BANA 273. Machine Learning
Final Project Report
Team 11:
Poonam Ahuja, Vincentia Angelica, Haohui (Martin) Gui, Huiyin (Cloris) He

Prediction of App Installs in Google play store

Executive Summary

Introduction and Business Implication:

Mobile app distribution platform such as Google play store gets flooded with several thousands of new apps everyday with many developers working independently or in a team to make them successful. With immense competition from all over the globe, it is imperative for developers to know whether they are proceeding in the right direction. The Google Play Store is found to be the largest app market in the world. The Play Store apps data has enormous potential to drive mobile app businesses to success. Actionable insights can be drawn for developers to work on and capture the Android market. Since many apps are entirely free, the revenue model is quite unknown and unavailable as to how the in-app purchases, in-app advertisements and subscriptions contribute to the success of those apps. Thus, it would be more straightforward for us to see the success of an app from the number of its installation. In this project, we are trying to dive deeper into the Google Play Store data, discovering how several attributes could affect installs and use those variables to predict installs for future apps, in order to find out if the app could be successful if it was launched to the app store.

Our goal:

The main goal of this project is to help the mobile app developers to predict the range/class of installs for their new mobile applications given the attributes. By building this machine learning algorithms, it will help the developers to see whether the applications they build will fall into the range of install they expected. This prediction should give them a bigger picture of how big the market they are about to enter. For example, if a developer is creating an action-gaming app or a family-education app, this model should be able to give them the class of installations or a picture of the market size. If their expected installation range match the prediction, then the developers are ready to launch the applications. If not, they have the options to fix the model (price, size, content, etc.) or build a strategy for the mobile applications they build. We are hoping this machine learning model will better assist the developers in their decision-making process (launch/not launch). Thus, this will give them some confidence and save them a lot of time of reviewing if their mobile app model/concept (category of the app, size, type, price, and content) will work.

Dataset details and how we built it:

We obtained our dataset of Google Play store from Kaggle. We are predicting the installation number range (less than 10,000, 10,000+, 1,000,000+, 100,000,000+) using variables including category of the app, size, Installs, Genres, price, content (e.g. for everyone, teen, mature 17+, etc) so that we could find out the possible bins that the new upcoming apps could fall into.

We did our preliminary exploration of the data sets using Jupiter notebook, cleaning and transforming the data, isolating relevant features that would be used in our prediction model and visualization and ran our models in Weka.

Out of all the attributes, we selected 6 attributes in the dataset: Category, Size, Installs, Price, Genres, Content Rating and have done classification via Naïve Bayes and Random Forest algorithms to do the prediction on the number of Installation of apps with 10-fold cross validation estimate. Since initial accuracy with both the models were low without pre-processing, there were multiple rounds of data pre-processing (binning, discretization and resampling), which we did to enhance overall performance of the 2 models. Post this, Random Forest algorithm showed up with a better performance accuracy of 54.5% vs. 50.9% of Naïve Bayes.

To summarize, we brought out interesting insights from a Google Play App Dataset through interactive visualization.

Challenges we ran into

The biggest problem that we ran into, was how we should preprocess and clean the data to improve the overall accuracy of the model. Given the current limited attributes, the prediction accuracy is also limited. However, those attributes are also what the developers have at their app-development process so after various pre-processing steps, we came up with our prediction models on the install ranges with insightful information to developers.

Data: Data summary, description & 2 Techniques

The dataset we have is the data from Google Play store with 9660 lines in the dataset consisting of application name, category, rating, reviews, size, installs, type(paid/free), price, con tent ratings (age group the app is targeted at- Children/Mature 21+/Adult), genres (an app can belong to multiple genres apart from category, e.g., a musical family game will belong to music, game, and family genres), last updated, current version, and android version. On a separate file, we have the reviews of the applications, sentiment (positive, neutral, negative), sentiment polarity score, and sentiment subjectivity score.

Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	7-Jan-18	1.0.0	4.0.3 and up
AUTO_AND_VEHICLES	3.9	967	14M	500,000+	Paid	$4.99	Teen	Art & Design;Pretend Play	15-Jan-18	2.0.0	4.2 and up
BEAUTY	4.7	87510	8.7M	5,000,000+	NaN	$3.99	Everyone 10+	Art & Design;Creativity	1-Aug-18	1.2.4	4.4 and up
BOOKS_AND_REFERENCE	4.5	215644	25M	50,000,000+	0	$6.99	Mature 17+	Art & Design;Action & Adventure	8-Jun-18	Varies with d	2.3 and up
BUSINESS	4.3	167	2.8M	100,000+		$1.49	Adults only 18+	Auto & Vehicles	20-Jun-18	1.1	3.0 and up
COMICS	4.4	178	5.6M	50,000+		$2.99	Unrated	Beauty	26-Mar-17	1	4.1 and up
COMMUNICATION	3.8	36815	29M	1,000,000+		$7.99		Books & Reference	26-Apr-18	6.1.61.1	4.0 and up
DATING	4.2	13791	33M	10,000,000+		$5.99		Business	14-Jun-18	2.9.2	2.3.3 and up
EDUCATION	4.6	121	3.1M	5,000+		$3.49		Comics	20-Sep-17	2.8	Varies with de
ENTERTAINMENT	3.2	13880	28M	100,000,000+		$1.99		Comics;Creativity	3-Jul-18	1.0.4	2.2 and up
EVENTS	4	8788	12M	1,000,000,000+		$9.99		Communication	27-Oct-17	1.0.15	5.0 and up
FINANCE	NaN	44829	20M	1,000+		$7.49		Dating	31-Jul-18	3.8	6.0 and up
FOOD_AND_DRINK	4.8	4326	21M	500,000,000+		$0.99		Education;Education	2-Apr-18	1.2.3	1.6 and up
HEALTH_AND_FITNESS	4.9	1518	37M	50+		$9.00		Education	26-Jun-18	NaN	1.5 and up
HOUSE_AND_HOME	3.6	55	2.7M	100+		$5.49		Education;Creativity	3-Aug-18	3.1	2.1 and up
LIBRARIES_AND_DEMO	3.7	3632	5.5M	500+		$10.00		Education;Music & Video	6-Jun-18	2.2.5	7.0 and up
LIFESTYLE	3.3	27	17M	10+		$24.99		Education;Action & Adventure	7-Nov-17	5.5.4	5.1 and up
GAME	3.4	194216	39M	1+		$11.99		Education;Pretend Play	30-Jul-18	4	4.3 and up
FAMILY	3.5	224399	31M	5+		$79.99		Education;Brain Games	20-Apr-18	2.2.6.2	4.0.3 - 7.1.1
MEDICAL	3.1	450	4.2M	0+		$16.99		Entertainment	20-Mar-18	1.1.3	2.0 and up
SOCIAL	5	654	7.0M	0		$14.99		Entertainment;Music & Video	12-Jul-18	1.5	3.2 and up
SHOPPING	2.6	7699	23M			$1.00		Entertainment;Brain Games	7-Mar-18	1.0.8	4.4W and up
PHOTOGRAPHY	3	61	6.0M			$29.99		Entertainment;Creativity	7-Jul-18	1.03	7.1 and up
SPORTS	1.9	118	6.1M			$12.99		Events	25-Apr-18	6	7.0 - 7.1.1

This is the list of unique values for each column. Without the dataset for customer reviews.
Categorical: Category, Installs, Type, Content Raring, Genres
Continuous: Size, Price

Since the goal of the model is to predict the range of installs, the class of the model is ‘installs’. In this dataset, ‘installs’ has 21 unique values (1,000+; 10,000+; 100,000+; 5,000+; 50,000+; 500,000+;…). In order to achieve the goal of the model, we are going to pick some attributes that

will help the model to make a good prediction. Though we have many attributes to pick, we can take only 5 attributes on our model: category, size, price, content rating, and genres. We couldn't take rating, reviews, and the sentiments because the model is intended for the developers who create new applications. For a new application, the information of rating, review, and sentiment will not be available. The developers will only have data about the application they are building. That's why we think it is very reasonable for us to only use those 5 attributes from the dataset to build our model.

We are going to use 2 algorithms techniques for our prediction model as a comparison. The first technique will be Naïve Bayes and second one will be Random Forest. With small number of features in our model, we decided to pick Naïve Bayes since Naïve Bayes works best with smaller dimension data. We are planning to process our data into all categorical, this gives us another reason to pick Naïve Bayes. As the goal of our prediction model is to correctly classify the range of install, Random Forest is another appropriate algorithm to do this prediction. Apart from that, Random Forest can also help us to predict with better accuracy. Compare to decision tree, Random Forest will help us to avoid the overfitting problem. However, these 2 algorithms have their own disadvantages. Naïve Bayes tend to assume that the features are independent and have the same level of importance. While Random Forest, it will create an overfitting problem if we model is not carefully built. Also, Random Forest is slower compare to Naïve Bayes since the algorithm computation is complex. We will try these 2 algorithms for our prediction model and see which one will do a better job in terms of accuracy and stratified accuracy.

Pre-processing and models

Classification with Naïve Bayes and Random Forest:

We have 5 attributes in the model: Category, Size, Price, Genres, Content Rating, with installs as the class. We are using Naïve Bayes and Random Forest algorithms to do the prediction on the number of Installation of apps with 10-fold cross validation estimate. After we converted the price variable into floats, we ran our model. However, the prediction accuracy is low.

Model Results:

Naïve Bayes:

Correctly Classified Instances	1930	17.8028 %
Incorrectly Classified Instances	8911	82.1972 %

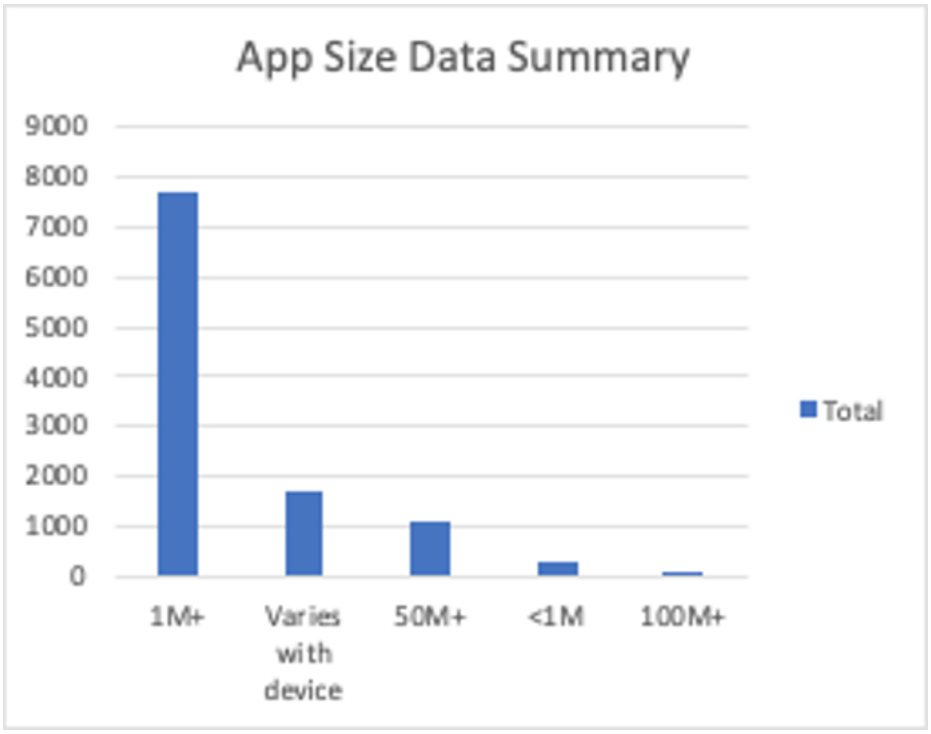
Random Forest:

Correctly Classified Instances	2486	22.9315 %
Incorrectly Classified Instances	8355	77.0685 %

We noticed that the Random Forest model did a better job, but the result was still not promising. Since we found out in the data that price and size are continuous variable and the others are categorical variables, we think that some of the categorical variables are put into small groups with few instances and that's the possible reason that our model did not perform well. Additionally, the continuous price attribute should also be discretized for a better fit of the algorithms. We would

take a few binning and discretization pre-processing approaches to discretize the price and put the data into bigger size of bin to see the performance of the model. Therefore, we performed several steps to see how the prediction accuracy would change in the two models.

1. **Bin size of app: Convert data from continuous to categorical**



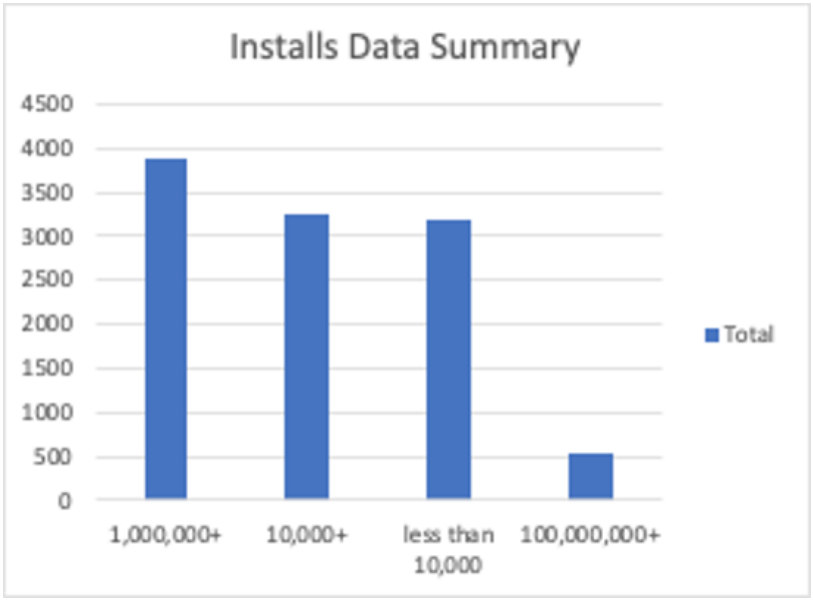
Model Results:
Naïve Bayes:

Correctly Classified Instances	1875	17.2955 %
Incorrectly Classified Instances	8966	82.7045 %

Random Forest:

Correctly Classified Instances	2292	21.142 %
Incorrectly Classified Instances	8549	78.858 %

2. Bin size of app + **installs: Increase the size of bin by reducing the class of installs**



Model Results:

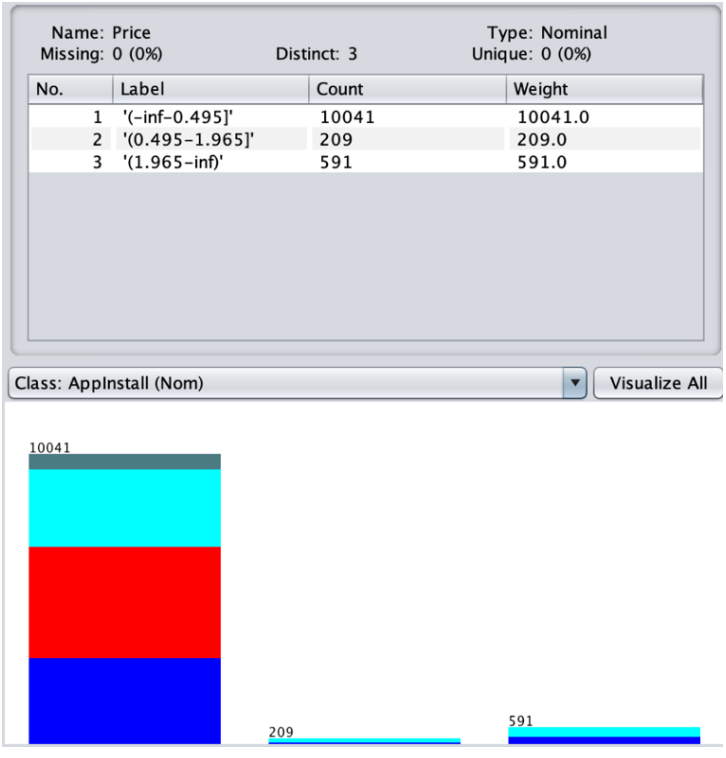
Naïve Bayes:

Correctly Classified Instances	4665	43.0311 %
Incorrectly Classified Instances	6176	56.9689 %

Random Forest:

Correctly Classified Instances	5675	52.3476 %
Incorrectly Classified Instances	5166	47.6524 %

3. Bin size + install & **discretize price: Convert the price data from continuous to categorical with Weka**



Model Results:
Naïve Bayes:

Correctly Classified Instances	5497	50.7057 %
Incorrectly Classified Instances	5344	49.2943 %

Random forest:

Correctly Classified Instances	5678	52.3752 %
Incorrectly Classified Instances	5163	47.6248 %

4. Bin size + install & Discretize price **& Resample**

Resample: Weka --> Filters--> Supervised-->Instance--> Resample (with replacement)

Model Results:
Naïve Bayes:

Correctly Classified Instances	5522	50.9363 %
Incorrectly Classified Instances	5319	49.0637 %

Random Forest:

Correctly Classified Instances	5917	54.5798 %
Incorrectly Classified Instances	4924	45.4202 %

After we tried binning, discretization and resampling, we found out that these pre-processing steps are necessary and valuable to improve the performance of the models. We chose to bin the size of app and installs because these two categorical variables are separated into small groups and binning them into a large group would make it more representative. Additionally, because the price variable is a continuous variable, we discretize it to make it categorical so that it could perform better in those two algorithms. The resample techniques is also helpful as it increases the random forest model by 2% and a slight increase in the Naïve Bayes model. Thus, the random forest model after all the pre-processing approaches would be our best performance model.

Data Findings
After picking random forest as our prediction model, we decided to dive deeper to each of the classes. Below is the detailed accuracy by class. As we can see in the table below, our model is doing the best job at predicting the class 1,000,000+ and the worst at predicting class 100,000,000+. When we look at each of the TP rate, the model was able to correctly classify group 1,000,000+ and <10,000 at the rate of 60%+. However, those 2 classes are experiencing more (FP) type 1 error as well (20%+).

With the False Positive rate in the detailed accuracy table, it shows that errors happen the least for 100,000,000+ group but higher for the less than 10,000+ group.

```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.373    0.184    0.464      0.373    0.414      0.203    0.691     0.473     10,000+
               0.643    0.215    0.625      0.643    0.634      0.426    0.804     0.704     1,000,000+
               0.635    0.247    0.517      0.635    0.570      0.368    0.797     0.599     less than 10,000
               0.354    0.013    0.579      0.354    0.440      0.431    0.944     0.528     100,000,000+
Weighted Avg.  0.546    0.205    0.543      0.546    0.540      0.342    0.775     0.595
```

Additionally, when we run the random forest model, we decided to get the attribute importance as well to see the contribution of each attribute to the model.

```
Attribute importance based on average impurity decrease (and number of nodes using that attribute)

      0.33 (   537)  Genres
      0.29 (  7564)  AppSize
      0.29 (  8332)  Price
      0.26 (  1833)  Category
      0.21 (  8559)  Content Rating
```

As we can see from the result above, genre is the most important attributes in the model and followed by app size, price, category, and content rating.

```
=== Confusion Matrix ===

    a    b    c    d   <-- classified as
 1210  778 1234   19 |    a = 10,000+
  650 2491  624  107 |    b = 1,000,000+
  725  426 2025   13 |    c = less than 10,000
   24  293   31  191 |    d = 100,000,000+
```

With the given model and condition of our case, the errors that happen would be worse if the model group has low installs but classified in a higher install group. This would give an impression to the developers that their apps might do great, but the truth is not. Therefore, this overprediction might not be good for the developers since they might lose motivation, incentive to launch the app, make further improvement or updates.

If we further investigate the confusion matrix, it is worse for Group a (10,000+) to be grouped as b or d but the risk is lower if they are classified as c because Group c is the lowest group. The error rate for overpredict is calculated as below:
(778+19)/(1210+778+1234+19) = 24.59%

And for Group b (1,000,000+), it will be worse for them to be grouped into d (underpredict) rather than grouped into a and c (overpredict). The error rate for overpredict is calculated as below:
(107)/(650+2491+624+107) = 2.76%

For group c (less than 10,000+), because it's the lowest group, it's not good for it be wrongly classified into any other groups. The error rate for overpredict is calculated as below:
(725+426+13)/(725+426+13+2025) = 36.5%

For group d (100,000,000+), it wouldn't have an overpredict problem since it is the highest class.

To summarize, we are aiming for a less percentage of error of overprediction. Below is the percentage of error for each class:

Minimize Error: C classified as A, B, D A classified as B, D B classified as D	Underpredict % A: 38.07% B: 32.9% D: 64.56%	Overpredict % C: 36.5% A: 24.59% B: 2.76%

To minimize the overprediction error, we have to minimize the error of: (1) C classified as A, B, D, (2) A classified as B, D, (3) B classified as D

Though the prediction is not perfect, but if we look the percentages in the table above, we are doing a pretty good job. The overprediction percentages for A and B are smaller than the underprediction.

For C, since it is the smallest range in the class, all the errors are overprediction errors. Class C is expected to make as little error as possible. Compare to the overall error rate, Class C is doing a good job compare to the model overall error rate. The error rate for C is 36.5% while the error rate of the model is 45.4%.

Takeaways

It is statistically interesting to experience that how the outcome and quality of the model varies with different selection of data-mining and algorithm methods that we chose. We practically pick up on the fact how critical these distinctive data mining and algorithms could tremendously affect the entire data analysis process. It is interesting to see how the prediction accuracy would go up after the pre-processing techniques. We are very glad that we could apply all these methods to this final project. Overall, it's been a good learning curve experience. The followings are some listed takeaways from this project.

In respect of Naïve Bayes, it didn't seem to generate promising outcome without pre-processed dataset in the first place because Naive Bayes is a simple and fast classification. However, Naive Bayes assumes that all attributes are independent which is rarely happening in the real world. The 6 attributes we selected (Category, Size, Installs, Price, Genres, Content Rating) are realistically correlated to some extent, which might violate the Naïve Bayes' assumption. Furthermore, since this dataset we chose is comparatively large, Naïve Bayes is generally better at prediction on a smaller dataset. This is another issue that might cause the low accuracy classification outcome. Additionally, Naïve Bayes works better at categorical data. And thus, after we discretized the price variables which is a continuous variable, the performance of Naïve Bayes goes up for about 7%.

In terms of the Random Forest model, this is the overall best technique that outperforms Naïve Bayes on this final project. It is because whether we have a regression or classification task, random forest is an applicable model for our needs. It can handle binary features, categorical features, and numerical features and thus, it is easier to deal with many types of data. The vastest element of utilizing random forests is feature reduction and balanced class weights that played an important role on this dataset-mining process.

Results Interpretation and Implications

After comparing the two machine learning algorithms with Naïve Bayes and Random Forest. We have comparatively higher accuracy (54.6%) outcome compared with Naïve Bayes algorithm. Fundamentally, the goal is to efficiently assist Google play application companies and developers to predict the class of installation of the new application (less than 10,000, 10,000+, 1,000,000+, 100,000,000+) based on the attributes including Category, Size, Price, Genres, Content Rating in order to maximize the confidence of developers before launching the application. Even though the prediction accuracy doesn't seem high here, the number is still better than random. Though, we do not have many attributes in our model, I think we are being realistic as the new app will not have too much information during the app development process. With these commonly collected attributes (Category, Size, Price, Genres, Content Rating) in the application development process, we will be able to predict and catch which installation groups a certain application will fall into.

As most developers are making application to eventually generate sales and revenue, the number of installs play an important role on representing the financial outcome for application developers because it is a fundamental indicator to reflect on how well the developers are, financially. With all the cost and expenses (advertising, labor, subscriptions fee, physical equipment, etc), the developers need an indicator to decide if they are ready to launch. Additionally, with this prediction, they are able to plan a rough pre-launch marketing strategy as well, such as: cost per action, cost per acquisition, cost per click and cost per install networks. Thus, installation number will be a straightforward way for us to access whether the app could be successful or not. Using our prediction model to generate the basic picture of what possible installation the apps could have, developers could adjust and improve their apps accordingly.

Other Recommendations

With the model we build, we could bring 2 types of benefits: Direct and Indirect. The direct benefit is predicting the class of installations for the new apps before launching for the developers to give them some confidence to launch the app if the prediction matches their expectation. However, if the prediction doesn't match their expectation, we believe that the prediction can be used for an even better job, such as strategy planning.

We are hoping that new apps could use the prediction to improvise their new application before the launch. Not only in terms of the capability of the application, but also from the business model/strategy point of view.

If the predicted class installs are lower and not as what the developer initially expected, it will be great if the developer could find a way and take some efforts to improve the apps. However, they

should try to look at the problem from a different point of view. For example: marketing point of view. Does the app have the correct target market (Content Rating)? Does the price of the app reflect its value?

If there is nothing could be improved to boost up the installs, they could find a way on how to cut the losses and do a strategy planning. Creating an app is not free and there are some costs that the developers need to pay. If generating profits is unlikely, at least they would be able to find a way to reach the break-even point.

In a bigger application developing company setting, mobile application marketing team starts long before developers ever bring it to market. Before the marketing team starts telling the developers to start building up apps with a million different things, they will reach out to their customers and talk to them. After all, they want to make sure that whatever they are building is going to be the best fit for customers' best interests. And that's why the installation number is also important for marketing since installation number also could indicate our customers' interests and enthusiasm.

If the app doesn't have a promising installation number, the marketing team will need to find out whether the direction of their apps is right or whether the apps need to have more exposure. Marketing their app can extend beyond their usual field of influence and often times, these foreign connections are exactly what their company needs. Developing honest, authentic, and mutually beneficial relationships with influencers will go a long way in helping to promote their application.

If they already have an existing website that is fully functional and mobile-friendly, then the site can be one of the greatest assets to promote their apps. At one point or another, all of their customers or anyone interested in their company will find a way to the website. When they do, it is important for the company to let their customers know their new apps.

Installation number of the apps is important as it could indicate many business perspectives. Our project is mainly giving developers a sense of how well the apps might do in terms of installation number. If they find out the installs is not what they expected, then not only the developers need to do something to improve the apps, but also the other body of the organization should take efforts to find out the reasons and then fix the problem.

References

https://www.kaggle.com/lava18/google-play-store-apps?select=googleplaystore.csv
http://www.bluecloudsolutions.com/2015/02/06/how-many-downloads-should-my-app-get/