ML1819 Research Assignment 2

a) **Team ID:** *19*

b) **Task ID and Title:** *102. Dataset Pruning: What is the effect on Machine Learning Performance?*

c) **Student Names and IDs:**

- *Aneek Barman Roy - 18304921*
- *Debrup Chakraborty - 18304460*
- *Viren Chhabria - 18301780*

d) **Contribution:**

Name	Student ID	GitHub username	Contribution	Percentage
Aneek Barman Roy	18304921	iamaneek	• Generated visualizations using python • Implementation of Ridge Regression • Literature Review • Report content writing	33.33%
Debrup Chakraborty	18304460	rupdeb	• Generating visualizations from Excel • Implementation of Support Vector Regressor • Parameter tuning for the algorithms • Literature Review • Report content writing	33.33%
Viren Chhabria	18301780	chhabriv	• Data Pre-Processing • Implementation of Random Forest Regressor • Comparison of evaluation metrics • Literature Review • Report content writing	33.33%

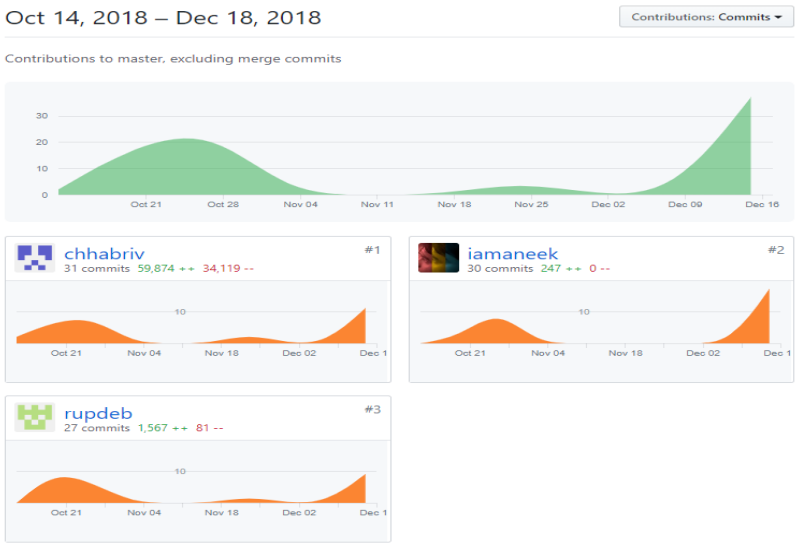
e) **Word Count:** 1414

f) **Exceptional circumstances:** N/A

g) **Repository link:** https://github.com/chhabriv/ML1819--task-102--team-19

h) **Repository contributor's link:** https://github.com/chhabriv/ML1819--task-102--team-19/graphs/contributors

i) **Commit Activity:**



Impact of Data Pruning on Machine Learning Algorithm Performance

Debrup Chakraborty
School of Computer Science & Statistics
Trinity College Dublin
Dublin, Ireland
chakrabd@tcd.ie

Viren Chhabria
School of Computer Science & Statistics
Trinity College Dublin
Dublin, Ireland
chhabriv@tcd.ie

Aneek Barman Roy
School of Computer Science & Statistics
Trinity College Dublin
Dublin, Ireland
barmanra@tcd.ie

Abstract: Dataset pruning is the process of removing sub-optimal tuples from a dataset to improve the learning of a machine learning model. In this paper, we compared the performance of different algorithms, first on an unpruned dataset and then on an iteratively pruned dataset. The goal was to understand whether an algorithm (say A) on an unpruned dataset performs better than another algorithm (say B), will algorithm B perform better on the pruned data or vice-versa. The dataset chosen for our analysis consisted of the user ratings of 10,841 applications from the Google Play application store. The ratings were continuous, in the range of 1-5. The dataset was pruned iteratively based on the number of reviews and three regression algorithms to predict the application ratings were implemented. The results indicated that an algorithm that performed better on an unpruned dataset also performed better on a pruned dataset.

Keywords: rating prediction, google play store apps, data pruning

1 INTRODUCTION

A fine line separates cleaning and pruning of a dataset. Cleaning mostly is a preprocessing step that involves removing unrequired data, data imputation, standardizing or normalizing the feature ranges and converting categorical values to numbers [1, 2]. In comparison pruning takes place after preprocessing, where certain data is strategically removed to improve the machine learning model. In this paper we try to bring forth the effect of dataset pruning on the performance of different machine learning algorithms, i.e. If an algorithm (say A) on an unpruned dataset performs better than another algorithm (say B), will algorithm B perform better on the pruned data or vice-versa.

2 RELATED WORK

Data pruning had been defined in 2005 as an automated process of noise cleaning and the performance of this mechanism was measured using SVC and AdaBoost algorithms [3]. Removal of certain portions of the dataset is determined to be worthwhile and said to affect the performance of machine learning algorithms [3].

Rating systems for mobile apps were examined to study whether the aggregated ratings capture changing user satisfaction levels [4].

A users interest in a specific app was predicted using preferences of the users on features of the app, rather than the app as a whole [5].

Sentimental analysis was used to predict star rating of an application based on the annotated review given by users [6].

3 METHODOLOGY

3.1 Dataset

The dataset used was chosen from Kaggle, and it consists of data regarding applications (app) available on one of the largest app stores in the world, Google Play [4, 7, 8]. It contains details about 10,481 apps with 13 attributes, where "Rating" indicates the app ratings on a scale of 1-5. The ratings are a cumulative average of individual user ratings of the app over all the versions [4]. The year wherein the app was last updated is shown in Figure 14. The histogram in Figure 1 shows the frequency distribution of the app ratings, depicting the rating between 4.2 – 4.6 to be the highest. The different features in the dataset along with their datatype are shown in Table 1.

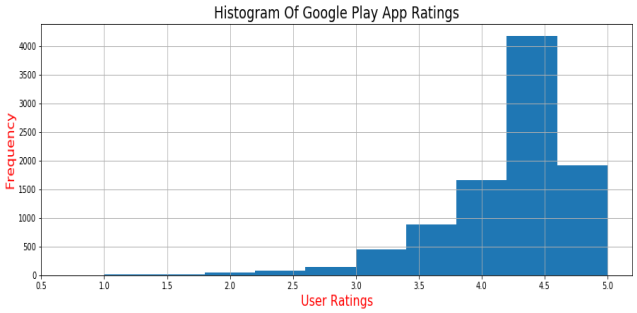


Figure 1: Frequency of app ratings on raw dataset

Table 1: Description of features in the dataset

Attribute	Description	Type
App	Name of the app	Character
Category	The category to which the app belongs	Categorical
Rating	Mean ratings given to the app	Numerical
Reviews	Number of reviews	Numerical
Size	Size of the app	Numerical
Installs	Number of installations	Numerical
Type	Type of pricing	Categorical
Price	Price value of the app	Numerical
Content Rating	Rating of the app content	Categorical
Genres	Genre to which the app belong	Categorical
Last Updated	Last date when the app was updated	Date
Current Ver	Current version of the app	Categorical
Android Ver	Minimum android version requirement(s)	Categorical

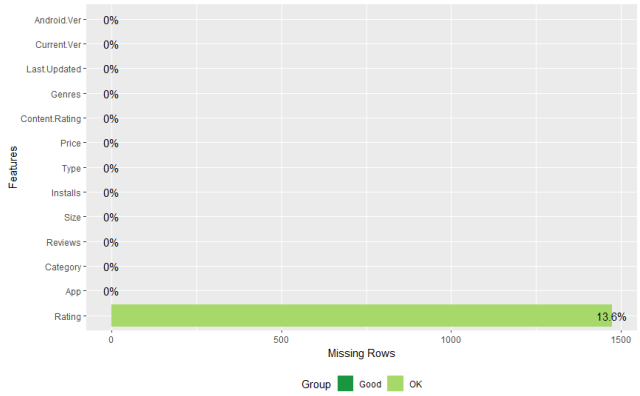


Figure 2: Missing data - feature wise

Table 2: Missing data values

Feature	% Missing
App	0
Category	0.01
Rating	13.6
Reviews	0
Size	0
Installs	0
Type	0.01
Price	0
Content Rating	0
Genres	0.01
Last	0
Current Ver	0.07
Android Ver	0.02

3.2 Pre-processing

3.2.1 Missing Data

Google Play app ratings have continuous values in the range 1-5. The percentage of missing data per column in the dataset can be seen in Figure 2 and Table 2. The 'Ratings' column, which is the target variable had 13.6% of the data missing. 5 other features also had missing data, but the number was less than 1%. Consequently, all the rows containing missing data were removed, since key data was missing, and it was not reasonable to impute the target variable and then use it for building the model.

3.2.2 Inconsistent Data

The feature size had data in 2 measures: Kilobytes(KB) and Megabytes (MB). This feature was made consistent by converting all the size values to KB.

3.2.3 Outliers

Scatter plot shown in Figure 3 was used to detect outliers in the dataset with respect to the feature reviews. The outliers were dealt with by standardizing the features. Figure 4 shows the scatter plot created using the standardized feature.

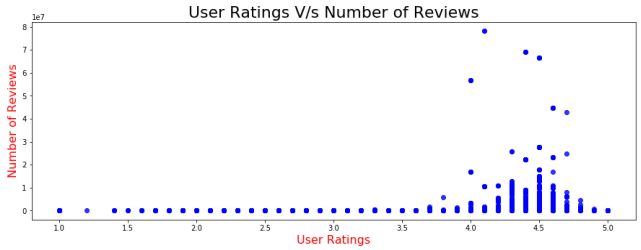


Figure 3: User Ratings vs Number of Reviews

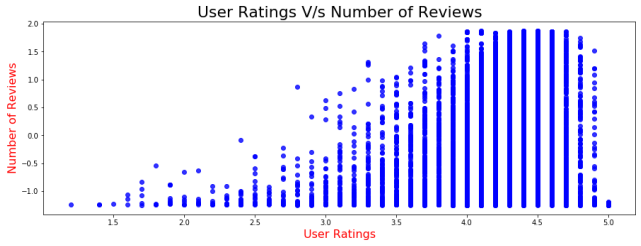


Figure 4: User Ratings vs Standardized Number of Reviews

3.2.4 Feature Transformation

String data[1] in the numeric columns were replaced with -1. Duplicate tuples were removed. Categorical features were transformed to binary representation using LabelEncoder and OneHotEncoder (utilities in scikit-learn framework) [2].

3.2.5 Feature Scaling

The feature with numeric data were standardized using StandardScaler (utility in scikit-learn framework) [9].

3.2.6 Feature Selection / Removal

Last updated date and current version of the app were removed from the dataset since they were irrelevant to the model. Heatmap shown in Figure 5 represents the correlation between the remaining 11 features in the dataset. The independent variable 'Genre' was dropped from the dataset since it was highly correlated with the variable 'Category'.

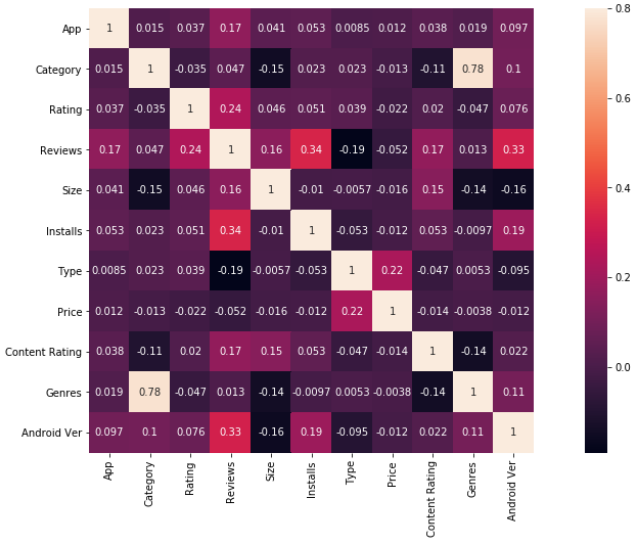


Figure 5: Correlation between the selected features

Random Forest was used to find the most important features in the dataset. The variable importance plot can be seen in Figure 6. The variables with less than 100 node purity (higher impurity), were not used for training the model.

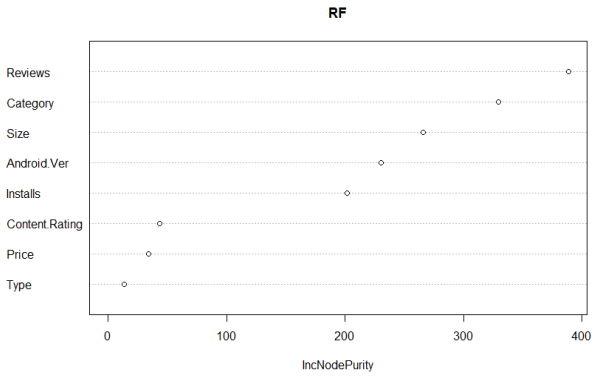


Figure 6: Random Forest variable importance plot

3.3 Data Pruning

The pre-processed dataset was then pruned based on the number of user reviews for an app. The dataset was iteratively pruned where an app had received less than [1..20] user reviews. This was done as a lower review count would make the rating of that app biased to a small (<20) number of user opinions [4]. Figure 3 shows a scatter plot depicting the number of user reviews v/s app rating.

3.4 Algorithms

Linear Regression, Random Forest Regression and Support Vector Regression were used to evaluate the sensitivity of

[1] In some of the features the numeric data had values "Varies with device"

machine learning algorithms to data pruning. All the algorithms were implemented using the scikit-learn framework [10-12]. Hyper parameter tuning for these algorithms was done on an unpruned dataset using k-fold cross validation (utility in scikit-learn framework), and the best parameter measure from the findings were used for each iteration of the pruned dataset [13]. The value of k used was for all the cases discussed in this paper was 10.

3.4.1 *Linear Regression*

Ridge regression was used with different values of the regularization parameter as shown in Figure 7, however, the performance across 10-fold cross validation remained the same, and hence a simple linear regression was used without any hyper parameter tuning.

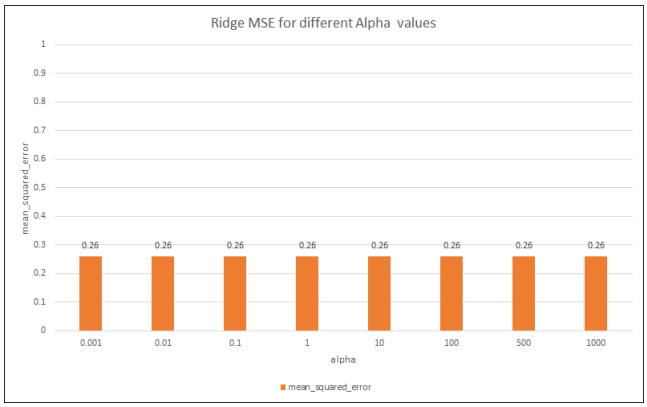


Figure 7: MSE with respect to alpha for Ridge Regression

3.4.2 *Random Forest Regression:*

The n-estimator (number of decision tree classifiers) for random forest was chosen from the range 10 to 100 in increments of 10, using 10-fold cross validation and it was found to be consistent 30 onwards as shown in Figure 8.

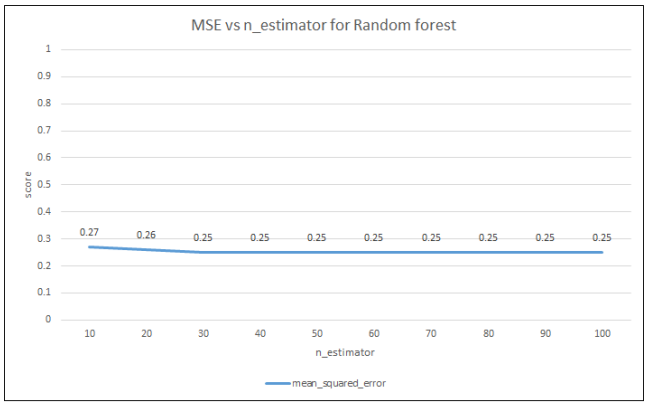


Figure 8: MSE with respect to n-estimators for Random Forest Regression

3.4.3 *Support Vector Machine (SVM):*

SVM weights were tuned to prevent overfitting on larger margins. For c (regularization parameter) in the range [0.001, 0.01, 0.1, 10, 25, 50, 100] using k-fold cross validation, SVM was found to perform best at 1 as shown in Figure 9.

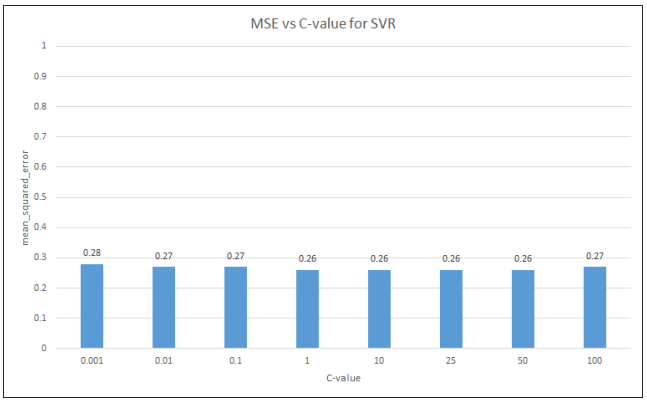


Figure 9: MSE with respect to c-value for SVR

3.5 Evaluation

10% of the dataset was kept aside before training the model. This was done to mimic this data as the future/out-of-sample data to test the performance of the model. 90% of the dataset was used to create the model and find mean validation metrics using 10-fold cross validation. The RMSE and R2 metrics for cross validation vs percentage of data pruned for each algorithm can be seen in Figure 10 and Figure 11.

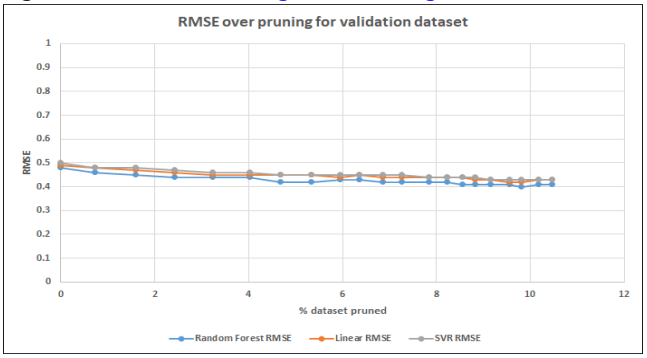


Figure 10: RMSE vs percentage of data pruned for each algorithm using 10-fold cross validation

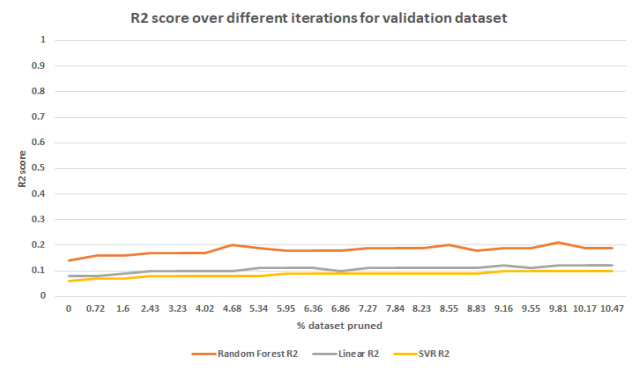


Figure 11: R2 vs percentage of data pruned for each algorithm using 10-fold cross validation

3.5.1 Metrics

Root Mean Square Error (RMSE) and Coefficient of Determination (R2 score) was used to evaluate the models. The metrics were calculated using the scikit-learn framework [14].

Rationale for choosing the metrics:

- RMSE: Explains the difference between the predicted value and actual value of the 'Rating' [5].
- R2 score: Explains the variability in the target variable 'Rating' with respect to significant variables in the model.

4 RESULTS AND DISCUSSION

4.1 Regression Metrics

The mean and standard deviation of the RMSE of the three algorithms for out-of-sample data has been shown in Table 3.

Table 3: Mean and standard deviations of RMSE

	Random Forest	Linear Regression	SVR
Mean	0.43	0.45	0.45
Standard Deviation	0.02	0.02	0.02

Result for each iteration: Figure 12, Table 5

The mean and standard deviation of R2 scores for the three algorithms for out-of-sample data have been mentioned in Table 4.

Table 4: Mean and deviations of R2 scores

	Random Forest	Linear Regression	SVR
Mean	0.19	0.11	0.09
Standard Deviation	0.03	0.01	0.01

Result for each iteration: Figure 13, Table 6

4.2 Discussion

Starting with an unpruned dataset, 20 iterations were run to prune the dataset to check how the three algorithms performed with each iteration. For the unpruned dataset (0% pruned), Random Forest Regression had the lowest RMSE and the highest R2 as shown in Figure 12 and Figure 13. The RMSE and R2 scores fluctuated as per Table 5 , Table 6 and improved with each iteration, but the ranking of the algorithms with respect to each other remained unchanged. The low values of the R2 metrics can be attributed to a small (5 features) number of features of the dataset being used to train the model as per the variable importance plot (Figure 6).

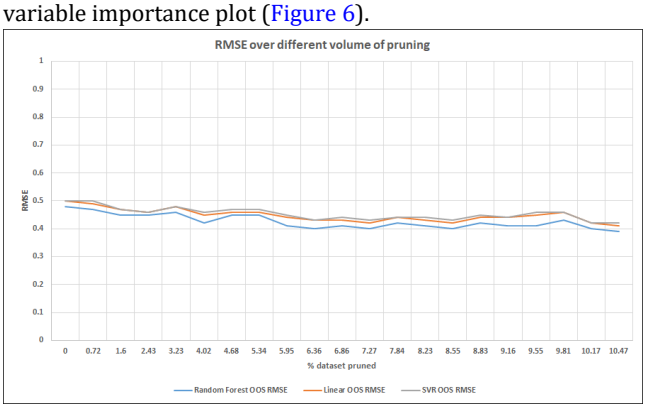


Figure 12: RMSE score of each algorithm per iteration

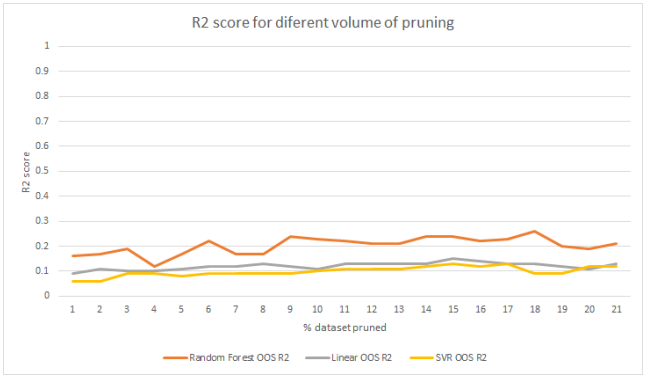


Figure 13: R2 score of each algorithm per iteration

4.3 Results

Related works on Google Play store apps dataset were mostly centered on sentimental analysis to predict app ratings and examining the effectiveness of the current app rating system [4, 6]. We ran an unbiased analysis on the three algorithms and observed that the performance an algorithm improved with respect to itself with each iteration of pruning, but did not perform better or worse than another algorithm as the dataset was pruned as shown in Figure 12 and Figure 13. For about 10.5% pruning of the dataset, RMSE improved by 7% for Random Forest, 8% for Linear Regression and 8% for SVR. R2 scores improved by 5% for Random Forest, 3% for Linear

Regression and 5% for SVR. Their rankings remain unchanged on unpruned and pruned datasets across the two metrics used, though several iterations showed fluctuations in their performance.

To conclude, pruning of datasets didn't affect the algorithm performance rankings.

5 LIMITATION AND OUTLOOK

The dataset had 13.6% of the target variable missing data. Removal of these values meant that the related useful information on the independent variable was lost. Low (5 features) number of important features chosen to train the model, explained less variability in the predicted values. Future work could include using another dataset with more features, pruning multiple features instead of one as done in this paper and using neural networks for comprehensive analysis.

6 ACKNOWLEDGMENT

This work was conducted as part of 2018/19 Machine Learning module CS7CS4/CS4404 at Trinity College Dublin [15].

7 REFERENCES

[1] (2018). *sklearn.preprocessing.StandardScaler â" scikit-learn 0.20.1 documentation* [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[2] (2017). *How to One Hot Encode Sequence Data in Python* [Online]. Available: https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/.

[3] A. Angelova, Y. Abu-Mostafam, and P. Perona, "Pruning training sets for learning of object categories," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, vol. 1, pp. 494-501 vol. 1.

[4] I. J. M. Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. E. Hassan, "Examining the Rating System Used in Mobile-App Stores," *IEEE Software,* vol. 33, no. 6, pp. 86-92, 2016.

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

[6] D. Monett and H. Stolte, "Predicting star ratings based on annotated reviews of mobile apps," in *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2016, pp. 421-428.

[7] (2018). *Google Play Store Apps Dataset* [Online]. Available: https://www.kaggle.com/lava18/google-play-store-apps.

[8] (2018). *Google Play* [Online]. Available: https://play.google.com/store?hl=en.

[9] (2018). *sklearn.preprocessing.StandardScaler â" scikit-learn 0.20.1 documentation* [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[10] (2018). *3.2.4.3.2. sklearn.ensemble.RandomForestRegressor* [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

[11] (2018). *sklearn.linear_model.LinearRegression* [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.

[12] (2018). *sklearn.svm.SVR* [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html.

[13] (2018). *3.1. Cross-validation* [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html.

[14] (2018). *3.3. Model evaluation: quantifying the quality of predictions â" scikit-learn 0.20.1 documentation* [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics.

[15] Joeran Beel and Douglas Leith. Machine Learning (CS7CS4/CS4404). Trinity College Dublin, School of Computer Science and Statistics. 2018.

Appendix

Table 5: RMSE per iteration for each algorithm

Pruned (%)	Random Forest	Linear Regression	SVR
0	0.48	0.50	0.50
0.72	0.47	0.49	0.50
1.6	0.45	0.47	0.47
2.43	0.45	0.46	0.47
3.23	0.46	0.48	0.48
4.02	0.42	0.45	0.46
4.68	0.45	0.47	0.47
5.34	0.44	0.46	0.47
5.95	0.42	0.45	0.45
6.36	0.40	0.43	0.44
6.86	0.42	0.43	0.44
7.27	0.41	0.43	0.43
7.84	0.42	0.44	0.45
8.23	0.41	0.44	0.44
8.55	0.41	0.43	0.43
8.83	0.43	0.44	0.45
9.16	0.41	0.44	0.44
9.55	0.42	0.45	0.46
9.81	0.44	0.46	0.46
10.17	0.40	0.43	0.42
10.47	0.39	0.42	0.42

Table 6: R2 score per iteration for each algorithm

Pruned (%)	Random Forest	Linear Regression	SVR
0	0.16	0.09	0.06
0.72	0.17	0.10	0.06
1.6	0.17	0.10	0.08
2.43	0.12	0.09	0.07
3.23	0.15	0.10	0.08
4.02	0.21	0.11	0.08
4.68	0.15	0.11	0.08
5.34	0.18	0.12	0.08
5.95	0.23	0.11	0.09
6.36	0.22	0.09	0.08
6.86	0.20	0.13	0.11
7.27	0.20	0.12	0.10
7.84	0.20	0.12	0.10
8.23	0.23	0.12	0.11
8.55	0.23	0.14	0.11
8.83	0.20	0.14	0.11
9.16	0.23	0.12	0.11
9.55	0.23	0.12	0.08
9.81	0.18	0.12	0.09
10.17	0.20	0.10	0.11
10.47	0.21	0.12	0.11

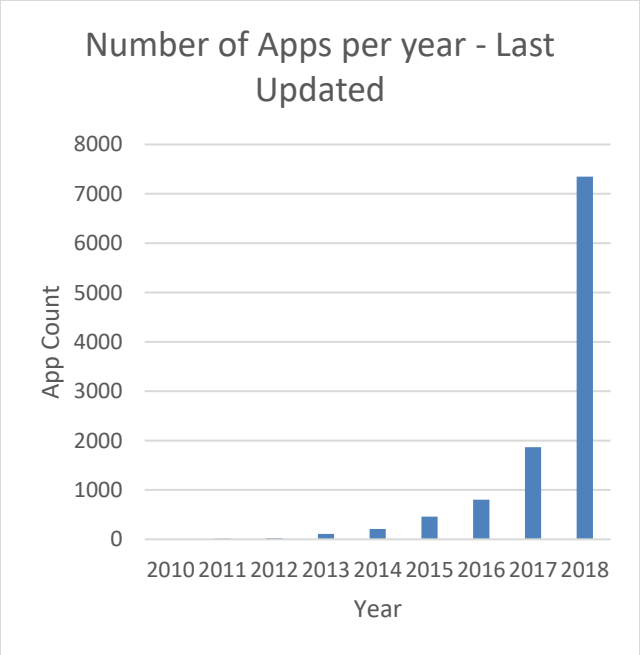


Figure 14: Year wise App Distribution