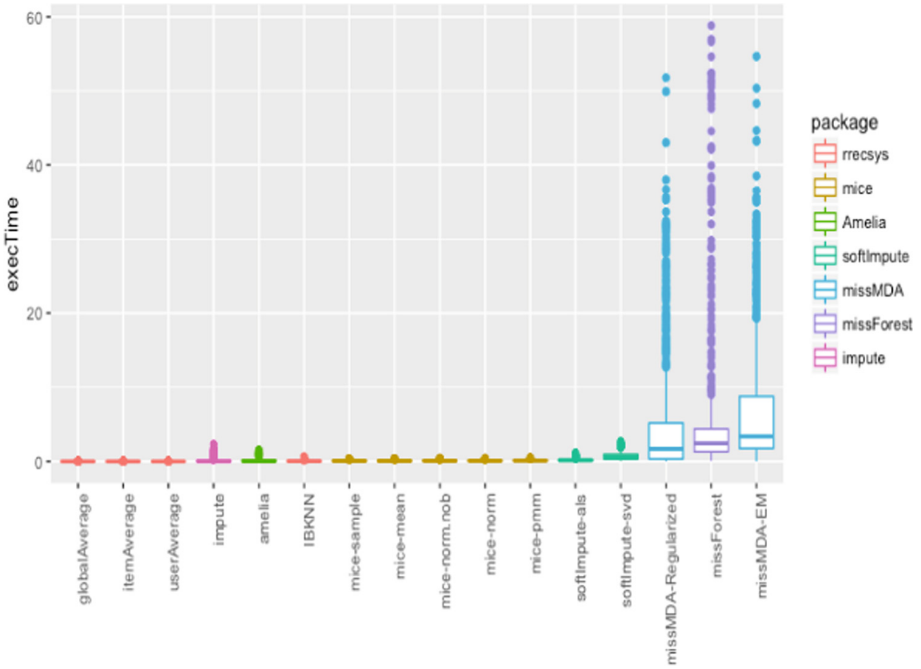

Fig. 3. The comparison of the execution time (in seconds) by model and package.

We can see that the **EM** model from missMDA was the one that took the longest to execute, followed by **missForest** and the **Regularized** method from missMDA and the **SVD** method from softImpute. This is in line with the results obtained in the previous points.

Accuracy Performance: We measure the error using the Normalized Root Mean Squared Error (NRMSE), which is similar to the RMSE but it divides the error by the variance of the dataset. The reason we used this metric is to be able to compare the performance of difference datasets regardless of the range or variance it has.

$$NRMSE = \sqrt{mean\left(\left(x_{true} - x_{pred}\right)^2\right)/var(x_{true})}$$

Model Performance with Respect Sparsity: Below we can see how the model's performance changed by how sparse the matrix was in every domain.

Here, the lines represent a fitted linear model and the shaded region the 95% confidence interval. We can see that the difference in performance between the models is not statistically significant in the demographic, medical, and environmental domain for most levels of sparsity. We can see that the worst performing model in the environmental and financial domain is IBKNN for all levels of sparsity, while the missMDA-EM is the worst performing model for the sales, industrial and sports

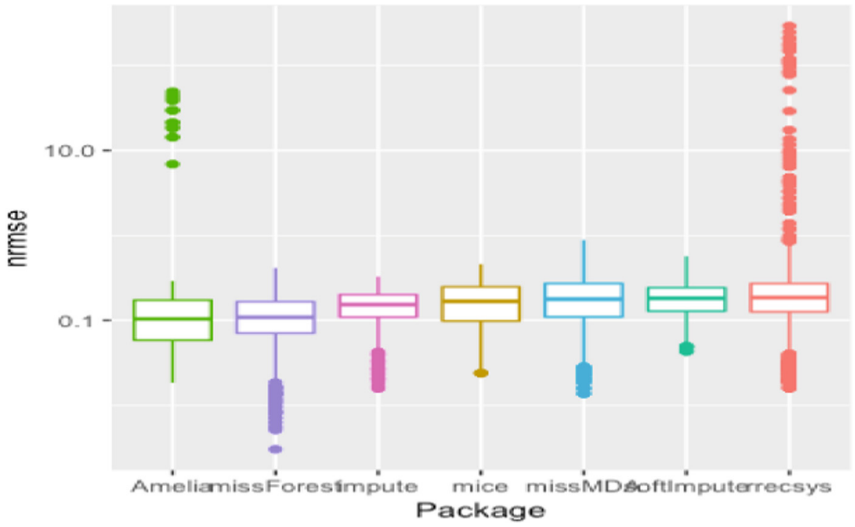

Fig. 4. We can see that all packages have very similar performance, with Amelia being the best followed by missForest and impute. It is important to note that Amelia took only a fraction of the time of missForest to execute, so it might be better for large datasets. However, the Amelia package only worked for datasets that were less than 20% sparse, so it is not possible to compare the performance as having more non-missing values improves the performance of the model.

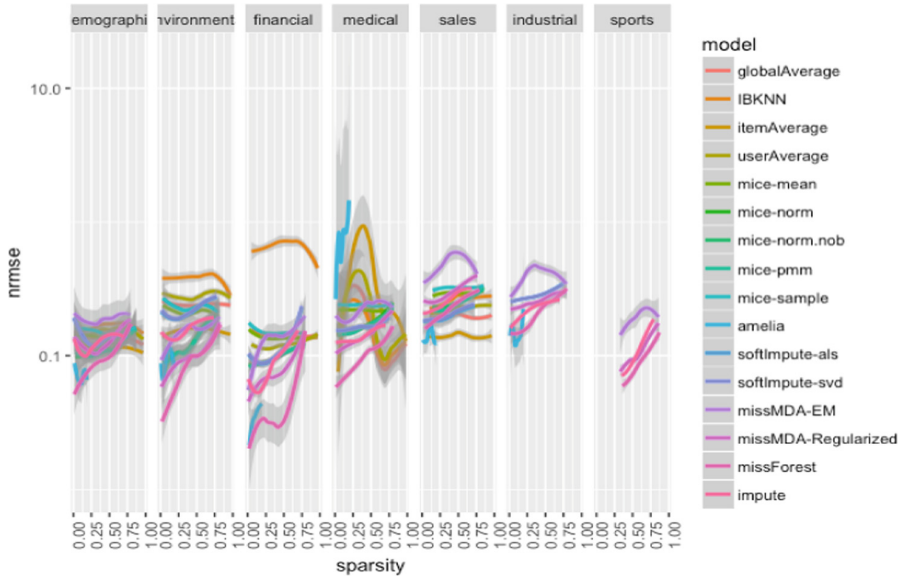

Fig. 5. As we can see, the performance of any model is a function of the sparsity of data.

domains with a statistically significant difference. The best performing model in most of the domains and sparsity levels is the missForest, which is an expected result as Random Forest work extremely well in terms of accuracy but are not scalable for large datasets. We can also see an upward trend in most of the models that shows that the model's performance decline if the matrix is sparser. This is an expected result, as having a sparser matrix implies a smaller training sample size and may cause the model to have high bias.

Model Performance with Respect to the Size of the Data: Below we can see how the model's performance (measured by the median NRMSE) is a function of the size of the data matrix.

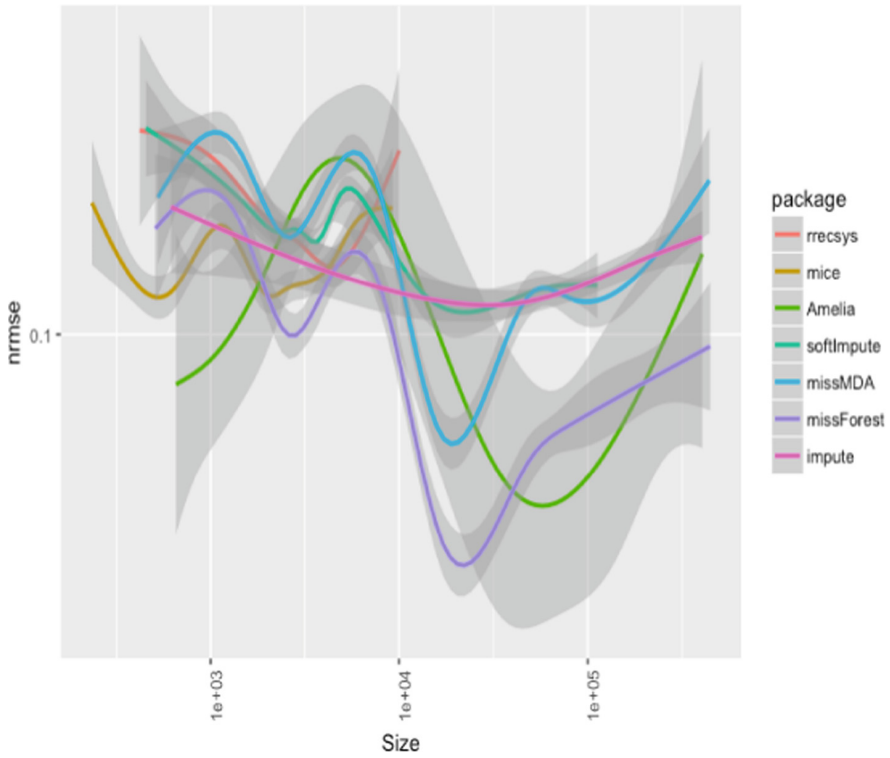

Fig. 6. Also, as it can be seen, the accuracy of any model depends on the size of the data as this figure demonstrates that point.

Execution Time with Respect to the Size of the Data: Below we can see how the model's performance (measured by the execution time in seconds) changed by size the matrix (rows $\times$ columns).

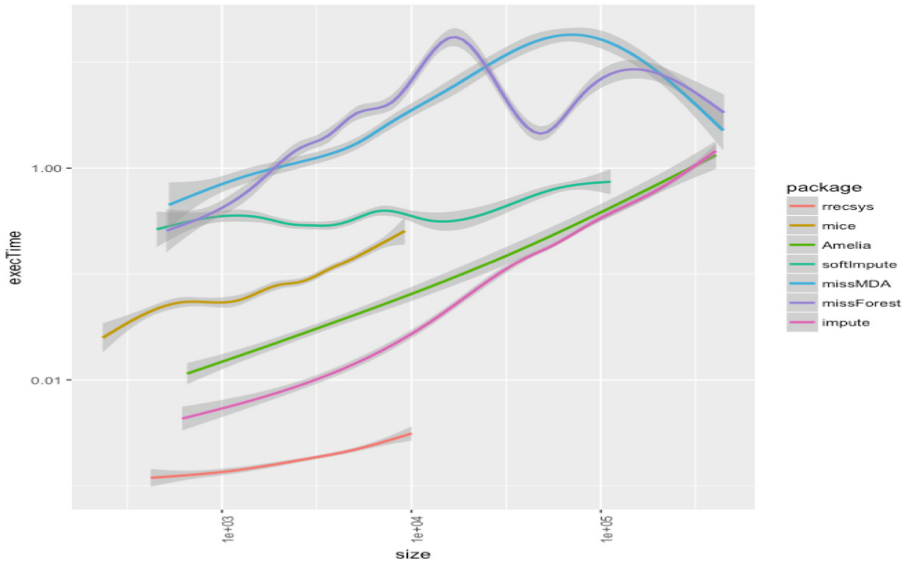

Fig. 7. This figure shows that all packages experience an upward trend, where increasing the size of the matrix increases the execution time which is an expected result. We can see that rrecsys is the package that executes the fastest, followed by impute, Amelia, and mice. At the top, we can see that the softImpute, missMDA, and missForest take considerably more time to execute in any size of the dataset. All these differences are statistically significant for all matrix sizes.

It is interesting that the trend in the Amelia package appear to be completely linear with a positive slope, while the softImpute has the smallest slope and therefore the smallest change in execution time by change in the size of the dataset.

All these issues point to the difficulty of evaluating of the performance of any content providing (recommendation) model and thus we need to have a generalized and physically interpretable measure in evaluation of all these models. This is the reason this work is providing such a metric, we call it GRMM.

1.5 The New Recommendation Metric, Generalized Recommender Model Metric (GRMM)

As we observed in Sect. 1.4, a major problem in the building and implementation of recommender systems (or content providing model) is the problem of evaluation of such systems. The reasons for the issue of evaluation is due to the presence of many various – and sometime conflicting – performance criteria. As an example, when considering accuracy as one of the performance criteria, we could have a recommender system that may be highly accurate for some quantitative data sets while displaying less desirable accuracy for non-quantitative (qualitative) data sets and so this makes it difficult to come up with a clear description for the accuracy of the recommender system model and/or implementation. The same is true for another important

performance measure for the recommender systems, i.e., the execution time. Given all different (and often conflicting) performance measures, it is very difficult to have a correct evaluation of any recommender system.

This work addresses the problem by creating a unified performance measure for Recommender Systems, "Generalized Recommender Model Metric, GRMM". This metric is a normalized measure between 0–1 with 1 to be the optimal value or the highest performance measure index. This is an extremely useful tool for an AI and machine learning scientist (model design), also for marketer and engineer (in charge of implementation of the recommender systems as different system implementation have different performances also) to have a single number indicating the overall performance of any recommender system, including and encompassing all possible performance measures and sub-measures. Thus, this new recommender system index (GRMM) helps all involved in the process of designing, implementing and using recommender systems to have a single number (from 0–1) to evaluate the performance of their systems and to choose the best system (of the highest GRMM metric).

This index metric, GRMM is a flexible and most generalized index allowing everyone dealing with content creation and content providing (scientists, marketers, consultants, engineers and so on) to evaluate the model of providing content using any specific metric or a set of metrics and by giving correct weights (reflecting the business importance).

In general, for any data set X, we could have many performance measures such as accuracy (for qualitative data, accuracy for quantitative data, and so on execution time (as a function of sparsity or dimensions and so on).

In the most general case, for any recommender system X, we could have as many performance measures that may be required, let's say "n" different measures m_i for i = 1:n i.e., $\mathrm{M(X)} = [m_1(X), m_2(X), \ldots, m_n(X)]$.

It is important to mention that each performance measure of m_i(X) is a sigmoid function with a continuous value of zero to one. As an example, if $m_1(X)$ is the accuracy measure for the recommender system X, then,

$$m_1(X) = \frac{1}{1 + e^{-\sum_{j=1}^{s} b_j a_j(X)}}$$

Where each function $a_j(X)$ is the specific accuracy (sub-measure of accuracy) performance of the corresponding recommender system model with respect to the factor j. There could be as many as "s" different sub-measure for accuracy or factors in here such as accuracy of the recommender system for continuous or quantitative variables, or the accuracy of the same recommender system for qualitative variables, accuracy with respect to size, sparsity and so on. The coefficients b_j are weights given (by scientists, marketers or engineers) to each sub-measure and are normalized,

$$\sum_{j=1}^{s} b_j = 1$$

Each coefficient b_j could be chosen depending on the specific application and/or specific weights chosen for each factor (accuracy factor for this measure, $m_1(X)$).

Thus, for any given recommender system X, our Generalized Recommender Model Metric, GRMM(X) is,

$$GRMM(X) = \frac{n}{\frac{1}{m_1(X)} + \frac{1}{m_2(X)} + \ldots + \frac{1}{m_n(X)}}$$

Though, a marketer, AI scientist or an engineer may like to give different weights to different performance measures, m_i(X). This is done by considering the corresponding weights, w_i for each performance measure of $m_i(X)$. We have these weights to be normalized so that,

$$\sum_{i=1}^{n} w_i = 1$$

Hence, the most general form of our index, GRMM, becomes,

$$GRMM(X) = \frac{1}{w_1 * \frac{1}{m_1(X)} + w_2 * \frac{1}{m_2(X)} + \ldots + w_n * \frac{1}{m_n(X)}}$$

This is the most general form of our index. This completely covers all possible performance measures (n number of them, with n to be any natural number larger or equal to 1) for any given recommender system. Also, the index includes the weights w_i to allow the marketers, scientists or engineers to choose the corresponding significance or weights for each performance measure. In this most general case, we also consider having different weights for any sub- performance measure. As explained above, for the example of accuracy measure, the weights b_j gives also the same flexibility for the marketers, scientists and engineers to give different importance or weights for different accuracy measures of any recommender system.

Surely, this is the most generalized version of GRMM index.

There is always value for simplicity and brevity in creating any model. That is the reason we tried to find ways to make the index a bit simpler yet covering the main points and performance measure and sub-measures. Thus, in our analysis, we have found that the pseudo-optimal value for n is 2 with the two measures to be accuracy and execution time (time complexity). We also found it's also very good pseudo-optimal choice of setting all weights, *to be equal. That is,*

$$w_i = \frac{1}{n}$$

That is also the same procedure, that for all suboptimal measure the practical value of all weights is to be equal. For the example above, for accuracy measure and its sub-measures,

$$b_j = \frac{1}{s}$$

Obviously, we have the option of using both the most general case of our GRMM index or the "simplified version". The latter version has some extra explanations and descriptions in the original submission.

GRMM is a new and unique metric addressing a major shortcoming in this domain of content providing and content offering. It is physically interpretable and understandable and easy to use for all individuals including marketers, product managers, engineers and scientists.

References

1. Bjorck, A.: Numerical Methods for Least Squares Problems. SIAM, Philadelphia (1996)
2. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
3. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann (1998)
4. Cai, J.-F., Candès, E.J., Shen, Z.: A singular value thresholding algorithm for matrix completion. SIAM J. Optim. **20**(4), 1956–1982 (2008)
5. Candès, E.J., Recht, B.: Exact matrix completion via convex optimization. Found. Comput. Math. **9**, 717–772 (2008)
6. Candès, E.J.: Compressive sampling. In: Proceedings of the International Congress of Mathematicians, Madrid, Spain (2006)
7. Chen, P.-Y., Wu, S.-Y., Yoon, J.: The impact of online recommendations and consumer feedback on sales. In: Proceedings of the 25th International Conference on Information Systems, pp. 711–724 (2004)
8. Cho, Y.H., Kim, J.K., Kim, S.H.: A personalized recommender system based on web usage mining and decision tree Induction. Expert Syst. Appl. **23**, 329–342 (2002)
9. Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., Sartin M.: Combining content-based and collaborative filters in an online newspaper. In: Proceedings of the ACM SIGIR 1999 Workshop on Recommender Systems (1999)
10. Çoba, L., Zanker, M.: rrecsys: an R-package for prototyping recommendation algorithms. In: RecSys 2016 Poster Proceedings (2016)
11. Abalone Data Set. https://archive.ics.uci.edu/ml/datasets/abalone
12. Air Quality Data Set. https://archive.ics.uci.edu/ml/datasets/Air+Quality
13. Batting Data Set. http://www.tgfantasybaseball.com/baseball/stats.cfm
14. Bike Sharing Dataset Data Set. https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset
15. Boston Data Set. https://archive.ics.uci.edu/ml/datasets/housing
16. Physicochemical Properties of Protein Tertiary Structure Data Set. https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Structure
17. Data: Census: click on the "Compare Large Cities and Towns for Population, Housing, Area, and Density" link on Census 2000. https://factfinder.census.gov/faces/nav/jsf/pages/community_facts.xhtml
18. Concrete Compressive Strength Data Set. https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength
19. ISTANBUL STOCK EXCHANGE Data Set. https://archive.ics.uci.edu/ml/dtasets/ISTANBUL+STOCK+EXCHANGE#
20. Parkinsons Data Set. https://archive.ics.uci.edu/ml/datasets/parkinsons

21. S&P 500 Index Options. http://www.cboe.com/products/stock-index-options-spx-rut-msci-ftse/s-p-500-index-options/s-p-500-index/spx-historical-data
22. seeds Data Set. http://archive.ics.uci.edu/ml/datasets/seeds
23. Waveform Database Generator (Version 1) Data Set. https://archive.ics.uci.edu/ml/datasets/Waveform+Database+Generator+(Version+2)
24. Breast Cancer Wisconsin (Prognostic) Data Set. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Prognostic%29
25. Yacht Hydrodynamics Data Set. http://archive.ics.uci.edu/ml/datasets/yacht+hydrodynamics
26. d'Aspremont, A., El Ghaoui, L., Jordan, M.I., Lanckriet, G.R.G.: A direct formulation for sparse PCA using semidefinite programming. SIAM Rev. **49**(3), 434–448 (2007)
27. Davies, A.R., Hassan, M.F.: Optimality in the regularization of ill-posed inverse problems. In: Sabatier, P.C. (ed.) Inverse Problems: An Interdisciplinary Study. Academic Press, London (1987)
28. DeMoor, B., Golub, G.H.: The restricted singular value decomposition: properties and applications. SIAM J. Matrix Anal. Appl. **12**(3), 401–425 (1991)
29. Donoho, D.L., Tanner, J.: Sparse nonnegative solutions of underdetermined linear equations by linear programming. Proc. Natl. Acad. Sci. **102**(27), 9446–9451 (2005)
30. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. Ann. Stat. **32**, 407–499 (2004)
31. Elden, L.: Algorithms for the regularization of ill-conditioned least squares problems. BIT **17**, 134–145 (1977)
32. Elden, L.: A note on the computation of the generalized cross-validation function for ill-conditioned least squares problems. BIT **24**, 467–472 (1984)
33. Engl, H.W., Hanke, M., Neubauer, A.: Regularization methods for the stable solution of inverse problems. Surv. Math. Ind. **3**, 71–143 (1993)
34. Engl, H.W., Hanke, M., Neubauer, A.: Regularization of Inverse Problems. Kluwer, Dordrecht (1996)
35. Engl, H.W., Groetsch, C.W. (eds.): Inverse and Ill-Posed Problems. Academic Press, London (1987)
36. Gander, W.: On the linear least squares problem with a quadratic Constraint. Technical report STAN-CS-78-697, Stanford University (1978)
37. Golub, G.H., Van Loan, C.F.: Matrix computations. In: Computer Assisted Mechanics and Engineering Sciences, 4th edn. Johns Hopkins University Press, US (2013)
38. Golub, G.H., Van Loan, C.F.: An analysis of the total least squares problem. SIAM J. Numer. Anal. **17**, 883–893 (1980)
39. Golub, G.H., Kahan, W.: Calculating the singular values and pseudo-inverse of a matrix. SIAM J. Numer. Anal. Ser. B **2**, 205–224 (1965)
40. Golub, G.H., Heath, M., Wahba, G.: Generalized cross-validation as a method for choosing a good ridge parameter. Technometrics **21**, 215–223 (1979)
41. Guo, S., Wang, M., Leskovec, J.: The role of social networks in online shopping: information passing, price of trust, and consumer choice. In: ACM Conference on Electronic Commerce (EC) (2011)
42. Häubl, G., Trifts, V.: Consumer decision making in online shopping environments: the effects of interactive decision aids. Market. Sci. **19**, 4–21 (2000)
43. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning; Data mining, Inference and Prediction. Springer, New York (2001). https://doi.org/10.1007/978-0-387-84858-7
44. Hastie, T., Mazumder, R.: Matrix completion via iterative soft-thresholded SVD (2015)
45. Hastie, T., Tibshirani, R., Narasimhan, B., Chu, G.: Package 'impute'. CRAN (2017)
46. Honaker, J., King, G., Blackwell, M.: Amelia II: a program for missing data (2012)

47. Hua, T.A., Gunst, R.F.: Generalized ridge regression: a note on negative ridge parameters. Commun. Statist. Theory Methods **12**, 37–45 (1983)
48. Iyengar, V.S., Zhang, T.: Empirical study of recommender systems using linear classifiers. In: Cheung, D., Williams, G.J., Li, Q. (eds.) PAKDD 2001. LNCS (LNAI), vol. 2035, pp. 16–27. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45357-1_5
49. Jeffers, J.: Two case studies in the application of principal component. Appl. Stat. **16**, 225–236 (1967)
50. Jolliffe, I.: Principal Component Analysis. Springer, New York (1986). https://doi.org/10.1007/b98835
51. Jolliffe, I.T.: Rotation of principal components: choice of normalization Constraints. J. Appl. Stat. **22**, 29–35 (1995)
52. Jolliffe, I.T., Trendafilov, N.T., Uddin, M.: A modified principal component technique Based on the LASSO. J. Comp. Graph. Stat. **12**, 531–547 (2003)
53. Josse, J., Husson, F.: missMDA: a package for handling missing values in multivariate data analysis. J. Stat. Softw. **70**(1), 1–31 (2016)
54. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. Internet Comput. **7**(1), 76–80 (2003)
55. Mazumder, R., Hastie, T., Tibshirani, R.: Spectral regularization algorithms for learning large incomplete matrices. JMLR **2010**(11), 2287–2322 (2010)
56. McCabe, G.: Principal variables. Technometrics **26**, 137–144 (1984)
57. Modarresi, K.: A local regularization method using multiple regularization levels, Stanford, April 2007
58. Modarresi, K., Golub, G.H.: An efficient algorithm for the determination of multiple regularization parameters. In: Proceedings of Inverse Problems Design and Optimization Symposium (IPDO), 16–18 April 2007, Miami Beach, Florida, pp. 395–402 (2007)
59. Modarresi, K., Diner, J.: An efficient deep learning model for recommender systems. In: Shi, Y., et al. (eds.) ICCS 2018. LNCS, vol. 10861, pp. 221–233. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93701-4_17
60. Modarresi, K.: Recommendation system based on complete personalization. Procedia Comput. Sci. **80**, 2190–2204 (2016)
61. Modarresi, K.: Computation of recommender system using localized regularization. Procedia Comput. Sci. **51**, 2407–2416 (2015)
62. Modarresi, K.: Algorithmic approach for learning a comprehensive view of online users. Procedia Comput. Sci. **80**, 2181–2189 (2016)
63. Sedhain, S., Menon, A.K., Sanner, S., Xie, L.: Autorec: autoencoders meet collaborative filtering (2015)
64. Stekhoven, D.: Using the missForest Package. CRAN (2012)
65. Strub, F., Mary, J., Gaudel, R.: Hybrid collaborative filtering with autoencoders (2016)
66. Van Buuren, S., Groothuis-Oudshoorn, K.: MICE: multivariate imputation by chained equations in R. J. Stat. Softw. **45**(3), 1–67 (2011)