Using Apple Store Dataset to Predict User Rating of Mobile Applications

Kevin Daimi[1] and Noha Hazzazi[2]
[1]Department of Electrical and Computer Engineering, and Computer Science, University of Detroit Mercy, Detroit, USA
[2]Department of Electrical Engineering and Computer Science, Howard University, Washington, USA
daimikj@udmercy.edu, noha.hazzazi@howard.edu

***Abstract*—The use of mobile phones is constantly growing. These phones are being considered exceedingly inescapable and vital. Consumers worldwide are devotedly appreciating their potential benefits. In parallel with this growth, mobile phone apps are booming with new applications and new versions produced on almost daily basis. This growth is making it hard for users to decide which app or version to buy and even harder for developers to ensure their apps will be marketable. In an attempt to assist users in making the right choice, this paper proposes utilizing Data Science to predict user rating of various mobile apps. To this end, Linear Regression, Neural Networks, Support Vector Machines, Random Forest, M5 Rules, REP Tree, and Random Tree algorithms will be employed using Weka and their output will be compared. All techniques will be applied to the Apple Store Dataset.**

Keywords—Mobile Apps Rating, Prediction, Neural Networks, Linear Regression, Support Vector Machines, Random Forest, Random Tree, M5 Rules, REP Tree, Weka

I. Introduction

Mobile phones are enormously used both nationwide and worldwide. People perform many tasks using these phones including email, online payments, online purchasing, and appointments. Many new applications are being created and new versions for currently used application are being produced. Many users are not technically-oriented, and therefore, need help with selecting the right app or right version of the app. Furthermore, apps developers are also interested in apps ratings to ensure that their developed apps will be marketable. Developers of mobile applications are inspired by high app ratings and excellent user reviews. The evaluation of mobile apps for iOS devices in the U.S. market were summarized in [1]. They gathered data from a total of 968 mobile applications and 48,374,030 user reviews and uncovered statistically substantial differences in reviews of different classes of mobile applications and stressed the positive impact of factors of app purchase for various evaluations. Their analysis revealed that user applications reviews were positively appraised.

Liang, Chen, Ying, Yu, Wu, and Zheng [2] indicated that predicting user preferences on apps turn into an inspiring issue. They assumed that users like an app because they prefer certain aspects of the app. Based on this assumption, they proposed a feature-oriented technique to predict user preferences on apps. The authors converted the original app rating matrix to feature rating data and predicted the unknown ratings on the features using a Latent Factor model rather than directly predicting ratings on apps. The predicted user ratings on features are then used to produce the ratings for apps. According to the authors, their approach integrated feature information to explore the details of user preference and can improve the understanding of the app ratings prediction.

An empirical evidence confirming the rating an app receives can be accurately predicted through the features it offers was emphasized in [3]. They based their findings on analyzing 11,537 apps of Samsung Android and BlackBerry worldwide app stores. These findings showed that the rating of 89% of these apps were predicted with 100% accuracy. Their prediction model was developed by combining feature and rating information of existing apps in app stores yielding highly accurate rating predictions with only 11-12 existing apps for case-based prediction. They further stressed that their outcomes offer new opportunities to aid the requirements elicitation process for app developers. Another app prediction approach targeting developers was introduced in [4]. They implied that the immense rate of mobile app market growth during the forgoing few years has pushed many developers to develop mobile apps. Despite this growth, some developers have been very successful with their apps while the majority were struggling. They emphasized that if the struggling developers can be familiar with high-rated apps, they will be shifting towards successful development and progress of their apps. To achieve their above statements, they investigated 28 elements along eight dimensions to realize how high-rated apps are different from low-rated apps and concentrated on the most dominant elements through applying a Random Forest classifier to discover high-rated apps. To this end, they performed a case study on 1,492 high-rated and low-rated free apps extracted from the Google Play store and concluded high-rated apps are statistically significantly different in 17 out of the 28 elements that they considered.

Yao, Zhao, Wang, Tong, Xu, and Lu [5] dealt with the technical challenge of recommending right apps to mobile users. They focused on a unique characteristic of mobile app relating to multiple releases and versions. Based on this characteristic, they suggested a version-aware app recommendation solution that is based on inferring the ratings of users on a specific version of an app rather than precisely acquiring the users' preferences over the apps. Their approach proposed weighting the review associated with each rating. They took into account two kinds of version-based correlations; acquiring the temporal correlations between multiple versions within the same app, and the aggregation correlations between similar apps. Baeza-Yates, Jiang, Silvestri, and Harrison [6] investigated

how to enhance home-screen apps' usage capability with a prediction mechanism that allows making decision on what apps to use in the immediate future. The prediction approach used is based on a set of features signifying the real-time spatiotemporal contexts perceived by the home-screen app. The prediction of a new app was treated as a classification problem and suggested a personalized method to solve this problem investing the full advantage of human-engineered features and further automatically derived features. They carried out large-scale experiments on log data obtained from Yahoo Aviate and concluded their approach is capable of accurately predicting the next app that attracts users.

N. Natarajan, D. Shin, and I. S. Dhillon [7] stressed that the next smart phone app a user will adopt is subliminally influenced by the sequence of apps used recently. The authors added, when users work with online systems, such as shopping websites and online radio, they click on items of interest in the current context (referred to as interactional context). They further indicated it was necessary for a recommender system to employ the context set by the user to update recommendations. Most existing context-aware recommender systems focus on a relatively less dynamic representational context signified by some attributes including season, location and taste. To this extent, they proposed the iConRank technique that has two steps. In the first step, users are first clustered by their change behavior and cluster-level Markov models are deployed, and during the second step, a personalized Page-Rank is calculated for a given user on the corresponding cluster Markov graph with a personalization vector derived from the current context. Fu, Lin, Liy, Faloutsos, Hong, and Sadeh [8] proposed a system, WisCom, to analyze a large number of user ratings and comments in mobile app markets at three different levels: detecting inconsistencies in reviews, identifying reasons behind preferences for a given app and offering an interactive view of users' reviews progressing, and presenting a valuable perceptions into the entire app market. A dataset consisting of over 13 million user reviews of 171,493 Android apps in the Google Play Store was used. The goal was to help mobile app market operator, app developers, and end-users.

With the remarkable growth of mobile application markets, users can purchase various apps with any needed functionality in these markets. However, the large number of apps makes it very hard for users to find out good and useful apps. To avoid regarding all apps equal without portraying the specific interests of each user, a Weight-based Matrix Factorization (WMF) model to catch user-specific interests and provide a more accurate prediction on these apps was proposed [9]. WMF considered each user as a document and each app as a word and computed the weight of each app for users in question. The weights were calculated by utilizing Term Frequency Inverse Document Frequency (TF-IDF) algorithm and were then fed into matrix factorization to predict app ratings. A real-world dataset with 5057 users and 4496 apps were relied on during this study to draw the conclusions. Yu, Li, Xu, Zhang, and Kostakos [10] presented population-level, city-scale analysis of application usage on smartphones. Using deep packet inspection at the network operator level, they acquired a geo-tagged dataset with over 6 million unique devices that used over 10,000 unique apps across the city of Shanghai within one week. They then introduced a technique that relied on transfer learning to predict apps that are most popular and estimated the whole usage distribution based on the Point of Interest (POI) information of that particular location. Further work in this regard could be found in [11]-[13].

In this paper, user rating of mobile phone apps will be predicted. Predictive analytics is the field of statistics that is concerned with extracting information from datasets to predict possible trends and behavior. Seven techniques will be used and compared: Linear Regression, Neural Networks, Support Vector Machines, Random Forest, M5 Rules, REP Tree, and Random Tree. All algorithms will be implemented using Weka [14]. Section II presents the dataset description. Section III deals with prediction using Weka for the seven methods. The paper is then concluded in Section V.

II. Dataset description

The Apple Store dataset [15] used in this paper contains customer evaluations for various applications. There are 7197 instances (rows) and 16 attributes (columns). Ten instances (unseen data) out of the 7197 instances are removed to be used for implementation. This leaves 7187 instances in the dataset. Sample rows of this dataset are provided in Table I. In this Table, the rows are written as columns due to space limitation. The dataset was cleaned by removing the columns for the following attributes: id, track_name, size_bytes, currency, price, ver, cont_rating, prime_genre, vpp_lic. The remaing attributes are as follows:

- rating_count_tot: User Rating counts for all version (integer).
- rating_count_ver: User rating counts for current version (integer).
- user_rating: Average user rating value for all version (double). This attribute will be predicted.
- user_rating_ver: User rating counts for current version (double).
- sup_devices.num: Number of supporting devices (integer).
- ipadSc_urls.num: Number of screen shots showed for display (integer).
- lang.num: Number of supported languages (integer).

III. Prediction with weka

Weka is used to predict the values of user rating (user_rating) of an app. 66% of the dataset is used for training and the rest (34%) for testing. Note that the values of user rating for the unseen instances are given below. These should

be interpreted as user rating is 3.5 for the first unseen instance, 4 is for the second unseen instance, and so on.

{3.5 4 4.5 4 3.5 3 4 4.5 3 3.5}

A. Using Linear Regression

Linear regression deals with finding the best-fitting straight line through the given points. The best-fitting line is referred to as regression line [16]. After applying the Linear Regression method to the Apple Store dataset, the model was used to predict user rating for 10 instances. Table II provides sample results for the prediction during the testing step, Table II illustrates the associated statistics, and Table III depict the prediction of user_rating using unseen instances.

Note that *Correlation Coefficient* is a number between −1 and +1 representing the linear dependence between sets of data. This relationship is considered strong when this coefficient is larger than 0.7, the *Mean Absolute Error* measures the average of the absolute errors in a set of predictions, *Root Mean Squared Error* is the square root of the average of squared differences between predictions and actual observations, *Relative Absolute Error* is the sum of the absolute differences between the predictions and actual observations divided by the sum of the absolute differences between the average of the observation and the observations, and *Root Relative Squared Error* is the square root of the sum of the squared differences between the predictions and actual observations divided by the sum of the squared differences between the average of the observations and the observations.

TABLE II. ASSOCIATED STATISTICS

Correlation coefficient	0.7719
Mean absolute error	0.6350
Root mean squared error	0.9576
Relative absolute error	56.311%
Root relative squared error	63.5983%

Only 7187 instances are used because 10 instances were saved for the unseen samples (dataset).

TABLE III. USER RATING PREDICTION - UNSEEN INSTANCES

Unseen Instance	Predicted User Rating
1	4.069656
2	3.573414
3	2.191008
4	4.138177
5	3.951197
6	3.292516
7	3.995180
8	4.914003
9	4.014383
10	4.523481

B. Using Nueral Networks

In addition to applying Linear Regression, Neural Networks techniques was deployed. Neural Networks are able to learn from examples and once their learning is completed, they will be able to capture hidden and potently nonlinear dependencies even with the existence of major noise in the training set [17]. The equivalent tables to Tables II and III are given below.

TABLE IV. ASSOCIATED STATISTICS

Correlation coefficient	0.7724
Mean absolute error	0.5986
Root mean squared error	0.9747
Relative absolute error	53.083%
Root relative squared error	64.7353%

TABLE V. USER RATING PREDICTION - UNSEEN INSTANCES

Unseen Instance	Predicted User Rating
1	4.658
2	4.660
3	3.022
4	4.284
5	4.378
6	3.931
7	4.616
8	4.718
9	1.277
10	3.500

C. Applying Support Vector Machines

The purpose of the Support Vector Machine (SVM) algorithm is to locate a hyperplane in an N-dimensional space, where N is the number of features, that plainly predicts the data points. There are many possible hyperplanes that could be used. The goal is to find a plane with the maximum distance between data points. Maximizing this distance provides support to conclude that future data points can be predicted with more confidence [18]. Table VI presents the statistics associated with SVM algorithm, and table VII shows the results on unseen data.

TABLE VI. ASSOCIATED STATISTICS

Correlation coefficient	0.7717
Mean absolute error	0.5590
Root mean squared error	1.1851
Relative absolute error	49.5663%
Root relative squared error	78.7085%

TABLE VII. USER RATING PREDICTION - UNSEEN INSTANCES

Unseen Instance	Predicted User Rating
1	4.500
2	4.500
3	0.006
4	3.999
5	4.000
6	3.499
7	4.997
8	4.718
9	0.005
10	2.501

D. Applying Random Forest

In this section, Random Forest will be applied. Table VIII and Table IX depict the statistics and the output of predicting unseen data. The Random Forest algorithm is an adaptable, straightforward to use machine learning algorithm that generates, even without hyper-parameter tuning, excellent results most of the time. It is one of the most used algorithms due to its simplicity and adoptability for both classification and prediction tasks [19].

TABLE VIII. ASSOCIATED STATISTICS

Correlation coefficient	0.9493
Mean absolute error	0.2966
Root mean squared error	0.4736
Relative absolute error	26.305%
Root relative squared error	31.4535%

TABLE IX. USER RATING PREDICTION - UNSEEN INSTANCES

Unseen Instance	Predicted User Rating
1	3.815
2	4.155
3	4.317
4	4.020
5	3.625
6	3.335
7	4.160
8	4.505
9	3.110
10	3.490

E. Applying M5 Rules

The M5 Rules algorithm is a single conjunctive rule learner and decision table majority predictor. It creates a decision list for prediction problems using separate-and-conquer. In each iteration it builds a model tree and crafts the "best" leaf into a rule [20]. Tables X and Table XI introduce the associated statistics and results of the algorithm on unseen data.

TABLE X. ASSOCIATED STATISTICS

Correlation coefficient	0.7493
Mean absolute error	0.3230
Root mean squared error	1.2087
Relative absolute error	28.6394%
Root relative squared error	80.2734%

TABLE XI. USER RATING PREDICTION - UNSEEN INSTANCES

Unseen Instance	Predicted User Rating
1	4.418
2	4.414
3	3.894
4	4.126
5	4.097
6	3.909
7	4.413
8	4.461
9	3.723
10	2.962

F. Applying REP Tree

The sixth algorithm used in this study is the REP Tree algorithm. It is a fast decision tree learner based on C4.5 algorithm that can be used for both classification and prediction. The statistics produced by the algorithm are demonstrated in Table XII.

TABLE XII. ASSOCIATED STATISTICS

Correlation coefficient	0.9492
Mean absolute error	0.3002
Root mean squared error	0.4742
Relative absolute error	26.6216%
Root relative squared error	31.491%

The predictions for unseen instances are displayed in Table XIII.

TABLE XIII. USER RATING PREDICTION - UNSEEN INSTANCES

Unseen Instance	Predicted User Rating
1	4.418
2	4.418
3	3.826
4	4.122
5	4.095
6	3.125
7	4.311
8	4.333
9	3.826
10	3.769

G. Applying Random Tree

The Random Tree algorithm functions just like the Decision Tree algorithm, however, for each split, it only picks out a random subset of attributes prior to applying them [21]. The needed tables for this study are depicted below.

TABLE XIV. ASSOCIATED STATISTICS

Correlation coefficient	0.9038
Mean absolute error	0.3911
Root mean squared error	0.6611
Relative absolute error	34.6777%
Root relative squared error	43.9042%

TABLE XV. USER RATING PREDICTION - UNSEEN INSTANCES

Unseen Instance	Predicted User Rating
1	3.5
2	4
3	4.5
4	4
5	3.5
6	3
7	4
8	4.5
9	3
10	3.5

IV. DISCUSSION AND CONCLUSION

To reveal various predictions, unfamiliar correlations, and other valuable knowledge to make well-informed decisions, this paper contributed by investigating the possible user rating for the Apple Store dataset. Knowing how users rate various applications and various versions of the same application is beneficial to both users and developers. Users can decide what application or version to purchase, and developers will invest time and money on on-demand apps only.

Seven methods were used to extract the user rating on apps and versions. By analyzing the tables for Associated Statistics, it is easily concluded that Random Forrest techniques scored the highest Correlation Coefficient value of 0.9493, which is closest to 1, while M5 Rules produced the lowest one. The lowest Mean Absolute Error (0.2966) was generated by Random Forest and the highest (0.6350) by Linear Regression approach. For the Root Mean error, Relative Absolute Error, and Root Relative Square Error, Random Forest scored the minimum of all (0.4736, 26.305%, and 31.4535% respectively). M5 Rules got the highest Root Mean Error of 1.2087, and the highest Root Relative Square Error of 80.2734%. On the other hand, Linear Regression recorded highest values for two errors: Mean Absolute Error (0.6350) and Relative Absolute Error (56.311%). In general, with respect to this dataset, the performance of both Linear Regression and M5 Rules was poorer than the other five algorithms.

With regards to the predictions of unseen instances, Random Tree algorithm produced exactly the same predictions as the actual unseen user ratings mentioned at the beginning of Section III. Random Forrest followed. In general, the Tree algorithms were superior to others. Support Vector Machines failed to predict a value near the actual unseen value 4.5 twice. The two values that were produced for 4.5 by this algorithm were 0.006 and 0.005.

Normally, the algorithm that is preferable is the one that has the highest Correlation Coefficient and the lowest error values for all the error types. Hence, the best algorithm for predicting the user rating for the Apple Store dataset is Random Forrest.

REFERENCES

[1] D. Erić, R. Bačík, and I. Fedorko, "Rating decision analysis based on iOS App store data, quality, innovation, prosperity, vol. 18, no. 2, pp. 27-37, 2014.

[2] T. Liang, L. Chen, X. Ying, P.S. Yu, J. Wu, and Z. Zheng, "Mobile Application Rating Prediction via Feature-Oriented Matrix Factorization," in Proc the 2017 IEEE International Conference on Web Services (ICWS), Honolulu, HI, USA, 2017, pp. 261-268..

[3] F. Sarro, M. Harman, Y. Jia, and Y. Zhang, "Customer Rating Reactions Can Be Predicted Purely using App Features," in Proc. the IEEE 26th International Requirements Engineering Conference (RE), Banff, AB, Canada, 2018, pp. 76-87.

[4] Y. Tian, M. Nagappan, D. Lo, A. E. Hassan, "What are the characteristics of high-rated apps? A case study on free Android Applications," in Proc. the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2015, Bremen, Germany, pp. 301-310.

[5] Y. Yao, W. X. Zhao, Y. Wang, H. Tong, F. Xu, and J. Lu, "Version-aware rating prediction for Mobile App Recommendation," ACM Transcation on Information Systems (TOIS) – Special issue: Search, Mining and their Applications on Mobile Devices, vol. 35, no. 4, Article Number 38, 2017.

[6] R. Baeza-Yates, D. Jiang, F. Silvestri, and B. Harrison, "Predicting the next app that you are going to use," In Proc. the Eighth ACM International Conference on Web Search and Data Mining (WSDM), Shanghai, China, 2015, pp. 285-294.

[7] N. Natarajan, D. Shin, and I. S. Dhillon, "Which App Will You Use Next? Collaborative Filtering with Interactional Context," in Proc. the 7th ACM Conference on Recommender Systems, Hong Kong, China, 2013, pp. 201-208.

[8] B. Fu, J. Lin, L. Liy, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your App — Making sense of user feedback in a mobile App store," in Proc. the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13), Chicago, Illinois, USA, 2013, pp. 1276-1284.

[9] J. Meng, Z. Zheng, G. Tao, and X. Liu, "User-specific rating prediction for mobile applications via weight-based matrix factorization," in Proc. the 2016 IEEE International Conference on Web Services, San Francisco, CA, USA, 2016, pp. 728-731.

[10] D. Yu, Y. Li, F. Xu, P. Zhang, and V. Kostakos, "Smartphone App usage prediction using points of interest," in Proc. the ACM Interactive, Mobile, Wearable, and Ubiquitous Technologies, vol. 1, no. 4, pp. 1-21, 2017.

[11] A. Holzer and J. Ondrus, "Mobile application market: A developer's perspective," Telematics and Informatics, vol. 28, no. 1, pp. 22–31, 2011.

[12] S. Lim, P. Bentley, N. Kanakam, F. Ishikawa, and S. Honiden, "Investigating country differences in mobile app user behavior and challenges for software engineering," IEEE Transactions on Software Engineering, vol. 41, no. 1, pp. 40–64, Jan 2015.

[13] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," IEEE Transactions on Software Engineering, vol. 43, no. 9, pp. 817–847, Sept 2017.

[14] E. Frank, M. A. Hall, and I. H. Witten, "The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

[15] Apple Store Dataset, Kagle, [Online]. Available: https://www.kaggle.com/ramamet4/app-store-apple-data-set-10k-apps. Accessed: May 12, 2019.

[16] D. M. Lane, "Introduction to Linear Regression," [Online]. Available: http://onlinestatbook.com/2/regression/intro.html. Accessed: May 12, 2019.

[17] Prediction using neural networks, [Online]. Available: https://www.obitko.com/tutorials/neural-network-prediction/introduction.html. Accessed: May 12, 2019.

[18] R. Gandhi, "Support Vector Machine—Introduction to machine learning algorithms: SVM model from scratch," June 2018, [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. Accessed: May 12, 2019.

[19] N. Donges, "The Random Forest Algorithm," February 2018, [Online]. Available: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd. Accessed: May 12, 2019.

[20] Class M5 Rules, [Online]. Available: http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/M5Rules.html. Accessed: May 12, 2019.

[21] RapidMiner GmbH, "Random Trees," 2019, [Online]. Available: https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/trees/random_tree.html Accessed: May 12, 2019.