Predicting Ratings and Analyzing Root Causes for Customer Sentiments for Google Play Store Apps

Sonali Mahendran
Princeton University
sonalim@princeton.edu

Priscilla Lee
Princeton University
palee@princeton.edu

Rachel Protacio
Princeton University
protacio@princeton.edu

Abstract

The app-making business has tremendous potential for growth. While a lot of effort has gone into analyzing Apple App Store data, analysis of Google Play Store apps is just beginning to take root. In this project, we predict the ratings for Google Play store apps and focus our efforts in gaining a better understanding of what features are most predictive of an app's rating. We implemented regularized linear regression techniques to predict app ratings and LDA topic analysis to qualitatively assess what comments users leave about apps when they give them positive and negative ratings. From our analysis with regression, we found that an app is likely to receive high ratings if it is free, uses moderate memory, is well maintained, and boasts a large number of user reviews. Further, from our topic analysis of the customer reviews, we found that users appreciate apps that are easy to use and install, pose helpful or captivating content, are inexpensive, and display ads minimally.

1 Introduction

Customer feedback is a key determinant of an app's success in the market. In this study, we predict the ratings for Google Play Store apps and analyze customer reviews to gain insight into what attributes, according to customers, contribute to positive and negative ratings for an app. Analysis using linear regression facilitated careful investigation into determining which features are most predictive of an app's rating. Having this information and being able to predict ratings gives app designers a better idea of what features contribute to overall user satisfaction, and thus the success of a new app.

In this study, we use a dataset from August 2018 containing web scraped for 10k Google Play Store apps on Kaggle [1]. It contains 13 features (including name, category, average rating, number of reviews, number of installs, price, etc) for each of the 9,660 apps. The dataset also contains the top 100 "most relevant" user text reviews for 1,074 apps, including pre-processed sentiment features. The data did not require much imputation, and all of our feature engineering steps are enumerated in Section 3.1.

Our analysis of app ratings reveals that an app is likely to receive high ratings if it is free, uses moderate memory, is well maintained, and boasts a large number of user reviews. Further, from our analysis of the customer reviews, we found that users appreciate apps that are easy to use and install, pose helpful or captivating content, are inexpensive, and display ads minimally.

The rest of the paper is organized as follows. Section 2 presents related work about predicting ratings and other approaches to solving this problem. Section 3 details the linear regression methods we implemented to predict ratings, and Section 4 details the LDA methods we implemented to analyze user reviews to discern the attributes that make apps successful and unsuccessful. Section 5 details the findings of our investigation, and Section 6 discusses the implications of our findings and concludes this paper.

2 Related Work

Tremendous effort has gone into analyzing reviews for apps, material products, and services. Analysis of app markets is becoming increasingly popular as competitors in the app market strive to produce successful apps. They need to know what features make apps successful and what their customer base wants next.

User reviews are an essential component of app markets, like Google Play Store. We need techniques to sift through millions of reviews and summarize them in a meaningful way. Schneider et al. proposed an approach that uses the textual content of consumer-generated reviews for the sales forecasting of new and existing products. They used a dataset that contains product information and consumer review data collected from Amazon.com and from websites of several tablet manufacturers. They analyzed this data and discovered that textual content of online consumer reviews is a form of data that is an area of great potential in terms of informing better business decisions [2].

Fu et al.[3], proposed WisCom, a system to analyze millions of user ratings and comments in mobile app markets. More specifically, WisCom is able to discover inconsistencies in reviews, identify reasons why users like or dislike certain apps and how their preferences change over time, and provide valuable insights regarding the app market. They first analyzed individual comments to quantitatively determine whether users like or dislike an app; this model could then be used to identify Text-Rating-Inconsistency. They used a regression model on the vocabulary that users used in the reviews to perform comment-level sentiment analysis. The model assigned each word a numeric weight that measures its average influence on a rating. A regularized linear regression model with Tikhonov regularization and ℓ_1 norm was implemented, and the parameters were tuned to minimize quadratic loss. Fu et al. then conducted topic analysis via a 10-topic LDA model to determine the underlying causes for user satisfaction and user dissatisfaction. They then combined this topic analysis with time-series plots to study how user reactions to apps change over time.

Fu et al. proposed an interesting approach to sentiment analysis via linear regression. However, for the purposes of our project, we implement linear regression to predict app ratings based on app market data. Further, we also implement LDA to determine the root causes of user satisfaction and dissatisfaction, but because Fu et al. worked with over 13 million reviews, they were able to model the review with 10 topics. We only have reviews for 815 apps, and after visual inspection, we decided that between 3 and 5 topics were sufficient to summarize the reviews most meaningfully.

In addition to these studies, several users of this Kaggle dataset have posted kernels that offer starting points for exploratory analysis as well as ideas for model-fitting. In particular, User 'Last Night' first visualized the data and then qualitatively investigated how well individual features, like 'Number of Installs' and 'Price', predict ratings [4]. We improved on this analysis by creating multivariate regression models that represent how well a combination of the provided features for this dataset predict app ratings.

3 Ratings Analysis Methods

We first performed supervised analysis on the quantitative market data from the Google Play Store. We predicted app ratings, which lie on a continuous scale, using three linear regression models. To perform this analysis, we make the assumption that there is a linear relationship between the collected features and the app rating. The distribution of ratings is shown in Figure 1.

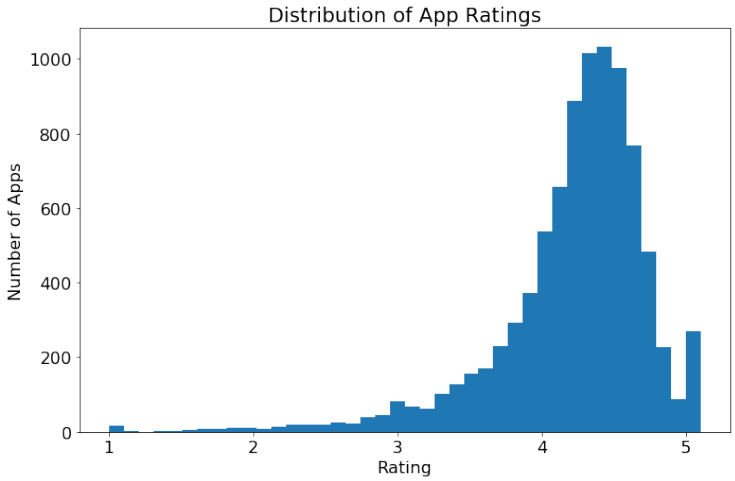

Figure 1: Distribution of Google Play Store app ratings.

3.1 Feature Engineering

Since our goal was to predict app ratings, we dropped 774 (out of 9,660) apps that did not have ratings, leaving us with 8,886 apps total. Each of these apps originally had 13 features that we parsed and engineered. We dropped a couple of features, such as app version and Android version, since we didn't expect them to be particularly predictive of ratings. We parsed numerical features, such as number of reviews, size in megabytes, number of installs, and app price. We also extracted one-hot categorical encodings for 33 categories, 6 content rating values, 115 genres, and 88 possible last updated month-year dates. This resulted in 247 engineered features.

Out of all of these engineered features, app size was the only feature with values that we had to impute. While most apps listed their sizes in numerical megabytes, 1,468 of them indicated "Varies with device". In order to impute this feature, we looked at the distribution of listed sizes and found a median of 14.0 and a mean of 22.76. After observing a steep skew in the distribution of app sizes, plotted in Figure 2, we decided to impute these 1,468 entries with the median size, 14.0 MB. As a final step, we normalized our 247 engineered features by scaling them to the range [0.0, 1.0].

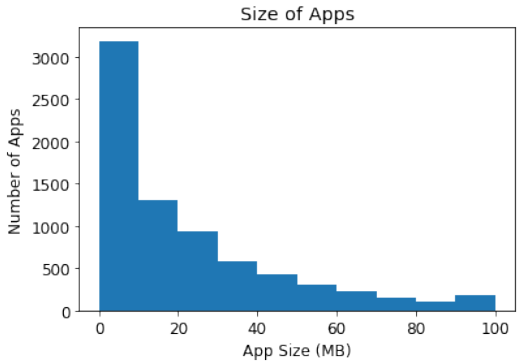

Figure 2: Distribution of listed app sizes.

We also incorporated qualitative review data for a separate round of model testing by using bag-of-words unigrams and bigrams (as detailed in Section 4.1) with TF-IDF vectorization, over all reviews. To turn the multiple review values per app into one vector, we took the mean TF-IDF for each app.

3.2 Regression Models

We used three regression methods from the SciKitLearn Python libraries [5] to model app ratings as a function of the other app features. All parameterizations are the default unless otherwise specified.

1. *Linear Regression*
2. *Ridge Regression*: using cross-validation on α
3. *Lasso Regression*: using cross-validation on α

Our cross-validation tested values of $\alpha = 10^x$ where $x \in \{-5, ..., 5\}$.

As suggested by [3], we used linear regression as a baseline model for rating prediction. Multivariate regression makes some key assumptions about the data. Features are treated as fixed value random variables and does not account for their distribution. Further, these models assume that the label is just a weighted linear combination of the features. The variance term is not a function of the features, and, finally, the residual errors and predictors are assumed to be independent. Despite these assumptions, linear regression has proven itself to be an essential data analysis tool [6].

In addition to ordinary least squares (OLS) linear regression, we also trained two regularized linear regression models. We used Ridge regression which applies an ℓ_2 penalty to weight magnitudes and therefore avoids overfitting. We also used LASSO regression, an ℓ_1 regularized linear regression, which favors sparse solutions and thus yields concise, interpretable results.

3.3 Training and Test Sets

For an initial exploration of our methods on the dataset, we handpicked features that we thought might be related to rating. In particular, we chose 'Installs' and 'Price' before opening our testing to all non-review-based features. This latter dataset of 8,886 apps with 247 features allowed us to predict a 8,886 x 1 vector of app ratings. We performed an 80-20 split our matrix for training and testing the performance of our models, resulting in a training set of 7,108 vectors and a testing set of 1,778 vectors.

Incorporating bag-of-words unigrams added 96 unigram features and 11 bigram features to our matrices but only included 815 apps with with sentiment labels. We tested each of our models on these matrices as well and compared against our models on the plain market data, limited to the same 815 apps.

3.4 Evaluation Methods

Since we split our data into training and testing sets, we were able to use mean squared error to evaluate our different regression methods. For a baseline performance metric, we used OLS linear regression, and our other regression models improved on this performance due to their penalty terms.

4 Review Analysis Methods

To gather unsupervised information about what influences the success of an app, we decided to explore the free-response text from user reviews. We isolated reviews by labeled sentiment and performed topic analysis. Thus, we were able to learn about what app attributes users discussed in positive reviews compared to negative reviews.

4.1 Bag-of-Words Feature Engineering

To learn what aspects of an app lead to positive reviews versus negative reviews, we separated the reviews into sets based on sentiment. We expected to retrieve meaningful topics from each sentiment of review. We separated our reviews by

(a) Labeled sentiment categories. This resulted in three sets: 19,015 positive, 6,321 negative, and 4,361 neutral reviews.

(b) Sentiment polarity values. The dataset maps values [-1.0, 0.0) to the negative category, a value of 0.0 to the neutral category, and values (0.0, 1.0] to the positive category. We chose the most extreme 0.5 values on each end to cull our sets, using values [0.5, 1.0] to select 6,226 positive reviews and values [-1.0, -0.5] to select 1,080 negative reviews.

For each set of reviews, we parsed them into a bag-of-words representation for use with LDA. Unigram parsing is insufficient to capture important qualifiers such as "not" or "very," however. So, we used both unigrams and bigrams in order to capture as much meaningful sentiment as possible. Our bag-of-words processing also included the removal of English stop words and low-frequency n-grams. Not only did this pare down less meaningful words, but it also reduced the number of dimensions for our final matrices. We did not need to ignore high-frequency n-grams for our separate sets of positive, negative, and neutral reviews since they might be indicative of each category.

We did, however, also test filtering our vocabularies into "unique sets" that represent the words that uniquely appeared in one set and not the others. This allowed us to ignore words that appeared in all categories and were therefore not indicative of good or bad app features.

We used regular word counts from SciKitLearn's CountVectorizer for our topic analysis, as well as Term Frequency - Inverse Document Frequency (TF-IDF) from SciKitLearn's TfidfVectorizer to test with our regression models from Section 3.

We tested our topic analysis model on unigrams alone, bigrams alone, and unigrams and bigrams together. We also tested over "full sets" versus "unique sets" and over sentiment label separation versus polarity value ranges.

4.2 Topic Analysis Model

As suggested by [3], we use Latent Dirichlet Allocation (LDA) for our unsupervised learning model since it is intended for use with bag-of-words corpora, for the purposes of discovering some k topics over n documents [7, 8]. In our case, our documents are user reviews, which may discuss various topics about an app. We use the LDA model from the SciKitLearn Python library [5]. All parameterizations are the default, including topic word prior η and document topic prior α values of $1/k\_components$. It is important to note that, as per [9], LDA assumes that each word, topic, and document are independent.

4.3 Evaluation Methods

The SciKitLearn LDA model's score method only provides an "approximate log-likelihood bound" and is therefore not very informative [5]. Lacking a good quantitative basis for choosing the number of components, we relied solely on visual inspection to select k. We also visually characterized our topics, as is standard in the literature [7]. To select the value k, the number of components, we tried out values in the range $\{3, ..., 9\}$ and visually examined the resulting topics.

Within the final topics, we looked at a number of informative aspects. Most importantly, we inspected the top 10 words per topic, i.e. the words that best define a given topic. We also looked at the distribution of topic percentages, being the number of reviews that were $p\%$ explained by a given topic.

5 Results

	MSE	α
Ordinary Least-Squares	0.259656	-
Ridge regression	**0.252109**	10
LASSO regression	0.252118	0.0001

Table 1: Mean squared error (MSE) of regression models for rating predictions and α value selected by cross-validation on all 247 normalized engineered features.

Ridge		LASSO	
Weight	Feature	Weight	Feature
-0.296318	'Last Updated: April 2012'	-2.574443	'Last Updated: April 2012'
+0.269750	'Last Updated: August 2018'	-0.442810	'Last Updated: February 2014'
-0.242402	'Last Updated: February 2014'	+0.435184	'Reviews'
-0.219325	'Genres: Educational'	-0.316770	'Price'
+0.211454	'Reviews'	-0.305086	'Genres: Educational'
+0.203995	'Last Updated: July 2018'	-0.281267	'Last Updated: October 2014'
-0.195444	'Price'	-0.279154	'Last Updated: January 2012'
-0.193072	'Last Updated: July 2014'	+0.268904	'Last Updated: August 2018'
-0.191116	'Last Updated: October 2014'	-0.266097	'Last Updated: July 2014'
-0.180970	'Genres: Card'	-0.263798	'Category: DATING'

Table 2: The top 10 (sorted by absolute value) features and their weights learned by Ridge and LASSO models.

5.1 Regression Results

Table 1 shows the mean square error (MSE) for each regression model's predicted ratings from our test set of 1,778 ratings over 247 normalized features. For OLS linear regression, there were 9 app vectors whose predicted ratings were on the order of 10^{10}, rather than in the feasible range [1.0, 5.0]. We therefore clamped the too-high predictions down to 5.0 for the sake of having a MSE to compare with our other models. As expected, the penalization from Ridge and LASSO regression greatly improve over this baseline, resulting in no predicted ratings above 5.0 in either model. We find that Ridge regression performs the best of all models, with a chosen value of $\alpha = 10$ from cross-validation.

In Table 2, we can see the most heavily weighted features in both the Ridge and LASSO models. It is interesting to see that, out of all the features of an app, the date of last update seems to be particularly predictive of an app's rating. Specifically, both Ridge and LASSO assign negative weights to older dates, such as 'Last Updated: April 2012' and 'Last Updated: February 2014', and positive weights to more recent dates, such as 'Last Updated: August 2018'. To put this into perspective, August 2018 is the most recent possible date in our dataset, since it was scraped that month. April 2012 and February 2014 are some of the oldest dates in the dataset, with the oldest value in the 'Last Updated' Feature being May 2010. These models indicate that apps that are well-maintained and up-to-date are likely to have higher ratings. Additionally, the weights of both models align with our intuition that apps with a higher number of reviews and a lower (and ideally free) price are more likely to have higher ratings.

The LASSO model, trained on 247 features, resulted in 97 features whose weights went to zero. These zero-weighted features included content rating, genres, categories, and last updated values. This seems to indicate that the rating of an app does not necessarily depend on its target audience (i.e. content rating, genre, or category), but more on its other qualities, such as price and up-to-date-ness.

Interestingly, including TF-IDF bag-of-words vectors with our market data did not improve our regression results. For OLS, it added too-low predictions below 0.0, and for our penalized regression models, the models had a worse MSE than with only market data. Many feature weights appeared to influence the prediction in the direction opposite of that expected. For example, features were assigned negative weights when we expected them to carry positive weights. For example, the unigram "pay" was assigned a positive weight even though our other analysis shows that paying for an app causes its rating to drop.

5.2 Review Topic Results

Since the neutral-labeled (i.e. 0-valued) reviews did not produce interesting results, we focus on the positive and negative sets for topic analysis. Moreover, the sentiment polarity sets yielded more meaningful results, which we discuss here.

We selected $k = 3$ for the unique-positive set and $k = 5$ for the unique-negative set. The results can be seen in Table 3. We can deduce labels for the topics in this table as they relate to attributes

Positive		
Topic 0	Topic 1	Topic 2
awesome	great	good
easy	best	love
amazing	perfect	nice
excellent	think	better
able	free	ok
use	thing	thing

Negative				
Topic 0	Topic 1	Topic 2	Topic 3	Topic 4
bad	worst	pay	useless	hate
annoying	game	customer	horrible	terrible
fake	disappointed	money	boring	stupid
wrong	load	don	unable	game
google	stars	install	awful	difficult
stop	getting	tried	makes	hated
people	hated	dont	load	data
getting	people	slow	people	stars
stars	data	disappointed	google	getting
install	slow	google	getting	boring

Table 3: Top words per topic for unique positive and unique negative unigrams.

of a successful app. For example, for the positive unigrams, Topic 0 indicates that 'ease of use' is important for a successful app, and Topic 1 indicates that free apps tend to receive positive ratings. Topic 2 seems to convey more middling opinions.

Figure 3 shows the distribution of topics as proportions over the reviews. We see that for the 'ease of use' Topic 0, over half of the 6,226 positive polarity reviews are [5% to 20%) explained by the topic. We can therefore conclude that usability is a common thread among positive reviews.

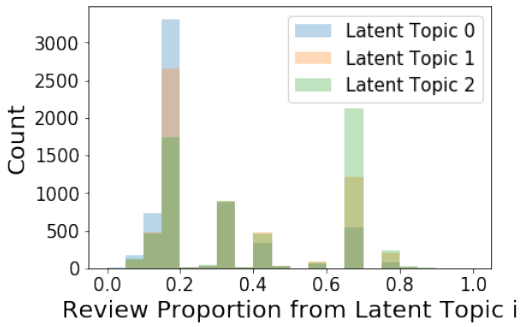

Figure 3: Overlayed histogram of (latent) topics among unique positive unigrams. Each bar shows the number of reviews out of the 6,226 positive polarity set that have some proportion explained by the Topic i.

From the Table 3 negative unigrams, Topic 1 suggests that apps that are slow or difficult to install amass negative reviews. Topic 2 revisits the idea that apps that cost money are likely to receive negative reviews. Finally, Topic 4 seems to reflect negative reactions to uninteresting games.

We selected $k = 5$ for both positive and negative sets of bigrams. The results can be seen in Table 4. A key observation that did not arise in our analysis of unigrams is apps that, according to Topic 0 for Negative bigrams, display too many ads tend to receive negative ratings. The topics obtained from analysis of bigrams are otherwise similar to that from the analysis of unigrams.

It is also worth noticing that positive topics mostly convey user contentment, while negative topics name more specific app issues.

6 Discussion and Conclusion

In this report we analyzed Google Play Store apps from two angles – applying regression to predict ratings and applying topic modeling to understand customer reviews. Regression emphasized features such as prize and date of last updated. These features were again highlighted in the topics from our customer review analysis. In particular, higher prices corresponded to poorer ratings and reviews, while app functionality and usability corresponded to higher ratings and reviews. We

Positive

Topic 0	Topic 1	Topic 2	Topic 3	Topic 4
i love	its great	great app	the best	very good
love it	very nice	good app	i think	i like
its good	i need	many ads	works great	easy use
i cant	great way	this best	really good	good i
love app	its really	app i	this great	i used
awesome app	i would	thank you	great i	love app
i wish	i loved	too many	it good	much better
its ok	very easy	this good	good work	love love
every time	this awesome	its awesome	i really	app it
i get	it is	best ever	great job	i found

Negative

Topic 0	Topic 1	Topic 2	Topic 3	Topic 4
cant even	worst ever	very bad	it is	i hate
this game	do not	worst game	hate it	i cant
i cant	keeps crashing	this worst	its bad	bad bad
very poor	it keeps	i hated	i want	i like
5 stars	get rid	i know	i dont	please fix
the game	worst worst	it bad	i think	hate game
annoying i	is bad	waste time	phone i	every time
ads annoying	horrible even	ever seen	app i	the worst
this bad	i get	bad game	i would	fix it
i cannot	even work	bad app	i try	time i

Table 4: Top 10 words per topic for positive and negative bigrams.

therefore conclude that developers seeking to create a well-received app should focus on usability and offer it at as low a cost as possible. They should regularly update the app and keep "annoying" advertisements to a minimum.

6.1 Limitations

Our regression methods make the assumption that the market features have a linear relationship with app ratings, meaning that they cannot capture nonlinear relationships. It is worth implementing linear regression with basis functions to see if this improves model performance, in the event that the relationship between market features and app ratings is not linear.

We made a few important assumptions in our work regarding the integrity of our datasets. In particular, we trusted that the user reviews and ratings were legitimate. If they contained many spam reviews, however, this could significantly change the features and words that we found to be meaningful for customer satisfaction. Additionally, for the review topic analysis, we trusted that the labeled sentiment attributes were accurate, and our results seemed to support this assumption.

It was not clear from the data whether the reviews and ratings were collected for all time for an app or just on the latest update, which could affect the accuracy of our discovered important features.

6.2 Future Directions

Given the relatively small size of our dataset, our work bears repeating with additional, possibly more detailed data. Using mobile app market data and reviews for non-Android apps could yield more robust models. To further drive production of good apps, being able to identify app features that are positive across platforms and distinct to individual platforms would be highly encouraged.

References

[1] Lavanya Gupta. Kaggle Google Playstore Apps Dataset. `https://www.kaggle.com/lava18/google-play-store-apps`. Accessed: 2019-04-17.

[2] Matthew J Schneider and Sachin Gupta. Forecasting sales of new and existing products using consumer reviews: A random projections approach. *International Journal of Forecasting*, 32(2):243–256, 2016.

[3] Bin Fu, Jialiu Lin, Lei Li, Christos Faloutsos, Jason Hong, and Norman Sadeh. Why people hate your app: Making sense of user feedback in a mobile app store. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1276–1284. ACM, 2013.

[4] Last Night. How to get "High" Rating on Play Store. `https://www.kaggle.com/tanetboss/how-to-get-high-rating-on-play-store`. Accessed: 2019-04-17.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] Engelhardt B. Linear regression lecture notes. Accessed: 2019-05-12.

[7] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. In *Advances in neural information processing systems*, pages 601–608, 2002.

[8] Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.

[9] Engelhardt B. Probabilistic topic models. Accessed: 2019-05-12.