GOOGLE APPS DOWNLOAD PROJECTION

BY XU HAN, DATA SCIENCE SCM

HTTPS://GITHUB.COM/BAO1981105/GOOGLE-APPS-DOWNLOAD-PREDICTION

INTRODUCTION

While many public data sets provided Apple store data, there is not enough analysis into apps in Google Play store, partly because it's more difficult to scrape data from Google Play store.

Nowadays, as mobile phones become more popular, people tend to spend more time on their phones and app usage increases a lot. While this is a great opportunity for many app developers, it also becomes a challenge for many developers/businesses to develop a popular app.

In this project, I will try to predict the number of installs (target variable) from some features of the app itself, displayed in google play store app page. I am trying to find out what kind of apps are more popular and tend to stay longer in people's phones. Since the exact number of installs was not available, but an estimate was given, it is treated as a categorical variable, so the problem becomes a classification problem.

DATASET DESCRIPTION

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	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
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	15-Jan-18	2.0.0	4.0.3 and up
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	1-Aug-18	1.2.4	4.0.3 and up
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	8-Jun-18	Varies with device	4.2 and up
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	20-Jun-18	1.1	4.4 and up
5	Paper flowers instructions	ART_AND_DESIGN	4.4	167	5.6M	50,000+	Free	0	Everyone	Art & Design	26-Mar-17	1	2.3 and up

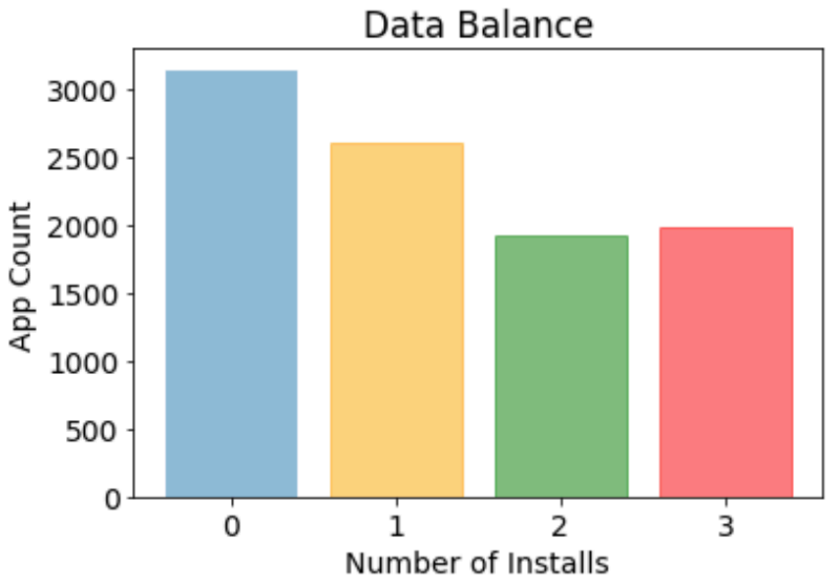
This dataset was downloaded from Kaggle and scraped from Google play store. The dataset was updated eight months ago.

1. googleaplystore.csv. This file has 10,841 entries, each representing an app listed in Google Store, and it has 13 columns. These columns are:
    - App: name of the App.
    - Category: categorical.
    - Rating: 1 to 5 (5 being the highest), continuous.
    - Number of reviews: continuous.
    - Size: file size, continuous. After preprocessing, all measured in megabyte, but some say "vary with device".
    - Number of installs: categorical target variable. After preprocessing, values include: <10k, <500k, <5 million, >5 million.
    - Type: categorial, Paid or Free.
    - Price: continuous, measured in US dollar.
    - Content Rating: categorical variable. Everyone, Everyone 10+, Teen, Mature 17+, Adult only 18+, Unrated. Unrated apps are treated like high-maturity apps until they get a rating.
    - Genres: categorical variable, similar to Category, but provides more detailed types.
    - Last Updated: last date this app was updated, at the time when this dataset was made.
    - Current Ver: current version of the app, treat as categorical. Some say "Varies with device".
    - Android Ver: Android operating system requirement, treat as categorical. Some say "Varies with device".

2. googleplaystore_user_reviews.csv
    - 64,296 entries, each representing one review. Columns: App, Translated_Review, Sentiment, Sentiment_Polarity, Sentiment_Subjectivity.

EDA

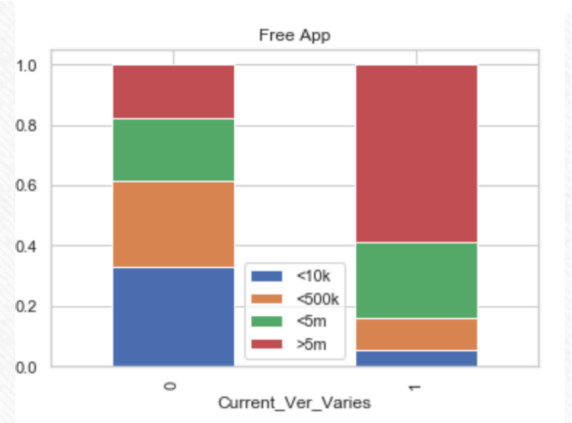
Below shows the balance of the dataset. Target variable *Number of Installs* has been transformed to class 0, class 1, class 2, class 3.

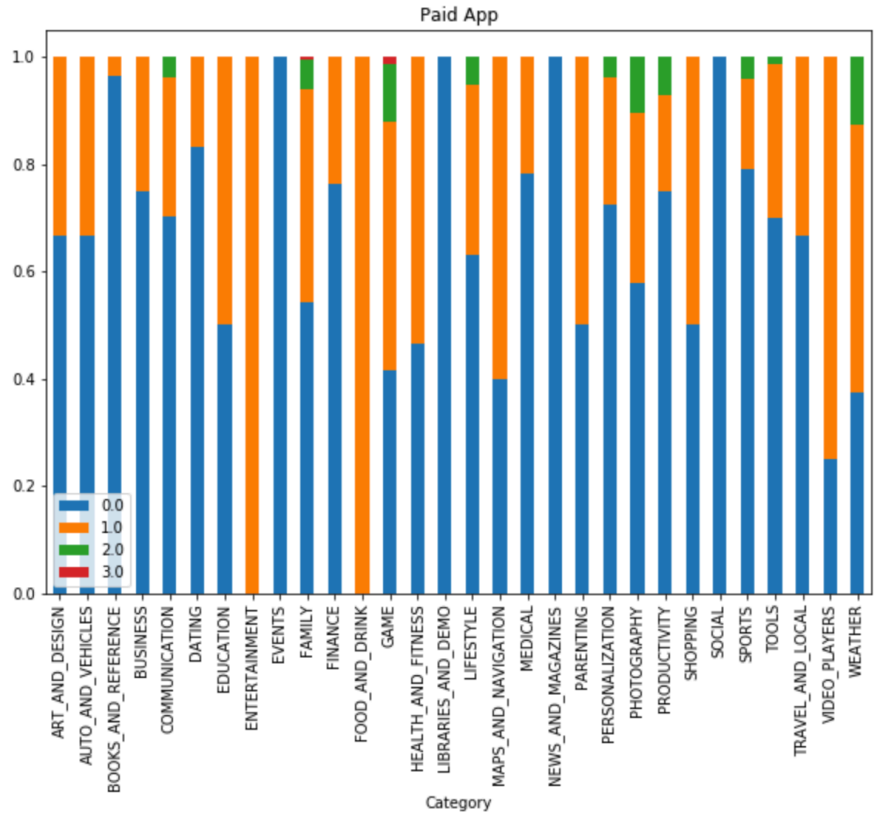


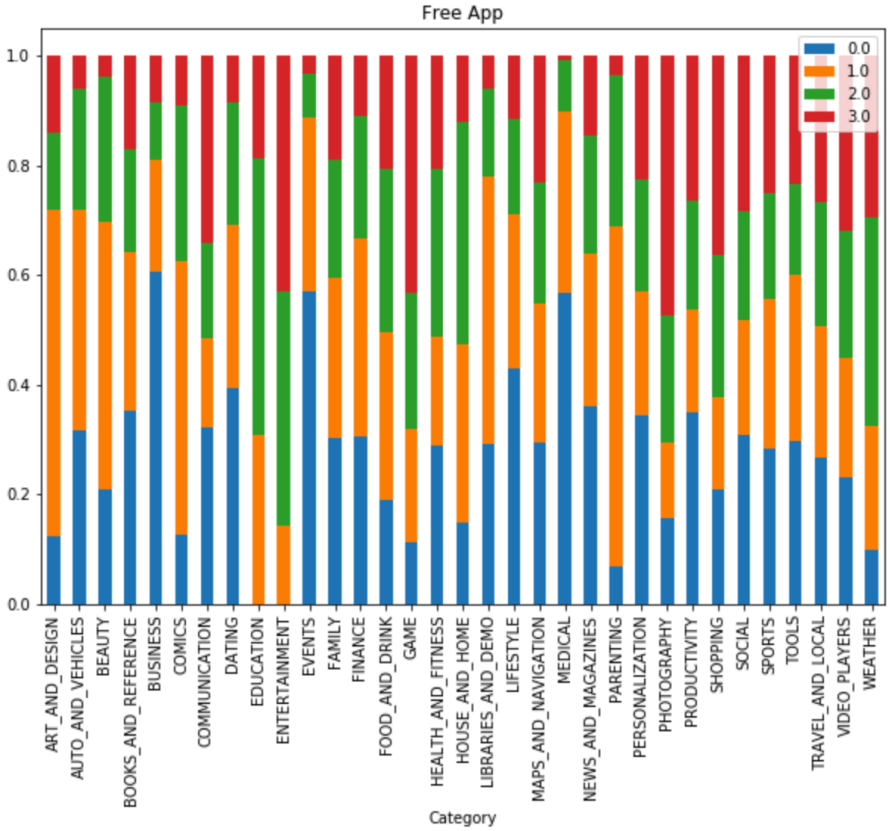
Based on my guess, free apps should be downloaded more often. From the stacked bar plots below, I do find that on average, green bars and red bars show up more often when the apps are free of charge.

Also, whether the app developer launch localized versions of their products with different file size considering the users' Android device does make a difference. More popular apps tend to have localized device-specific versions.

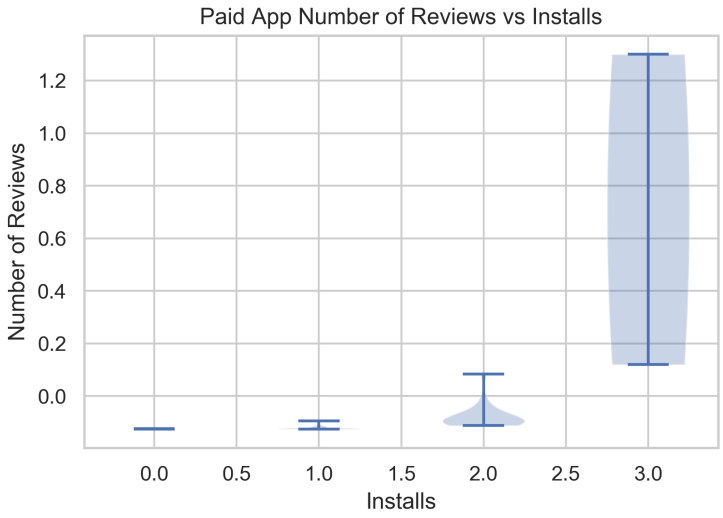
Furthermore, entertainment, food_and_drinks, video_players categories seem to be more popular (larger number of installs), compared to lifestyle and medical.

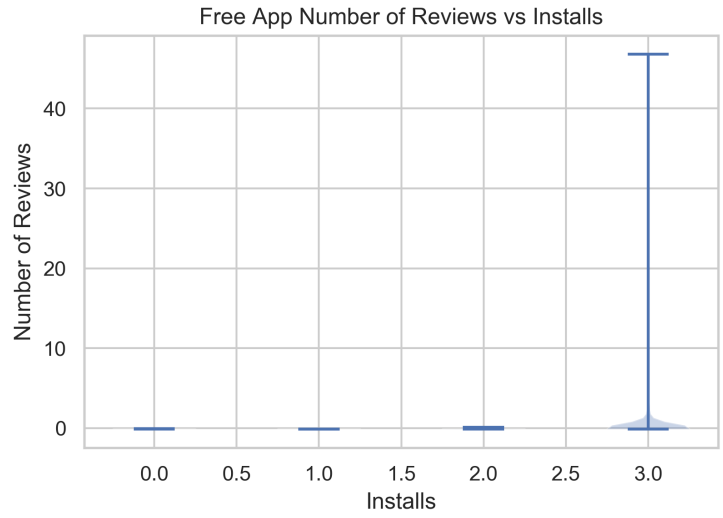



Paid App
1.0
0.8
0.6
0.4
0.2
0.0
0.0
1.0
2.0
3.0
ART_AND_DESIGN
AUTO_AND_VEHICLES
BOOKS_AND_REFERENCE
BUSINESS
COMMUNICATION
DATING
EDUCATION
ENTERTAINMENT
EVENTS
FAMILY
FINANCE
FOOD_AND_DRINK
GAME
HEALTH_AND_FITNESS
LIBRARIES_AND_DEMO
LIFESTYLE
MAPS_AND_NAVIGATION
MEDICAL
NEWS_AND_MAGAZINES
PARENTING
PERSONALIZATION
PHOTOGRAPHY
PRODUCTIVITY
SHOPPING
SOCIAL
SPORTS
TOOLS
TRAVEL_AND_LOCAL
VIDEO_PLAYERS
WEATHER
Category


Free App
1.0
0.8
0.6
0.4
0.2
0.0
0.0
1.0
2.0
3.0
ART_AND_DESIGN
AUTO_AND_VEHICLES
BEAUTY
BOOKS_AND_REFERENCE
BUSINESS
COMICS
COMMUNICATION
DATING
EDUCATION
ENTERTAINMENT
EVENTS
FAMILY
FINANCE
FOOD_AND_DRINK
GAME
HEALTH_AND_FITNESS
HOUSE_AND_HOME
LIBRARIES_AND_DEMO
LIFESTYLE
MAPS_AND_NAVIGATION
MEDICAL
NEWS_AND_MAGAZINES
PARENTING
PERSONALIZATION
PHOTOGRAPHY
PRODUCTIVITY
SHOPPING
SOCIAL
SPORTS
TOOLS
TRAVEL_AND_LOCAL
VIDEO_PLAYERS
WEATHER
Category

Also, it seems like the more popular apps have more reviews, and this is much more obvious with paid apps. My understanding is that when people spend money on an app, they are more likely to care more about it and leave reviews about it.

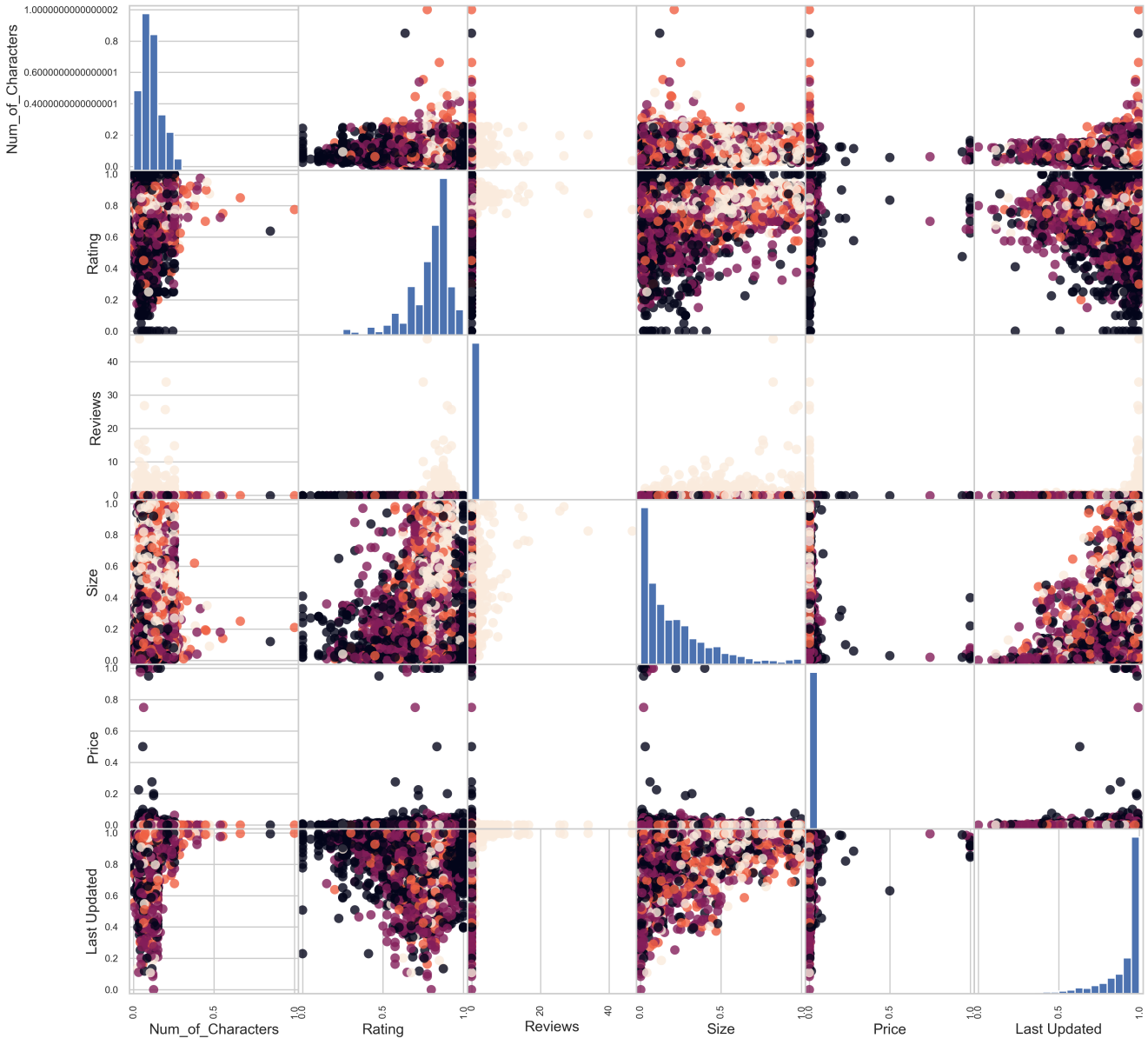




Now let's look at the scatter matrix.

Looking at the distributions of all the continuous features, the most prominent founding is that they are all screwed. The number of characters in the app's name is usually small, rating is usually high (if rated), most apps have few reviews, more apps have small file sizes, most apps are free or very cheap, most apps update frequently.

From the scatter plot, number of reviews is a strong indicator of whether the app has more downloads or not. Also, it seems like the higher the rating, the more popular the app is.



METHODS

DATA PREPROCESSING

1. Data Cleaning
    - drop duplicates
    - use RegEx to extract data and change data types to what they should be
    - group the values in *Number of installs* to <10k, <500k, <5 million, >5 million
    - join googleplaystore_user_reviews and googleplaystore on app name
2. Feature engineering
    - Make a column for number of characters in app name
3. Missing value
    - Drop a few rows where ***Android Ver***, ***Current Ver***, or ***Type*** is missing
    - Drop columns ***Translated_Review***, ***Sentiment***, ***Sentiment_Polarity***, ***Sentiment_Subjectivity*** because these columns have over 91% missing
    - Run MCAR test on ***Rating*** and Size (p = 0), need to impute them
    - For ***Size*** (continuous), when the value is "Varies with device", replace it with np.nan and treat it as missing, and create a new column called *Size_varies* which gives value 1 or 0, use iterative imputer to impute *Size* when needed
4. One-hot encoding: *Category, Genres, Current_Ver_truncated, Android_Ver_truncated*.
5. Ordinal encoding: *Number of installs, Type, Content Rating.* Type has values Free and Paid, so actually it does not matter. Content Rating is from general public to high restrictive.
6. Standard scaler: *Reviews, Size, Price, Last Updated, Num_of_Characters,* These values do not have an upper bound.
7. Minmax scaler: *Rating*, since it has value from 1 to 5.
8. After preprocessing, 9671 rows and 228 columns.

RANDOM FOREST

1. Parameter
    - max_depth: 30, 35, 40, 45, 50, 55, 60
    - min_samples_split: 2, 3, 4, 5
    - n_estimators: 100
    - random_state: given by function call
2. Metric
    - Baseline accuracy.
    - Accuracy score on test set. Since this is a classification problem and the data is relatively balanced, accuracy score can be used.
    - Standard deviation of accuracy score, since it measures splitting and non-deterministic uncertainties. The lower the standard deviation, the better (stable) the model.
    - Number of standard deviations above baseline.
    - Result: 0.844 +/- 0.005, 103.8 std above baseline

3. Pipeline
    - Split data in a stratified manner into other and test because the four classes take different percentages.
    - Split other into 5 stratified folds.
    - Impute the continuous features with missing values using iterative imputer. From MCAR test I know that they are not missing completely at random.
    - Process appropriate continuous features using standard scaler or minmax scaler (see above for reasoning), fit and transform only train set, transform CV and test sets.
    - Process appropriate categorical features (see above for reasoning) using one-hot encoder or ordinal encoder.
    - Use parameter grid to choose the best parameters and model.
    - Run it 8 times to compute standard deviation of accuracy score.

SVC

1. Parameter
    - C: 1e+3, 1e+4, 1e+5, 1e+6, 1e+7
    - gamma: 1e-4, 1e-1, 1e+2
2. Metric
    - Baseline accuracy, accuracy score on test set, standard deviation, number of standard deviations above baseline, same as above.
    - Result: 0.672 +/- 0.014, 24.8 std above baseline
3. Pipeline
    - Same as above except that I use PCA first to reduce the number of components to 25 before using SVC to reduce running time.

K-NEAREST NEIGHBORS

1. Parameter
    - n_neighbors: 20, 25, 30, 35, 40, 45
2. Metric
    - Baseline accuracy, accuracy score on test set, standard deviation, number of standard deviations above baseline, same as above.
    - Result: 0.501 +/- 0.012, 14.7 std above baseline
3. Pipeline
    - Same as above.

XGBOOST

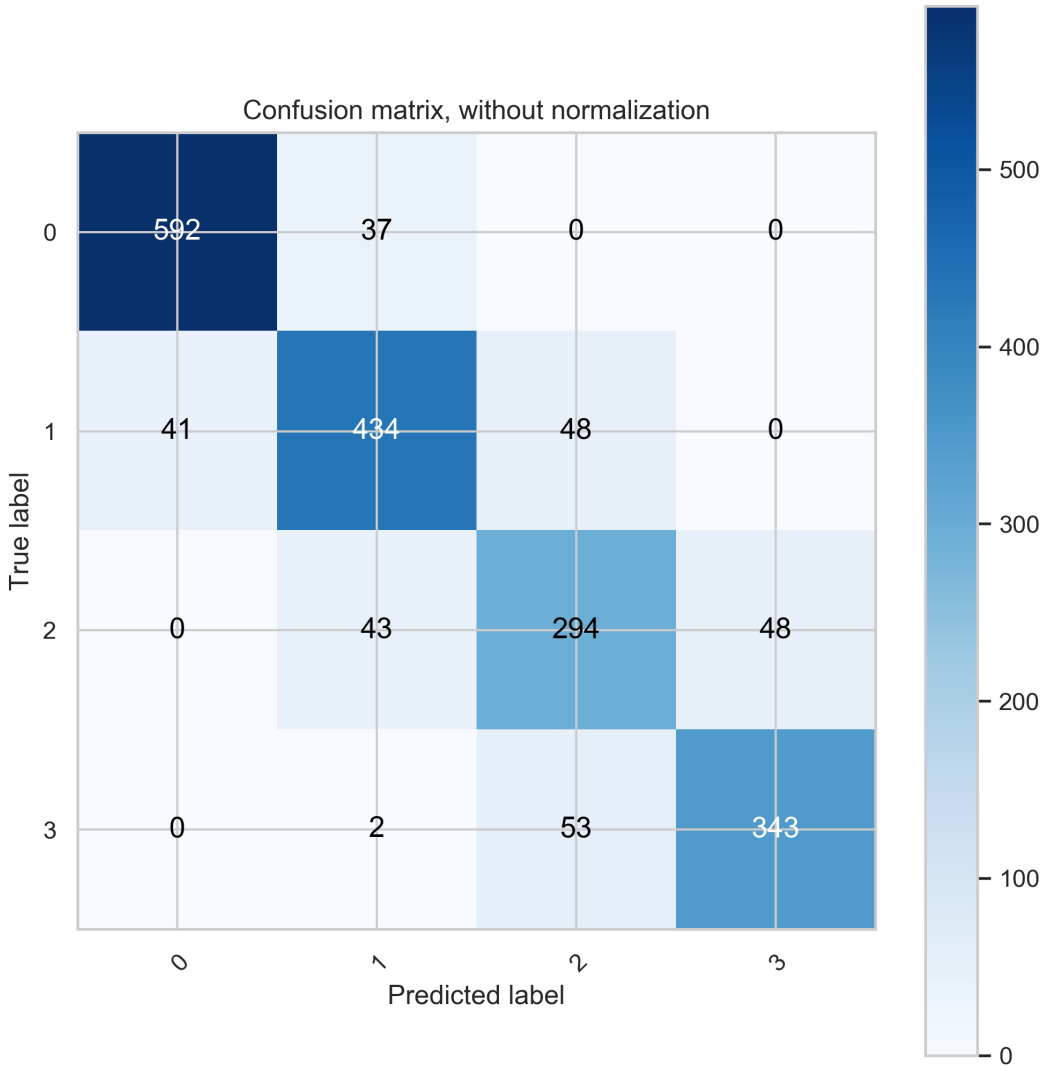
1. Parameter

    - colsample_bytree: 0.75
    - learning_rate: 0.03
    - max_depth: 20, 30, 40, 50
    - n_estimators: 100
    - subsample: 0.68
    - random_state: given at each run
2. Metric
    - Baseline accuracy, accuracy score on test set, standard deviation, number of standard above baseline, same as above.
    - Result: 0.858 +/- 0.002, 266.5 std above baseline
3. Pipeline
    - Same as above except that I did not impute missing values.

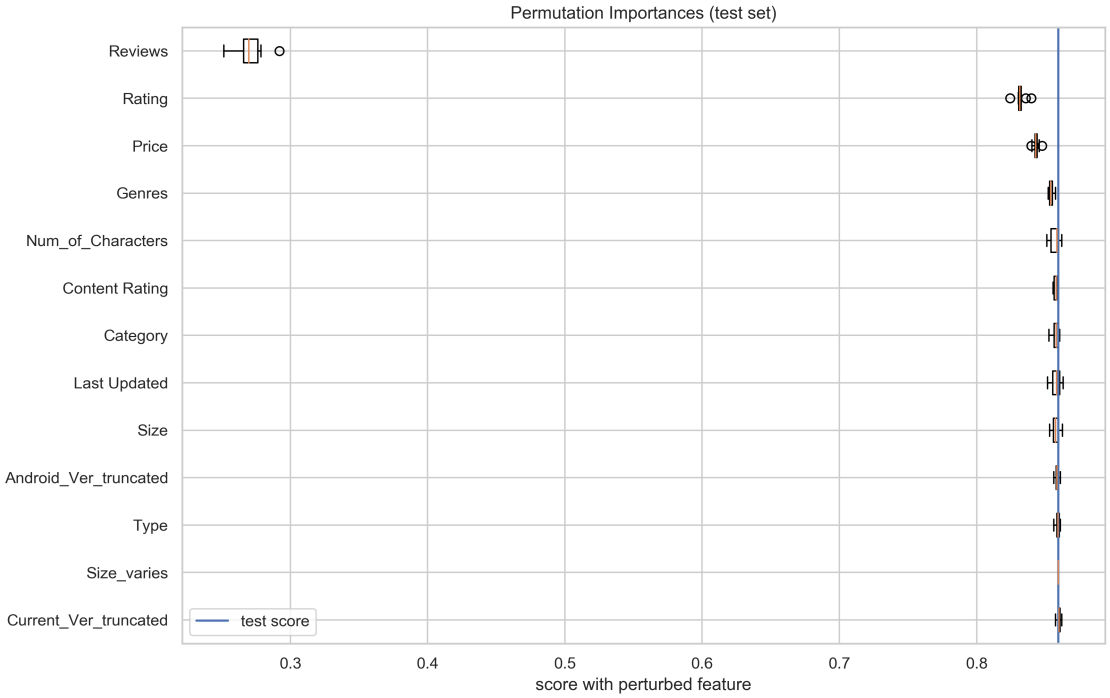
RESULTS

Baseline model accuracy: 32.5%, the percentage of class 0 (number of installs < 10k).

XGBoost has the best performance, with average accuracy score of 85.8% and a standard deviation of 0.2%, achieving 266 standard deviations above baseline. Below is the prediction result using one XGBoost model.



To break it down, the classification report shows that the f1 score for class 0 is the highest (93.8%), followed by class 3 (86.9%), class 1(83.5%), and class 2 (75.4%).



From permutation test result, Number of reviews, Rating, Price, and Genres are the four most important features. When Number of reviews is permuted, the model accuracy will decrease from 86% to less than 30%.

From the results above, when an app developer wants to make a blockbuster app, he wants to shoot for more reviews, higher ratings, and make it available for free. Also, he need to choose genres that usually attract more users than others such as entertainment, video games, food & drinks.

OUTLOOK

Interestingly, from the EDA I found that producing different versions of app catering to users' devices should make a difference, but the model does not support this finding. Since I treat *Current_Version* as a categorical feature, the reason might be that since there are many other values other than "Varies with device", the global feature important test does not capture the effect of this particular value on the target variable.

Also, the global feature plot above does not tell how does a particular category/genres impact number of installs. Since the column category and genres are highly correlated, I could have merged them into one feature, and try to use fewer categories in models.

Furthermore, *Number of reviews* is a feature that an app developer does not have good control of (unless they buy and fake it), so getting more reviews is not feasible advice.

Because of the time constraint, I could have tuned more parameters to improve accuracy, tried SVC models using other kernels, and try other models such as Naïve bayes classifier, logistic regression, and neural network.

Since column *Size* has a lot of missing values when "Varies with device" is treated as missing, I could try reduced features models to predict, or try to impute it n times and run XGBoost on them and compare the results to my current model (no impute).

For most apps, even when they have different versions for different devices, size is still available (as a range), model accuracy should improve if I can collect this information and compute the mean of this range.

REFERENCES

Kaggle public dataset: https://www.kaggle.com/lava18/google-play-store-apps

Sarang Narkhede. *Understanding Confusion Matrix 2018-05: Towards Data Science.* [Online: Medium] 2018. Available from: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Felix Revert. *Fine-tuning XGBoost in Python like a boss 2018-08: Towards Data Science.* [Online: Medium] 2018. Available from: https://towardsdatascience.com/fine-tuning-xgboost-in-python-like-a-boss-b4543ed8b1e