Midterm Project Report

Xiongfeng Wang

Introduction

This project will perform a supervised machine learning process on the Google play store Apps dataset[1] using rating as the target variable. This is a regression problem considering that the rating scales from 1 to 5, decile between integers.

This is an interesting task because it allows us to predict the rating of an App with certain features. For the existing Apps, comparing the prediction result with the true value can assist in finding outliers with specialty. For recently developed Apps, a precise prediction result reveals the feedback in the future which saves the value of time or gives people a hint about the App's customer experience.

The dataset initially has 10841 samples with 1 key (App), 1 target variable (Rating) and 11 features. For description and property, please refer to the table below.

There are plenty of public projects on this dataset, mostly focusing on EDA, data visualization, machine learning on rating prediction and data analysis. The top voted notebook[4] analyzed almost every column of the dataset except for the version ones and discussed some relationship between certain columns, for example, Users prefer to pay for apps that are light-weighted. The notebook's conclusion is neat but market-related practical. A trending notebook[5] plots the data with different plotting packages but just like most other visualizations, author discussed the relationship between rating and one feature individually. The top voted machine learning notebook[2] provides integer encoding RFR model, dummy encoding SVR model and random forest regressor model but cannot conclude which model has the best predictive.

EDA

Two rounds of EDA are applied. In the first round, I dropped completely duplicated rows with 10358 samples left following by samples with no target variable with 8893 samples left.

The Distribution of Category plot shows the that samples in Family and Game category is way higher than others, later in the Distribution of Category and Subcategory[3] we will find out the reason for this is that there are plenty of subcategories under these two groups.

One problem with the data set is that feature ‘Genres’ actually stands for the subcategory of the App if applicable. The Distribution of Category and Subcategory plot clearly reveals the component of the main category. As we can see for ‘Family’ and ‘Game’, there are more subcategories in it. The reason for this is that the database owner updated the database and the new data are categorized differently from the old data. There was no category ‘Family’ or ‘Game’. An app used to have

category ‘Action’ now have category ‘Game’ and genres ‘Action.’ Therefore, I will use genres but not category feature for the group structure as it contains more information. Another reason is that genres help us identify ‘almost duplicate’ samples in the second round EDA.
Once we figure out which feature to use for the group structure, we can go solve the second the problem: we have ‘almost duplicate’ samples which are identical except for the nuance in the reviews feature. In the second round of EDA, I removed them by only keep the sample with highest number of reviews grouping by app name and genres. Genres is necessary because same App can have totally different data if they have different genres, meaning the data come in different updates, see ‘Bubble Shooter’. Name of the App is case sensitive. For example, ‘Blood Pressure’ and ‘Blood pressure’ are two Apps with totally different data. We have 8211 samples left.
The Scatter Matrix for float & int variables plot shows the relationship between variables with the type of float & int. If we look at the histogram for Distribution of Size, Price and Days (see figure repo) we will find out the variable is quite imbalance, x-axis must be in semi-log to show a fair plot. This explains why some of the matrix plot has heavy skewness. This suggests that we should apply StandardScalerEncoding later.
I use bin = 39 for the Rating over Category graph because of the rating scale I mentioned above, we noticed that Family and Gaming seems to have a higher average score than the overall rating.

Methods

Splitting Strategy

This is not an IID dataset because clearly there are group structures in the dataset: Category and Genres. Also, this is not a time series dataset because even though for very small number of Apps, there are samples for different version of it and the rating might vary, the proportion is too small to be consider as a general feature. Most of the Apps with different version only differ in number of reviews by nuance.
Considering there are 8211 samples with a valid target features, the size of the data set is fair to apply an 8-1-1 split. Category with lowest counts is 42 for Beauty, 9 categories' count below 100, and only 2 categories' count above 800, so the 8-1-1 split is rational because we don't need to worry about group in train/test set should not appear in the other. We can implement plenty of n_split on the dataset.
Since there's an obvious group structure, I will choose GroupShuffleSplit to split the dataset. I choose GroupShuffleSplit over GroupKFold because the data set is also imbalance as we can see from the distribution of categories or genres.

Preprocessing

For Encoder selection and reason, please refer to the table below.

	Variable	Description		Encoding	Note on reasons
Key	App	Name of the App	Categorical	No Encoding	
Feature	Category	Category of the App	Categorical	OneHotEncoder	No order
	Reviews	Number of reviews	Numerical	StandardScaler	Skewness
	Size	Size of the App	Numerical	StandardScaler	Skewness
	Installs	Scale for number of installs	Categorical	OrdinalEncoder	With order
	Type	Free or Paid	Categorical	OneHotEncoder	No order
	Price	Price of the App	Numerical	StandardScaler	
	Content Rating	Restriction rating	Categorical	OneHotEncoder	No order
	Genres	Category; Subcategory	Categorical	OneHotEncoder	No order
	Last Updated	Last updated date in Month DD, YYYY	Numerical	N/A (see Days)	
	Current Ver	Current version of the App	Categorical	N/A	
	Android Ver	Android version that supports	Categorical	OneHotEncoder	No order
	Subcategory	Subcategory	Categorical	N/A	
	Days	Day since last update	Numerical	StandardScaler	Skewness
Target	Rating	Rating for the App	Numerical	No Encoding	

Missing Value

There are 4 missing values for categorical feature ‘Current Ver’ and they are turned in to ‘None’. 1171 missing values for numerical feature ‘Size’ take more efforts to handle. XGBoost puts the missing value to either left or right with lower impurity. Unfortunately, XGB is only for gradient boosted trees but not regression or SVM. The easiest way is to drop rows with missing size which is about 14% the data we have. However, as we can see from figure 3, feature ‘Days’ seems to have a fair correlation with ‘Size’, which means ‘Size’ is MAR (Missing conditionally at random). So, I apply iterative imputation with ‘Size’ and ‘Days’.

Evaluation Metric

MSE/RMSE is my choice of evaluation metric. This is a regression problem so basically, I'm choosing between MSE and R^2. MSE is the average of the square of errors while R^2 is total variance explained by model over total variance. Two metrics are correlated but I choose MSE because it reflects the model's prediction power in a straighter forward way.

Model Selection

The general idea is to train as much models as possible. So I trained 6 most commonly used models: Lasso, Ridge, Elastic net, Random forest regressor (RFR), Support machine regressor (SVR) and KNeighbor regressor (KNR). Please refer to the table below for hyperparameters tuning and corresponding values I tried. Alpha in Lasso, Ridge and Elastic net tends to be really small to get the rid of the effect on penalty (weaker regularization). Ratio in Elastic net is 0.1. One thing really interesting is that the average test score for ElasticNet and Lasso are almost the same while the test score for best model in each random states are rather different. For RFR I first tune the hypermeter with the following values and the best value is max_depth = 1 which suggest the model tend to be very easy: using only one decision node is somehow underfitting. Besides in the previous example, RFR is the model with the highest test score while here it has the lowest one. So I tied set the max_depth to default and tune max_features only. The result is slightly worse than the previous one. The value for hyperparameters of SVM and KNR are pretty fixed for different random states too. In figure 5 we can see KNR with n_neighbors = 10 generates the best model. I bet the model would take it if I allow even smaller n_neighbors.

Model	Hyperparameter(s)	Values to try	
Lasso	alpha	[1e-15, 1e-10, 1e-5]	
Ridge	alpha	[1e-15, 1e-10, 1e-5]	
Elastic net	alpha; l1_ratio	[1e-10, 1e-6, 1e-3, 1e-1, 1e0, 1e1]	[0.1, 0.3, 0.5, 0.7, 0.9]
RFR	max_features; max_depth	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6]	[1, 3, 5, 10, 30, 50, 100]
SVM	gamma; C	np.logspace(-5, 5, 11) :1e-05	np.logspace(-5, 5, 11) :1e-05
KNR	n_neighbors; weights	np.linspace(10, 200, 20, dtype = int)	['uniform', 'distance']

Result

The result for each model is shown in figure 5. Random forest regression is the worst one and KNeighbor regression is the best one. I'm glad the overall score beats the one in the previous example. All models show that a good model here is tend to be simple, maybe because the dataset is diverse. And this indicates that the global importance of the feature is very close to the correlation of the feature as we see during the EDA, also shown in the scatter matrix

Outlook

Same App can have totally different data if they have different genres. Besides, some Apps change their names as they upgrade like ‘Basket Manager 2016 Free’, ‘Basket Manager 2017 Free’ and ‘Basket Manager 2018 Free’. Time does come in play in these cases, but these cases are rare. Here we treat them as independent samples with different genres just like the rest of the samples. By treating them as independent samples might loss the information which some samples, even just a small portion, are correlated. To enhance we can setup a weight for samples of the same App.

Figures

(See figures repo for more figures)

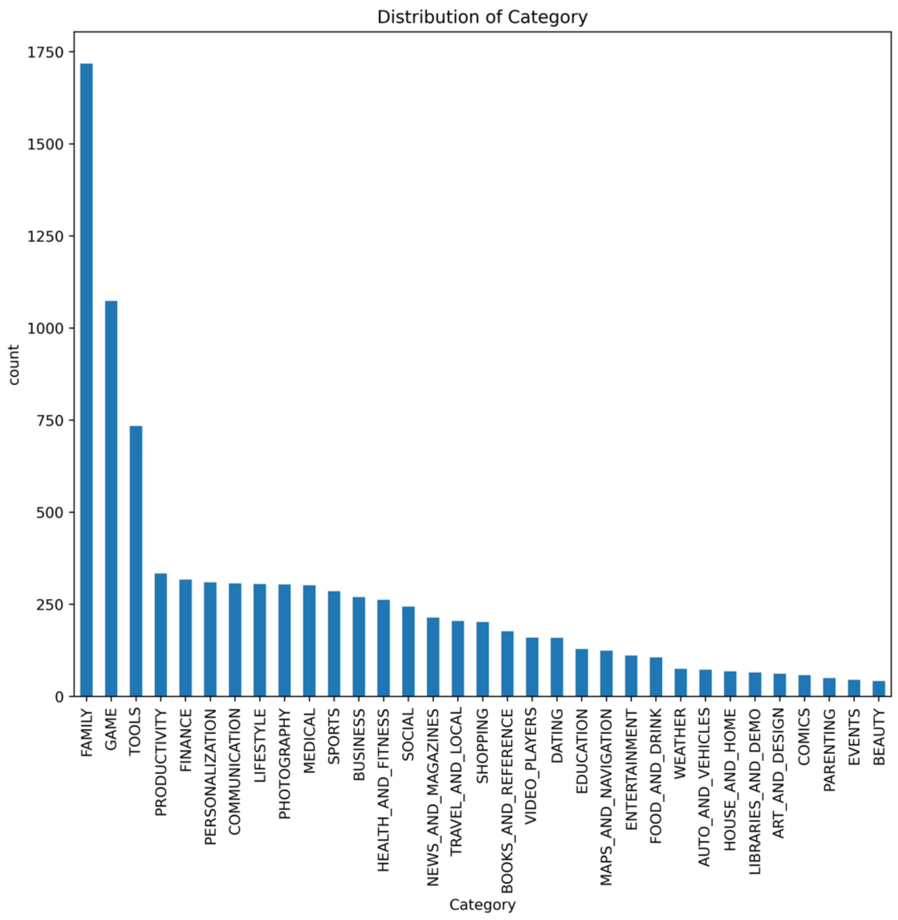

Figure 1: Distribution of Category in first round EDA

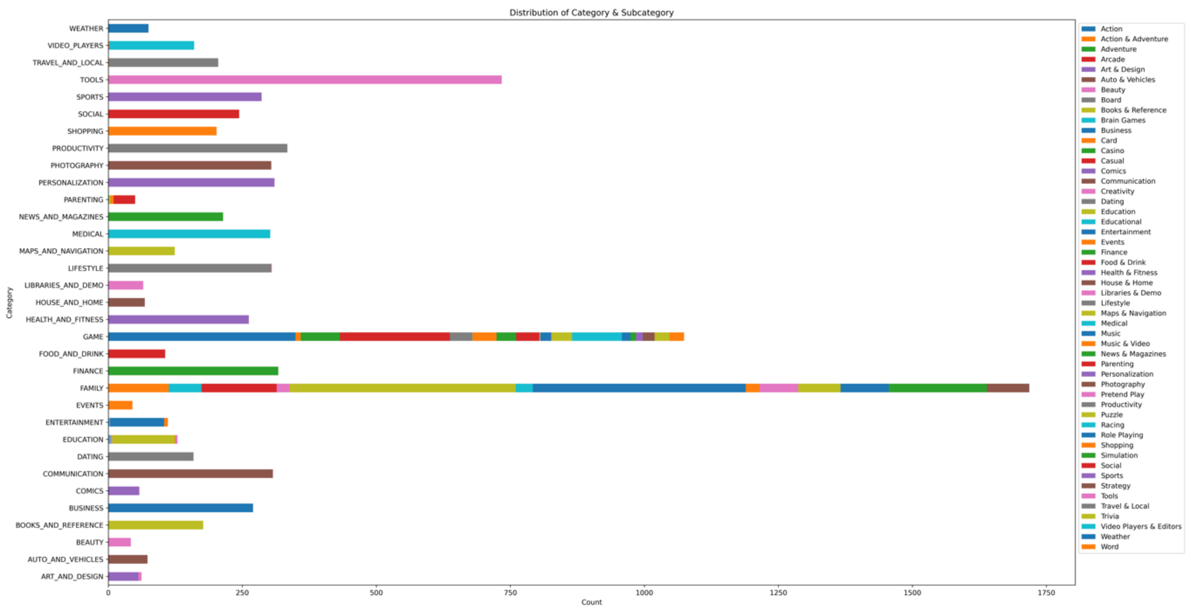

Figure 2: Distribution of Category & Subcategory in first round EDA

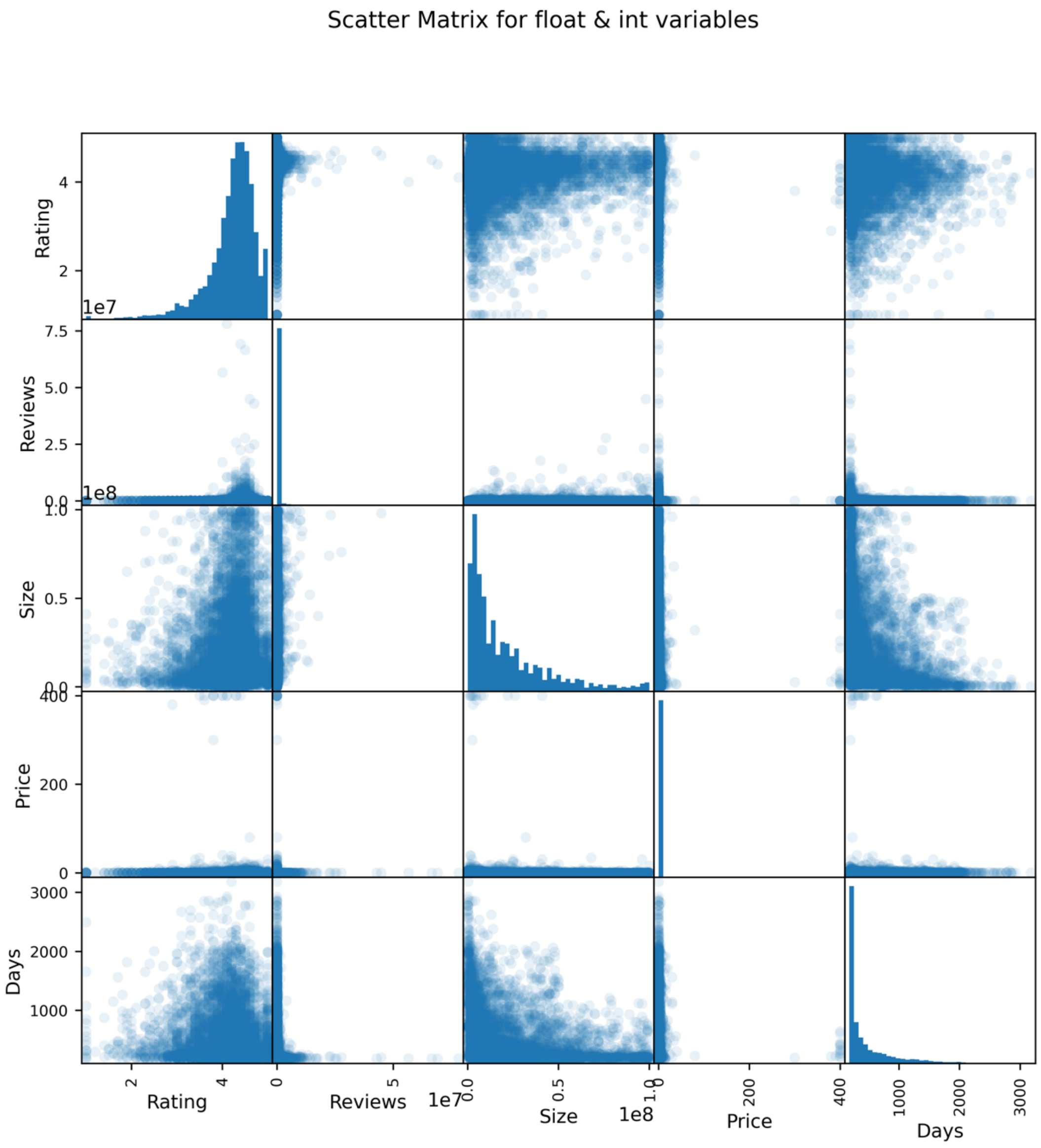

Figure 3: Distribution of Category & Subcategory in second round EDA

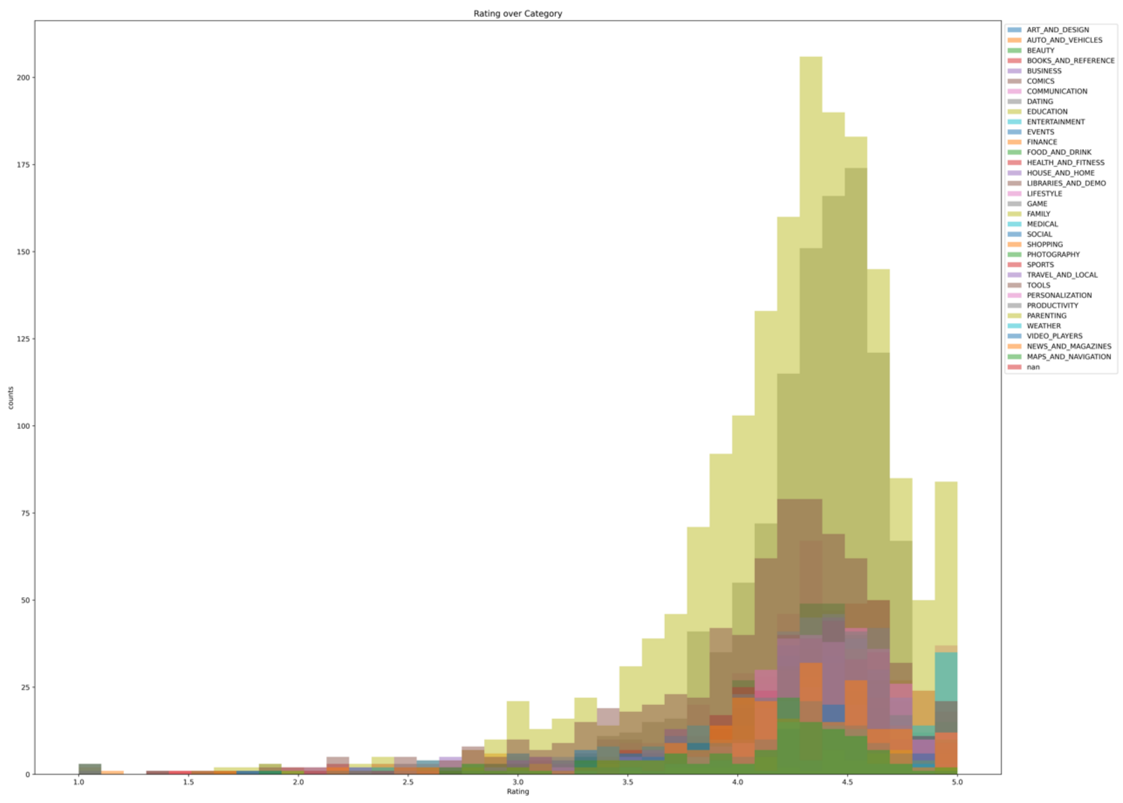

Figure 4: Rating over Category in second round EDA

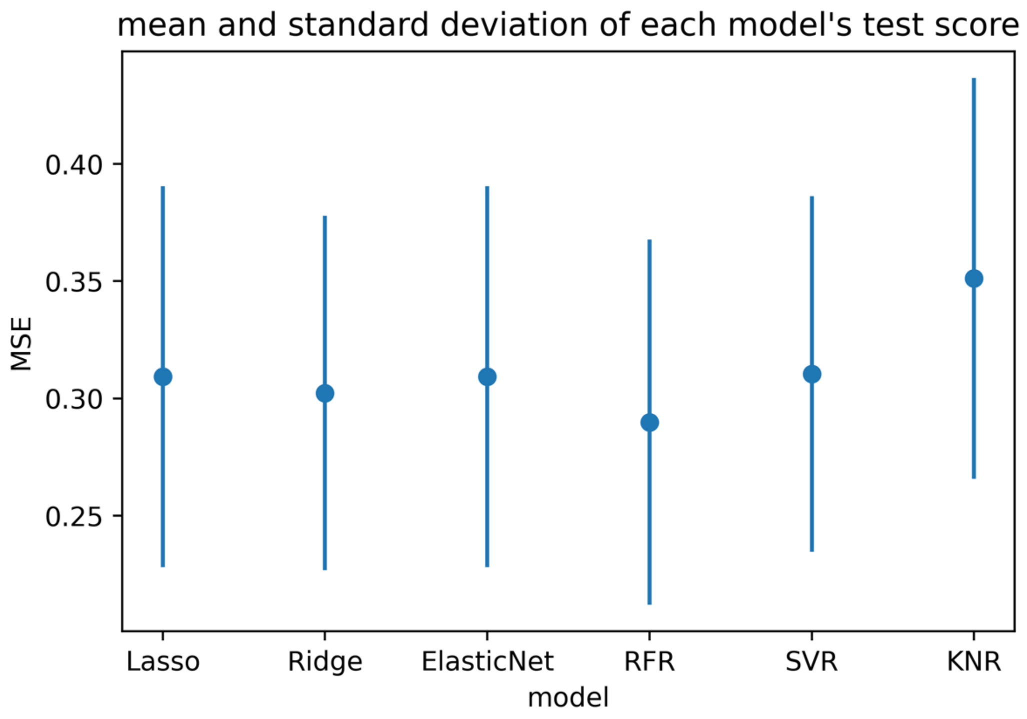

Figure 5: mean and standard deviation of each model's MSE

GitHub Repository

https://github.com/XiongfengWang/1030project

Reference

1. Gupta, L. (2019, February 03). Google Play Store Apps Dataset. Retrieved October 13, 2020, from https://www.kaggle.com/lava18/google-play-store-apps
2. Jemseow, J. (2018, September 27). Machine Learning to predict app ratings. Retrieved October 13, 2020, from https://www.kaggle.com/jemseow/machine-learning-to-predict-app-ratings
3. Siddiqi, S. (2018, September 28). Google Play Store Apps - Data Cleaning. Retrieved October 13, 2020, from https://www.kaggle.com/sabasiddiqi/google-play-store-apps-data-cleaning
4. Gupta, L. (2018, September 19). All that you need to know about the Android market. Retrieved October 13, 2020, from https://www.kaggle.com/lava18/all-that-you-need-to-know-about-the-android-market
5. Jha, R. K. (2020, January 27). ML to Visualization &amp; Prediction of App Ratings. Retrieved October 13, 2020, from https://www.kaggle.com/rajeshjnv/ml-to-visualization-prediction-of-app-ratings