**Google Play Store App Analysis**

Finding the highest accuracy model for analyzing Google Play Store

|  |  |  |
| --- | --- | --- |
| **Chia-Jung Chang** University of Pittsburgh Pittsburgh, PA 15260 chc276@pitt.edu |  | **Ming-Hsuan Chiang** University of Pittsburgh Pittsburgh, PA 15260 mic128@pitt.edu |

**ABSTRACT**

Our project mainly focuses on finding the best fit model for analyzing Google Play Store apps. We collected the Google Play Store Apps dataset from kaggle and used mainly the techniques that we learnt from machine learning lecture with necessary modification. Although the dataset is unbalanced, we evaluate the models not only with accuracy but also the F-measure to make it more reliable. Finally, we compared all of our results and concluded our model selections.

**Keywords**

Google Play Store, Mobile Application, Machine Learning, Classification, Unbalanced Data

**1  INTRODUCTION**

Mobile applications have become an important part in our lives. There are over 3.3 million available apps in the Google Play Store, and the market is still growing in a fast pace. It became important for developers to analyze the need and interests of human being and decide what kind of apps to launch. We are interested in the mobile application market performance and hope to find out the best fit model to classify google apps that have high rating, large installation, or huge review numbers. This analysis would be useful because we can also apply the methods and what we learned from the project to other real world unbalanced market data in the future. In the following sections, we first talk about works that are related to the original paper. Next, we analyze and preprocess the Google Play store dataset as well as our modification to the implementation. Then, we describe the concepts of approaches and ran the data by applying several classification algorithm that we learned from the lecture, including logistic regression, KNN classification, dummy classifier, decision tree, Naive Bayes, LDA, QDA, SVM, and neural networks model. After the progress of classification, we compare our results with different features and models to give out possible evaluations. The challenges we faced and limitations from several aspects are discussed for future improvements. Last but not least, we give a conclusion to the best fit model of the paper.

**2  RELATED WORK**

We tried to seek for related work making predictions with Google Play Store Apps. Since this is a topic intensely close to our daily life, we thought this should be a very popular topic in machine learning. Unfortunately, we surprisingly found that there are only few related work and they mainly target on detecting spam reviews or analyzing user reviews. Therefore, we focused more on finding relative work about analyzing any rating or reviews using any machine learning methods for our reference. Relative work about classification and mobile application prediction will be illustrated in this section.

In “Applying Naive Bayes Classification to Google Play Apps Categorization” by Babatunde Olabenjo, we learned that the author used cross-validation and the learning curve to determine the performance of each classifier. The paper realized that the Multinomial Naive Bayes classifier shows a better result than the Bernoulli Naive Bayes classifier. We applied this classification method to our dataset, however Multinomial Naive Bayes classifier doesn’t work best for all the time. Gaussian Naive bayes as a result works better when classifying installs and reviews.

There is a paper we read called “App Store Analysis: Using Regression Model for App Downloads Prediction” by authors Shanshan Wang, Wenjun Wu, and Xuan Zhou that revealed strong correlations between app downloads, app name score, app rank, and app rating. They used app downloads as dependent variable and three relevant attributes as independent variable to establish a multiple nonlinear regression model. We applied the concepts to predict our installation feature along with rating and reviews.

We also read “Predicting Movie Success Based on IMDb Data” by Nithin VR, Pranav M, Sarath Babu PB, and Lijiya A. This paper used the logistic regression model, linear regression model and the SVM model to predict the success of IMDb movies by judging the gross revenue and the rating. They find it more accurate when using the linear regression model with a 51% accuracy while the logistic regression model is 42.2% accurate and SVM regression model is 39% accurate. Although this article target on regression models and the regression models did not work well in our data, we tried the logistic regression model and the SVM classification model since we also desired to predict the success mobile application.

We thought of making use of the user\_reviews dataset because there are lots of potential in text analysis and sentiment analysis. There is an interesting paper called ‘Why people hate your app: Making sense of user feedback in a mobile app store.’ by Bin Fu, Jialiu Lin, Lei Li, Christos Faloutsos, Jason Hong, and Norman Sadeh that think user review is an important part of mobile app market as well. The paper use linear regression to detect inconsistencies in reviews, identify the reasons why users dislike or like the app, and last but not least, identify the major preferences and concerns of app. From micro to macro level of analysis, the concepts could be used in different secondary Android markets as well as other online trending business. In addition, this paper is the only one that consider not only accuracy, but also recall, precision and F-measure as evaluation method.

The challenge is that our user\_reviews dataset doesn’t have the keyID that could connect with the app information dataset. It would be difficult to give conclusion that relates to both app performance and user reviews so we decided to apply the technique in the future. Although we fail to make use of user reviews, we applied its evaluation idea to our project since our data is very unbalanced, and using simply accuracy to select final model is not reliable enough.

**3  PROBLEM AND DATA DESCRIPTION**

**3.1 Problem Setting**

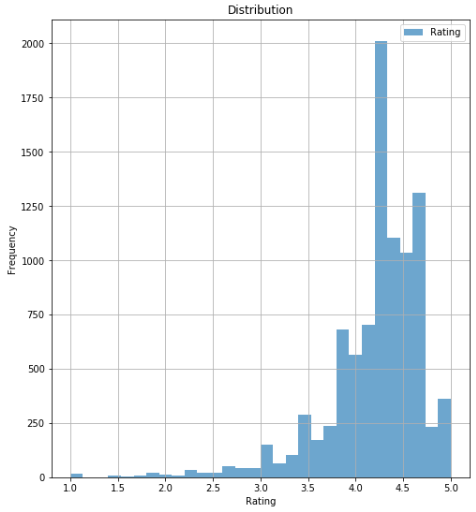
At first, We planned to make predictions of rating and price by finding important features and considering in the points below:

* What features are important to customers?
* How much does the customers willing to pay?
* Find out important interaction between features by exploring the correlations.

However, when we did the data cleaning and normalization, we found that the dataset is very unbalanced. The related papers we searched usually crawl their own google play store dataset so that they didn’t mention situation about unbalanced data. We didn’t crawl our own data due to time limitation.

Most of the data have high ratings since the overall rating average is around 4.2 (Fig 1). On the other hand, the price varies a lot, but most of them are free. Although we had another dataset that contains some users’ reviews, the lack of application id made it difficult to connect both datasets together for meaningful text and semantic analysis. The linear regression model performs badly.

After facing these problems, we asked for our instructor Dr. Mai Abdelhakim’s advice and changed our project to focus on classification problems. We selected five features including price, size, rating, installs, and reviews. Among the selected features, rating, installs, and reviews are our targets. We encoded the continuous rating and reviews feature to multiclass label and changed string type feature (Free and Paid) to binary class so that we could use them for our classification models. We used different approaches from machine learning class and applied them to our dataset. From logistic regression, KNN classification, dummy classifier to Decision Tree, LDA, and QDA, we compared our results with different features and settings of models to reach better performances and evaluations to find the best classification method for our google play store analysis. This could hopefully apply to future project that also contains unbalanced data.



**Figure 1: A skewed bar chart that shows the rating distribution.**

**3.2  Data Preprocessing**

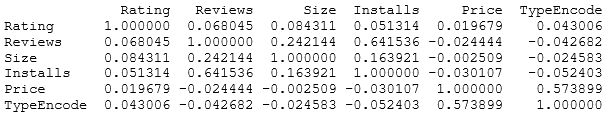
We collected the Google Play Store Apps data csv file from Kaggle. The dataset includes 10800 rows with 13 features that describes a given app. We find out there are some empty values and garbled information in the original dataset, so we decided to delete them. Also, there are some apps with incredibly high prices, as we explore further, we noticed that they are all spam apps with names like” I am very rich” or “ most expensive app”. We removed the data from the dataset to avoid them making any impact on our evaluation, and there are 9300 data left.

After importing our data into jupyter notebook, the information shows that the number of data varies from each columns, we figured that there are still some missing values so we used the dropna() function to remove them. For the data normalization, several features are not identical so we have to normalize them. For example, the Size feature in the dataset is not identical, some are just numbers and some include words like M or k (1.2M and 582k). The Price feature on another hand has ‘$’ in front of the value. There are also unnecessary symbols like ‘+‘ and ‘,’ from the Installs feature. We removed all the useless symbols to make them into numerical values for further comparison and processing.

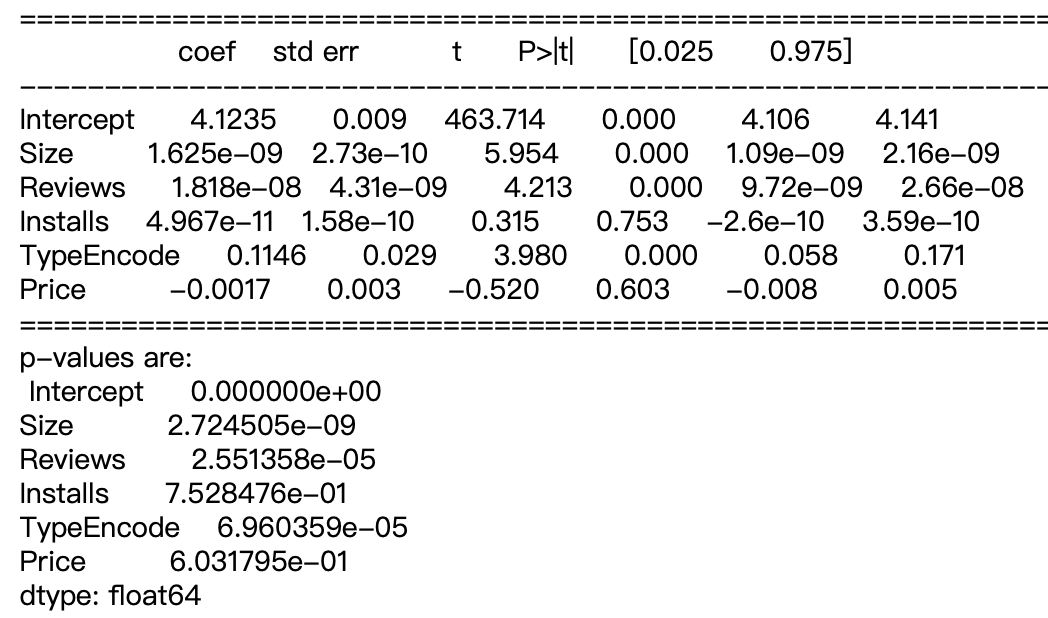
The type feature is a binary feature that very similar to the price feature, but not as specific to have numbers of price. We created another column for TypeEncode: Free as 0 and Paid as 1. In addition, We used panda.cut to categorize continuous numbers in rating and reviews in order to apply them to multiclass classification problems.

The corr() function in python could help us easily find the correlation between features and see which of them are important to the targets and how they are related to each other.

Figure 2 shows that reviews and installs is highly correlated to each other with a positive correlation of 0.64. We can assume that customers are more likely to download an app more if there are more reviews, and many active users would leave a feedback for other users’ reference.

**Figure 2: The correlation value between features**

If we use rating as target for an example, we could see from Figure 3 that size, reviews, and TypeEncode would have more significant impact on Rating because they have smaller p-values.

**Figure 3: The p-value between rating and other features**

**4  FORMULATION AND ANALYSIS**

We used several models including regression, but mainly classification methods in our project. After applying them to our dataset, we used accuracy, recall, precision, and F-measure to evaluate the result.

According to the class materials and the research papers we studied, accuracy is usually the most popular method to evaluate results, however, it is not convincing enough for unbalanced data. When the data is very unbalanced, it is likely to get a high accuracy by predicting test data into class that has a larger amount. In order to solve this problem, we also added the F-measure as another valuable measure, which is a combination of two important concepts, precision and recall. We interpreted the F-measure in python as a weighted average of precision and recall, where F-measure could have a value between 0 and 1.

* 1. **Linear Regression Model**

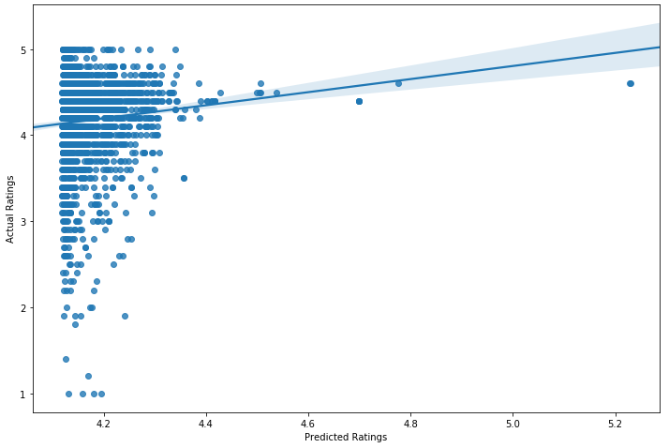
The idea of linear regression model is to seek for relationship between features with a linear function.

At first, we tried to implement the linear regression model to our dataset by predicting the rating and installations, but the result were terribly low with rating. In regression model, mean square error is a very common method used to estimate error variance. According to our results shown in Table 1 and basic knowledge, we conclude that it is not accurate to realize a dataset that is not linear with a linear regression model and the data distribution shows that our data is very unbalanced, so we consider that it should be more predominant to focus on the classification models instead. The installation shows a higher accuracy than rating, but the mean square error is intensely large. We believe the severely high mean square error is due to the large value range of the installation, the installation value range is from 1 to 1,000,000,000. The probability of predicting a correct installation is higher than rating, but once it predict wrong, will lead to extensively wrong.

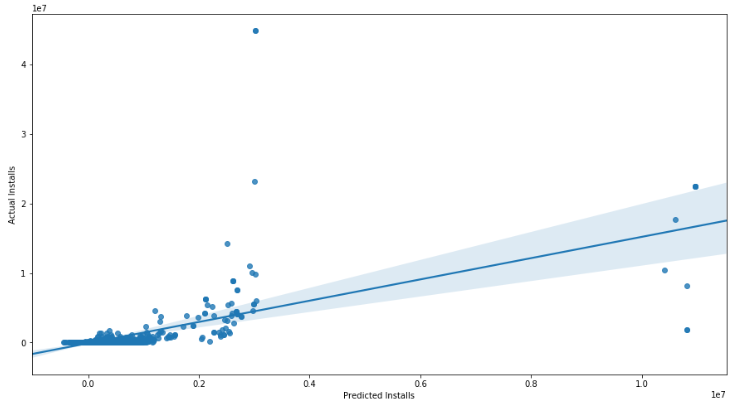
On the other hand, the performance of reviews seems quite similar to the installations. The accuracy is much higher than rating, but the mean square error is large as well.

**Table 1: Results of linear regression model**

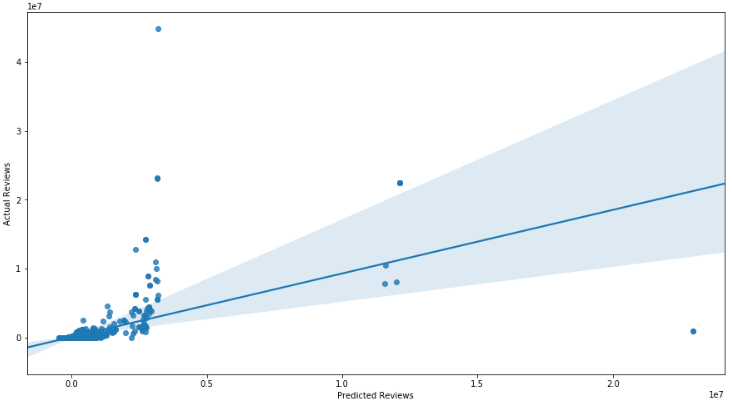
|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Accuracy | 0.0119 | 0.2303 | 0.3039 |
| MSE | 0.2860 | 1715218559356613 | 2553550794450 |



**Figure 4: Output of actual rating and predicted rating using linear regression model.**



**Figure 5: Output of actual installs and predicted installs using linear regression model.**



**Figure 6: Output of actual reviews and predicted reviews using linear regression model.**

**4.2 Dummy Classifier Model**

The dummy classifier could be our “baseline” measure of performance, because it represents the success rate one should expect to achieve even if simply guessing.

There are several strategies to use in a dummy classifier model, we choose three of them to implement to our case. We implemented stratified , most frequent and uniform. The “most\_frequent” strategy always predicts the label that is most frequent in the training data, the results show that our data has a higher accuracy in this case.

**Table 2: Accuracy results of dummy classifier model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Stratified | 0.6606 | 0.058 | 0.1739 |
| Most Frequent | 0.7029 | 0.001 | 0.2496 |
| Uniform | 0.7029 | 0.001 | 0.1603 |
| Best | 0.7029 | 0.058 | 0.2496 |

**Table 3: Recall score results of dummy classifier model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Stratified | 0.6585 | 0.0595 | 0.2068 |
| Most Frequent | 0.7029 | 0.001 | 0.2496 |
| Uniform | 0.7029 | 0.001 | 0.1347 |

**Table 4: Precision score results of dummy classifier model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Stratified | 0.564 | 0.0845 | 0.209 |
| Most Frequent | 0.494 | 1.091e-06 | 0.0623 |
| Uniform | 0.494 | 0.0988 | 0.1797 |

**Table 5: F- measure results of dummy classifier model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Stratified | 0.5919 | 0.1044 | 0.2078 |
| Most Frequent | 0.5802 | 0.0451 | 0.1 |
| Uniform | 0.5802 | 0.0685 | 0.15 |
| Best | 0.5919 | 0.1044 | 0.2078 |

**4.3 Models based on Bayes Theorem**

***4.3.1 Naive Bayes Classifier***

Naive Bayes classifier model is used in the related Google Play Apps research paper we’ve studied. Although they used it for classification to Google Play Apps Categories, and we focus on classifying rating, installs and reviews, we still decide to fit our data into this machine learning algorithms. Naive Bayes classifier is based on Bayes theorem, which uses conditional probability and assume that all feature has a condition that is independent. There are three types of Naive Bayes classifier model under the scikit learn library, including Gaussian, Bernoulli and Multinomial. The Gaussian Naive Bayes performed badly with the rating, because it is usually used when assuming all features are in a normal distribution, and the rating does not follow the normal distribution. We assume the Multinomial Naive Bayes to perform the best, because it is usually best used in a discrete dataset, the results show that BernoulliNaive Bayes and the Multinomial Naive Bayes has the same performance. As to the installs and the reviews, the Gaussian Naive Bayes performs the best, then the Multinomial Naive Bayes and the Bernoulli Naive Bayes. Although we presume the Multinomial Naive Bayes to perform the best in all features, the result is still acceptable.

**Table 6: Accuracy result of Naive Bayes classifier model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Gaussian | 0.1681 | 0.3843 | 0.3843 |
| Bernoulli | 0.7029 | 0.1755 | 0.1755 |
| Multinomial | 0.7029 | 0.2825 | 0.2825 |
| Best | 0.7029 | 0.3843 | 0.3843 |

**Table 7: F- measure result of Naive Bayes classifier model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Gaussian | 0.5802 | 0.3373 | 0.3373 |
| Bernoulli | 0.5802 | 0.0654 | 0.0654 |
| Multinomial | 0.3144 | 0.2951 | 0.2951 |
| Best | 0.5802 | 0.3373 | 0.3373 |

***4.3.2 Linear Discriminant Analysis***

The LDA model assumes the density function of the features in each class follows the Gaussian density distribution.

Since both Linear Discriminant Analysis and Quadratic Discriminant Analysis is based on Bayes Classifier, but even more popular when doing multi-class tasks. We decide to implement them into our project.

**Table 8: Results of LDA model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Accuracy | 0.7029 | 0.2052 | 0.5352 |
| F- measure | 0.5802 | 0.125 | 0.4977 |

***4.3.3 Quadratic Discriminant Analysis***

The QDA model also assumes the density function of the features in each class follows the Gaussian density distribution. However, while LDA models ignores the covariance matrix, the QDA model assume that each class has a different covariance matrix. Also, LDA models have a linear decision boundary while QDA models don’t.

At first, the QDA model performed a bad accuracy at 0.1812. We implemented MinMaxScaler and added a parameter to regularize the covariance estimate, the result improved a lot as shown below in Table 9.

**Table 9: Results of QDA model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Accuracy | 0.7029 | 0.3906 | 0.3906 |
| F- measure | 0.5802 | 0.3659 | 0.3659 |

**4.4 Logistic Regression Model**

We implemented the logistic regression model to our multiclass classification to softly define each class. The logistic regression model is an binary classification algorithm due to probability. We encoded rating and reviews into groups for classification, and installations happens to be discrete values which is suitable for classification. We used ridge regulation and StandardScaler in our model to improve the results, but the improvement is not very obvious.

**Table 10: Results of logistic regression model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Accuracy | 0.7039 | 0.3023 | 0.4715 |
| F- measure | 0.5763 | 0.0015 | 9.155e-05 |

**4.5 Decision Tree and Ensemble Methods**

***4.5.1 Decision Tree Model***

Decision tree models might not have the highest accuracy, but it has a very high interpretability. In decision tree classifier, low accuracy could be caused by overfitting problem. Since our data is not too big, we consider this is not the reason of our low accuracy. For the parameter setting, we set the maximum depth of the tree as 5 after trying out several numbers. As we can see from table 11, reviews shows a very high accuracy, but we don’t consider it a great fit since the F-measure is very low. In this case, we would not choose decision tree model as our best fit model even if the accuracy is high.

**Table 11: Results of decision tree classifier model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Accuracy | 0.7023 | 0.5415 | 0.7337 |
| F- measure | 0.5802 | 0.0031 | 9.164e-05 |

***4.5.2 Random Forest Model***

Random forest model have been very popular recently, due to it’s great efficiency and predicting ability, it is massively used in machine learning. We can consider it as a combination of several decision trees. In the parameter setting part, we set the same parameters for installs and review since the are highly related to each other and performs best using the same parameters. For rating, we set two trees in the forest with maximum depth of one. For installs and rating, we applied four trees in the forest with maximum depth of nine.

The installs has the highest accuracy when applying the random forest model among all other models. The accuracy of reviews is minorly lower than the decision tree classifier, but the F-measure performs much better.

**Table 12: Results of random forest model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Accuracy | 0.7029 | 0.5525 | 0.7321 |
| F- measure | 0.5802 | 0.5109 | 0.7333 |

***4.5.3 AdaBoost Model***

We implement the AdaBoost model to our dataset and expect a better accuracy than the decision tree model, because the AdaBoost model focuses on the areas where the former tree performed badly. For the parameter setting, we used n\_estimators = 5 to combine the trees in order to have better outcomes than decision trees. We did get a relatively high accuracy and F-measure in rating. However, installation and reviews feature did not show a better performance than the decision tree model.

**Table 13: Results of AdaBoost model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Accuracy | 0.7117 | 0.4997 | 0.7117 |
| F- measure | 0.7115 | 0.4159 | 0.7115 |

**4.6 Nearest Neighbors Classification Model**

The nearest neighbors classification simply computes the majority vote of closest neighbors of each point. This means the test sample would be assigned by the label according to the largest amount of label among its neighbors. Choosing a k for nearest neighbors classification is important, and we run k=15 to avoid overfitting.

**Table 14: Results of nearest neighbors classification model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Accuracy | 0.6658 | 0.3008 | 0.4851 |
| F- measure | 0.6244 | 0.2846 | 0.4892 |

**4.7 SVM Model**

The SVM model is very good at finding a decision boundary that can seperate classes whether in 2D spaces or in higher dimensional spaces.

We used the five fold cross validation to find the best tuning parameter for our final result in the SVM model. While rating has the best accuracy result with 0.1 as gamma and C, installs and reviews have the best result with 100 as their gamma and C value.

Among all models, the SVM model has an average performance. However, the processing time length is the longest, so we conclude that it doesn’t show a high efficiency.

**Table 15: Results of SVM model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Accuracy | 0.7029 | 0.3258 | 0.5352 |
| F- measure | 0.5802 | 0.3007 | 0.4977 |

**4.8 Neural Networks Model**

In the neural networks model, we applied StandardScaler to our dataset. In addition, we set the number of neurons in the hidden layer as 100 and 10, which indicates that the first hidden layer has 100 neurons and the second hidden layer has 10 neurons. We implemented the rectified linear unit function for our hidden layer activation function. Last but not least, for the optimization problem, we implemented the ‘lbfgs’ since it works well for small data.

**Table 16: Results of neural networks model**

|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Accuracy | 0.7039 | 0.4193 | 0.6235 |
| F- measure | 0.5883 | 0.3674 | 0.6078 |

**5 SIMULATION**

**5.1 Target and Features**

We tried to implement some machine learning algorithms from the papers we studied to our data by using them in different kinds of features in order to find out the best solution.

For target, ther research paper we studied has a great diversity of selecting targets, some use downloads ( which is the installs in our case)and ratin, while some uses the user reviews. As a combination, we decided to use installs and rating as mentioned above. We also applied the number of reviews as our target, since we don’t have application ID.

For features, since the selection of features depends on each datasets, the difference between datasets will lead to different implementation of features. In “App Store Analysis: Using Regression Model for App Downloads Prediction” , the authors focused on using rank, rating and name score to predict downloads of each category. As for “On the automatic classification of app reviews”, they focus more on user reviews, so the features are the stop words, lemmatization and the rating. The features varies from different data, so this is just for reference, not the main part of our simulation.

**5.2 Machine Learning Algorithms**

Besides the algorithms from related work, we added the other classification methods from class. In “App Store Analysis: Using Regression Model for App Downloads Prediction”, they focused on regression models. We tried fitting into regression model, however, our data performs badly with regression models. We changed our objective to classification models. Other research paper used Naive Bayes classification, decision tree classification, logistic regression and SVM regression. We tried all of the above model and other classification algorithm we studied from class.

On top of the concepts above, some classifier supports multiclass classification, for example, random forest and naive Bayes, but some of them don’t, like SVM, so we implemented OneVsRest strategy in our project.

**5.3 Evaluating method**

For evaluating method, most of them believe accuracy is adequate. On the other hand, “On the automatic classification of app reviews” seems like a more cautious paper, considering accuracy, recall, precision and F-measure as the evaluating method. Considering that we have an unbalanced data, we want to ensure our result to be more convincing, we implement the accuracy, recall, precision and F-measure for evaluation.

Despite of the evaluating method described above, we also used scalers and cross-validation to make sure we have better parameter setting and higher accuracy. The use of scalers did not lead to a significant change, but we still applied it for assurance.

**6 CONCLUSION**

Table 17 and table 18 shows the final results of our research. We choose dummy classifier as our baseline model, assuming all other result we consider should be better than dummy classifier. Accuracy is not reliable enough for evaluation, so F-measure is also the main purpose of our model selection. After comparing all the numbers, we think AdaBoost suits rating with highest outcome.  
Installs and reviews both have best results when using random forest.

Figure 7 and Figure 8 indicates that installs and reviews have similar result in both accuracy and F-measure. Their similar tendency might be due to their high correlation. In Figure 7, rating shows a high accuracy in most models, especially in classification models. However, we assume this result might be due to the unbalanced data. Figure 8 is the F-measure graph, showing some difference from the accuracy graph. All of them performs relatively better on random forest classifier and AdaBoost classifier. We could see that installs and reviews have bad results at logistic regression and decision tree classifier, so that even if the accuracy is high, we would not choose them as our best fit model.

Figure 9 is the overlook of the precision, recall, and F-measure of rating. They worked really well when using the AdaBoost classifier.

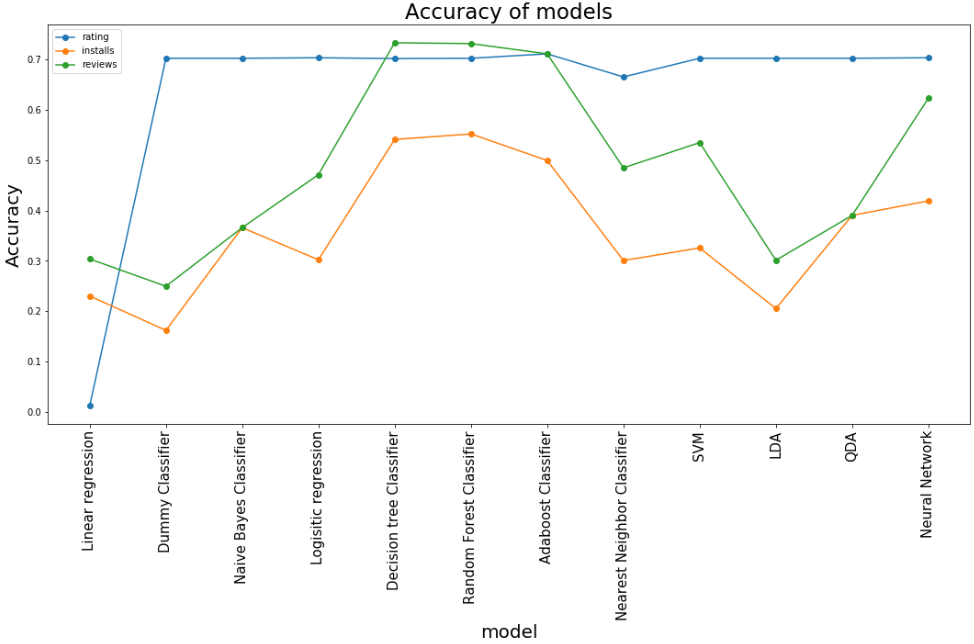
Figure 10 and Figure 11 shows the precision, recall, and F-measure of installs and reviews. As you can easily see, they have similar results on all three evaluating method. Installs and reviews have highest performance when using random forest classifier. They have bad outcome at logistic regression and decision tree classifier, but the other models are having better results than our baseline.

Although the decision tree classifier didn’t work well, the ensemble methods turned out to have great result. After comparing all the values, we think AdaBoost suits rating with highest outcome.

Installs and reviews both have best results when using random forest classifier. We hope the result will help us analyze and understand the google market more easily in the future.

**Table 17: Accuracy of all models**

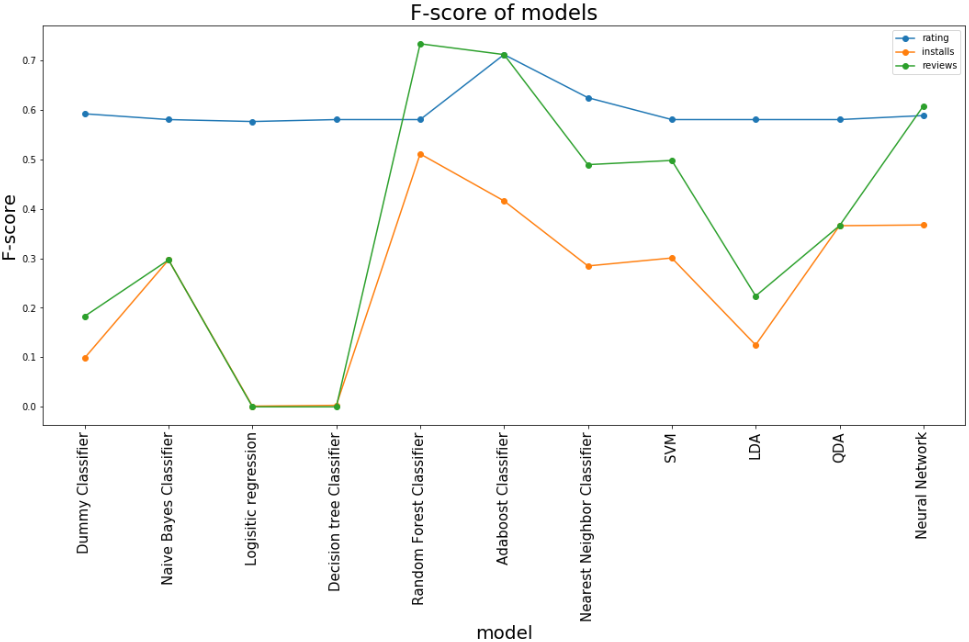
|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Linear Regression | 0.0119 | 0.2303 | 0.3039 |
| Dummy Classifier | 0.7029 | 0.1619 | 0.2496 |
| Naive Bayes | 0.7029 | 0.3843 | 0.3843 |
| Logistic Regression | 0.7039 | 0.3023 | 0.4715 |
| Decision Tree | 0.7023 | 0.5415 | 0.7337 |
| Random Forest | 0.7029 | 0.5525 | 0.7321 |
| AdaBoost | 0.7117 | 0.4997 | 0.7117 |
| Nearest Neighbor | 0.6658 | 0.3008 | 0.4851 |
| SVM | 0.7029 | 0.3258 | 0.5352 |
| LDA | 0.7029 | 0.2052 | 0.5352 |
| QDA | 0.7029 | 0.3906 | 0.3906 |
| NN | 0.7039 | 0.4193 | 0.6235 |



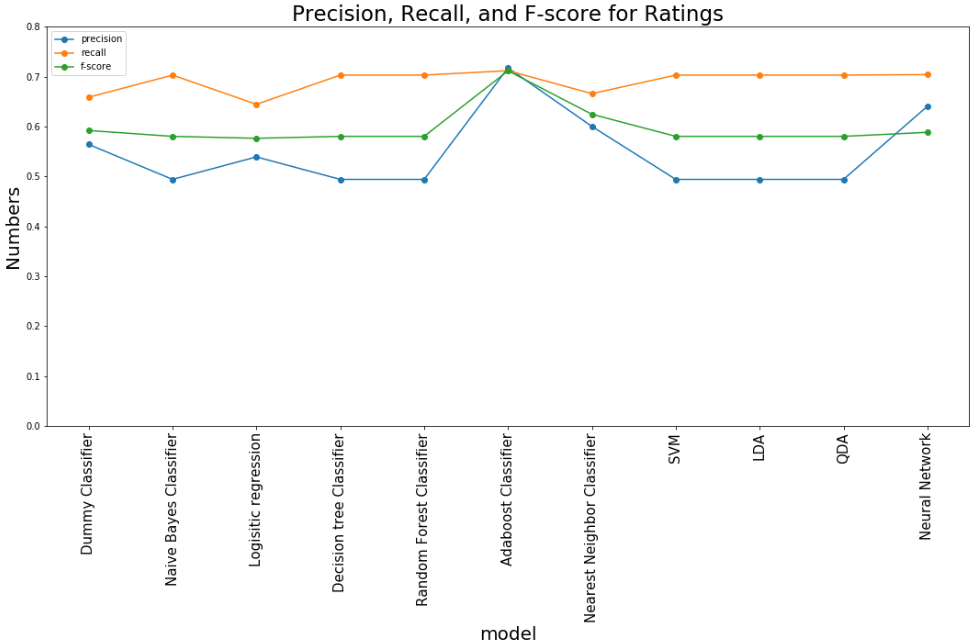
**Figure 7: Line graph of accuracy of each machine learning algorithm used in different features**

**Table 18: F-measure of all models used**

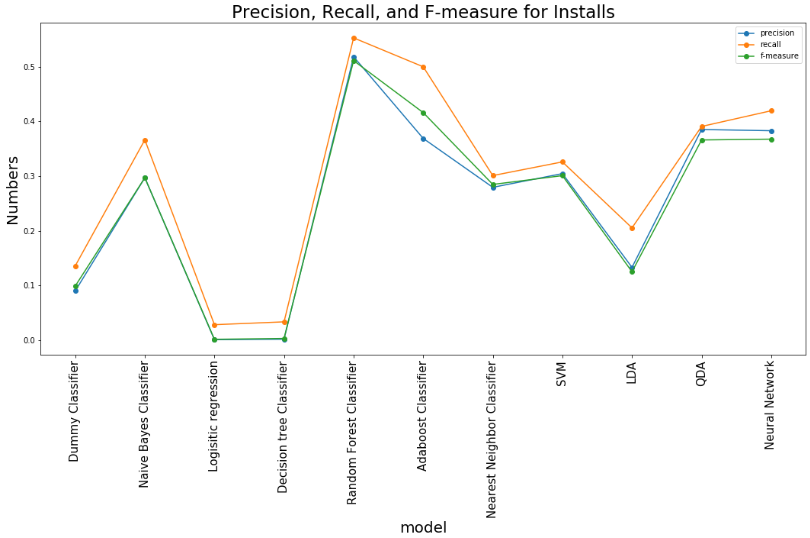
|  |  |  |  |
| --- | --- | --- | --- |
|  | rating | installs | reviews |
| Dummy Classifier | 0.5802 | 0.1044 | 0.2078 |
| Naive Bayes | 0.5802 | 0.3373 | 0.3373 |
| Logistic Regression | 0.5763 | 0.0015 | 9.155e-05 |
| Decision Tree | 0.5802 | 0.0031 | 9.164e-05 |
| Random Forest | 0.5802 | 0.5109 | 0.7333 |
| AdaBoost | 0.7115 | 0.4159 | 0.7115 |
| Nearest Neighbor | 0.6244 | 0.2846 | 0.4892 |
| SVM | 0.5802 | 0.3007 | 0.4977 |
| LDA | 0.5802 | 0.125 | 0.4977 |
| QDA | 0.5802 | 0.3659 | 0.3659 |
| NN | 0.5883 | 0.3674 | 0.6078 |

****

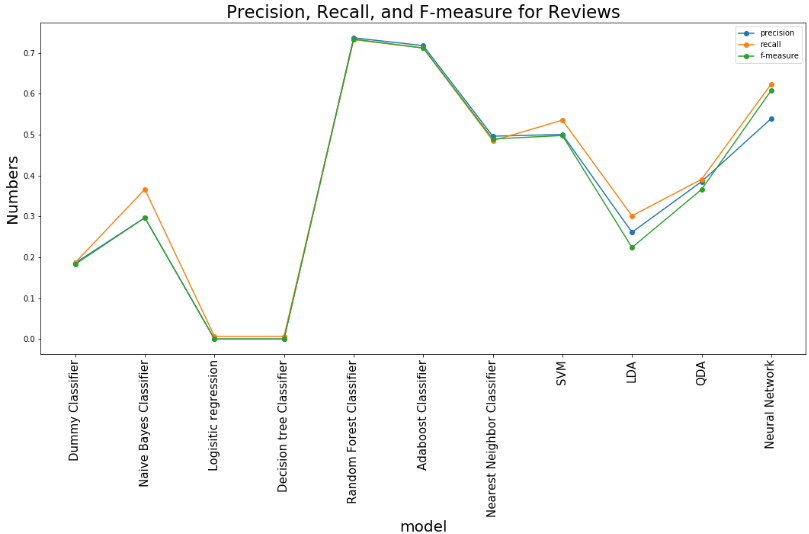
**Figure 8: Line graph of F-measure of each machine learning algorithm used in different features**

****

**Figure 9: Line graph of precision, recall and F-measure for rating using each machine learning algorithm**

****

**Figure 10: Line graph of precision, recall and F-measure for installs using each machine learning algorithm**

****

**Figure 11: Line graph of precision, recall and F-measure for reviews using each machine learning algorithm**

**ACKNOWLEDGMENTS**

We would like to thank Dr. Mai Abdelhakim for her instructions and suggestions in each milestone of the project sincerely. The data employed in this project is from the Kaggle website. We appreciate for authors and organizations to record and share those data.

**REFERENCES**

|  |  |
| --- | --- |
| [1] | Fushiki, T. (2009). Estimation of prediction error by using K-fold cross-validation. *Statistics and Computing, 21*(2), 137-146. doi:10.1007/s11222-009-9153-8 |
| [2] | Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J., & Sadeh, N. (2013, August). 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 (pp. 1276-1284). ACM.. |
| [3] | Hosmer, D. W., Hosmer, T., Cessie, S. L., & Lemeshow, S. (1997). A Comparison Of Goodness-Of-Fit Tests For The Logistic Regression Model. *Statistics in Medicine,16*(9), 965-980. doi:10.1002/(sici)1097-0258(19970515)16:93.0.co;2-o. |
| [4] | Louridas, P., & Ebert, C. (2016). Machine learning. IEEE Software, 33(5), 110-115. doi:10.1109/MS.2016.114 |
| [5] | Maalej, W., Kurtanović, Z., Nabil, H., & Stanik, C. (2016). On the automatic classification of app reviews. *Download PDF Requirements Engineering,11*(3), 311-331. doi:https://doi.org/10.1007/s00766-016-0251-9. |
| [6] | Mcilroy, S., Ali, N., Khalid, H., & Hassan, A. E. (2015). Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering, 21*(3), 1067-1106. doi:10.1007/s10664-015-9375-7 |
| [7] | Nithin. V. (2017). Predicting Movie Success Based On Imdb Data. *International Journal for Research in Applied Science and Engineering Technology, V*(X), 504-507. doi:10.22214/ijraset.2017.10074 |
| [8] | Olabenjo, B. (2016). Applying Naive Bayes Classification to Google Play Apps Categorization. arXiv preprint arXiv:1608.08574 |
| [9] | Wang, S., Wu, W., & Zhou, X. (2016). App Store Analysis: Using Regression Model for App Downloads Prediction. *Communications in Computer and Information Science Social Computing*, 206-220. doi:10.1007/978-981-10-2053-7\_19 |
| [10] | Qian, Y., Zhou, W., Yan, J., Li, W., & Han, L. (2014). Comparing Machine Learning Classifiers for Object-Based Land Cover Classification Using Very High Resolution Imagery. *Remote Sensing, 7*(1), 153–168. doi:10.3390/rs70100153 |