Final Report:
Ed-Tech Apps Ratings Analysis

Problem Statement

Educational tech apps are committed to the development of a new way of generating and delivering knowledge to motivate learning. They have been widely used in a variety of ways, including online learning, interactive whiteboards, and other forms of digital learning. EdTech can also be used to create virtual classrooms, where students can interact with each other and their teachers in real time. EdTech is becoming increasingly popular in schools and universities, as it allows for more personalized instruction and can be used to help students with a variety of learning styles.

EdTech is arguably to have the following benefits: enhance parent/caregiver communication, maintain updated information, motivate kids learning, individualize learning, increase test scores, improve comfort and skill with technology, make progression private. For these reasons, efforts should be made to integrate m-learning into development of more effective and efficient educational apps, to complement traditional learning channels.

Data Wrangling

The data is part of the data that were collected in June 2021, by Gautham Prakash And Jithin Koshy (Kaggle: https://www.kaggle.com/datasets/gauthamp10/google-playstore-apps
Backup repo: https://github.com/gauthamp10/Google-Playstore-Dataset)

The raw dataset contained 262,398 Educational Apps with 23 columns. 478 duplicated records were dropped. Apps with no rating or ratings as zero were dropped.
1) Among 23 columns, features that would offer little or no information in determining App ratings, such as Developer ID, Developer Email, Privacy Policy, Minimum Installations, Maximum Installations, etc. were dropped completely. The features left in analysis with missing info are displayed in Figure 1.
1) Features “Installs”, “Size” all have missing values less than 0.01% ,(therefore almost invisible in missing-value plot below), and rows with missing values in either one of these features were dropped.
2) For rows with missing in “Released” but non-missing in Minimum Android version , “Released” were imputed by the mean of all available Released dates of the same Minimum Android version. Rows with missing in both were dropped completely. “Minimum Android” was then dropped after this step.
3) Nearly all Apps that are not free are priced in US Dollars, with the exception of 35 rows. 34 Apps are missing in Currency and 1 App has “SGD” as Currency. These 35 rows were excluded from classification.
4) Feature “Free” (Yes/No) was dropped and Used “Price” instead, with Price=0 being Free.

At the end of data cleaning step, we ended with a dataset of 140,424 rows and 12 columns. Information of all raw features are given in Table 1.

Pre-Processing

Some of the features in Table 1 are entered as they are and some are re-valued. There are also calculated new features, as summarized in Table 2. In aprticular,
1) Continuous features such as Installations, Size, Price are grouped into ordinal categories.
2) Content Rating is regrouped into fewer nominal categories, down from 4 categories to 3, namely ”Everyone”, “Adult Only” and "Teen Only"
3) Age, Updated_age are newly calculated continuous features that contain information on Apps’ history.
4) Continuous raw rating scores were first scaled by the logarithmic values of their rating counts, and these scaled values were then further categorized and mapped back into the standard 5 point likert scale, 1-5. Distributions of raw ratings and transformed ratings are shown in Figure1.
5) Features such as Editors Choice ,In-App Purchase , Ad supported, are boolean and are entered as they are.

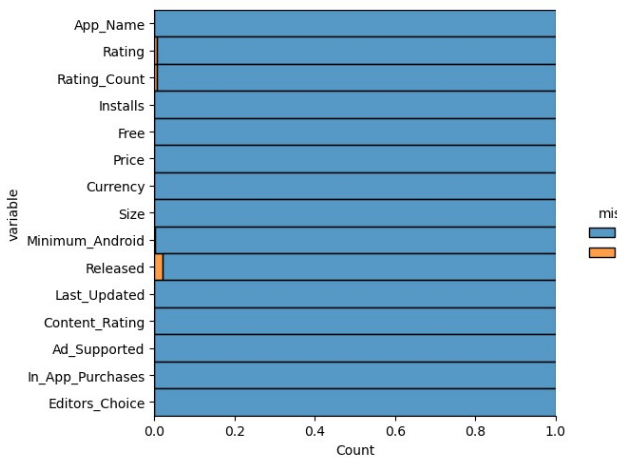

Figure 1. Missing values Distribution by Features

Features	Definition
App_Name	App name
Rating	App rating score
Rating_Count	The counts of ratings
Installs	The number of installations
Price	The price of App
Released	Time when first released
Size	Size of App in gig-byte
Last_Updated	Time of the latest update
Content_Rating	User Groups permitted to rate App
Ad_Supported	Is the App ads free
In_App_Purchases	Is there in_app purchase
Editors_Choice	Is the App rated as Editors Choice

Tab1e 1. Raw Features Information

Features	Old Values	New Values
Install	Continuous	"under 1,000", "1,000 to 10,000", "10,000 to 100,000", "above 100,000"
Content_Rating	“Everyone”, “Unrated” “Mature 17+”, “Adults only 18+” "Teen"	} ”Everyone” } “Adult Only” "Teen Only"
Size	In thousands In Millons ‘varies with device”	“K” “M” “V”
Price	Continuous	“Free” “<= $10”, “> $10”
Age*		Time in Years from “Released” to Jun. 15th 2021** Standardized by StandardScaler()
Updated_Age*		Time in Years from “Last_Updated” to Jun. 15th 2021** Standardized by StandardScaler()
Rating_Scaled_cat*		1) Rating_Scaled= Rating/*log* (Rating_Counts) 2) Re-map Rating_Scaled into standard rating scales of 1-5
Editors_Choice	Yes/No	Yes/No
In_App_Purchase	Yes/No	Yes/No
Ads_Supported	Yes/No	Yes/No

* New Calculated Features
* * Jun 15th 2021 is when the data was scraped

Tabel 2. Feature Entered into Modeling

Explanatory Data Analysis

First I examined the distribution of each feature, and compared the target value distributions of before and after transformation (Figure 2). There is class imbalance in the calculated target, with rating "2" accounting for ~82% , rating "3" accounting for ~11.5%, rating "1" accounting for ~5.8% , and rating "4" and "5" together accounting for <1%. Correlation among features was examined and presented in a correlation heatmap, where only correlation strength greater than 0.2 is annotated. As we can see in the map that

1) There is a weak correlation between rating scores and installations, more frequently installed Apps get higher rating score, which can also be seen in Figure 5.

2) Features that can promote installations are: having the in-App purchase feature, supporting ads,and having longer history (larger Age) on Google Play store. However, each of these feature by itself does not show effect on App's rating score. This observation is backed up by the correlation coefficients in corelation Heatmap.

3) Even though more installations is associated with higher rating counts, more ratings does not necessarily bring higher rating scores.

4) Being on the Editor's Choice list is positively related to higher rating counts, which consequentially entails an indirect positive relationship of Editor's Choice with Rating scores, as shown in Figure 4.

5) By itself the age of an App seems unrelated to its rating score. However, after controlling for installations, older Apps are shown to have higher rating scores, which is especially true for Apps with installations above 100,000. This might can be explained by the fact that "old age" means a longer life, a longer period of being availability and more time for adoption.

6) Older Apps are updated less frequently and have longer duration since last updates.

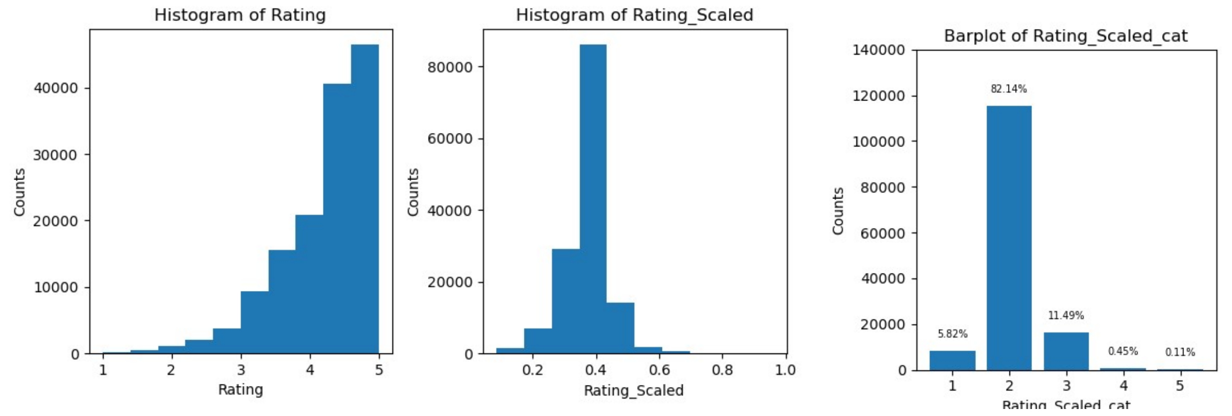

Figure 2: Distribution of Rating, Scaled Rating and Scaled-Categorized Ratings

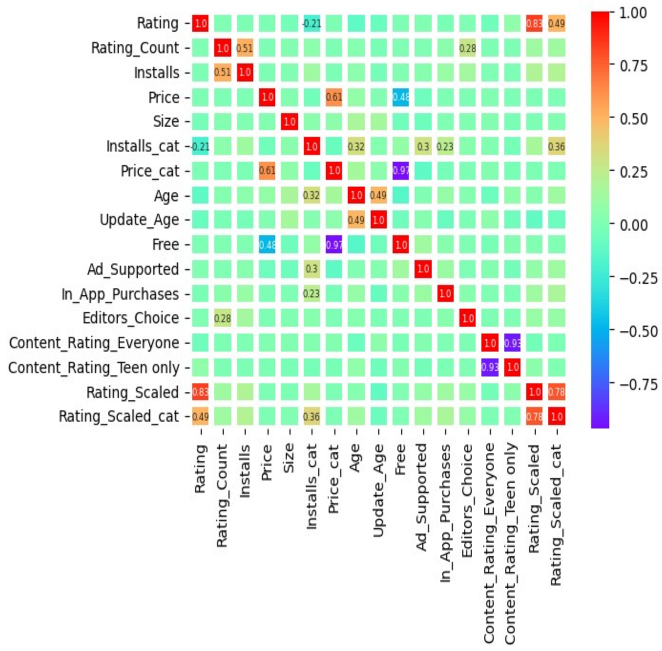
Figure 3: Correlation Heatmap

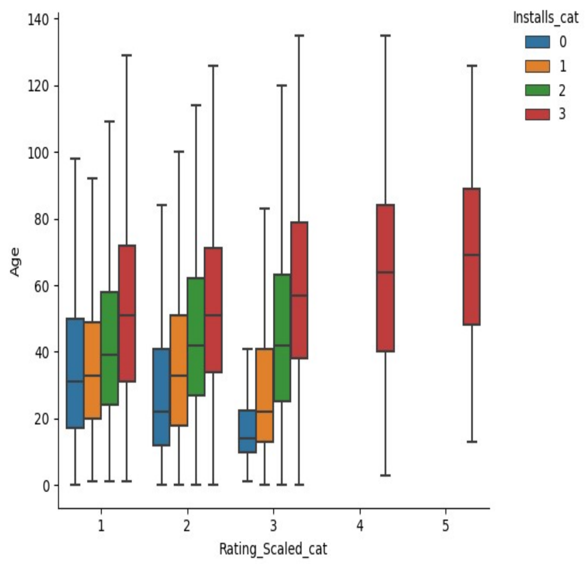
Figure 4: Boxplots of Age by Rating & Installation

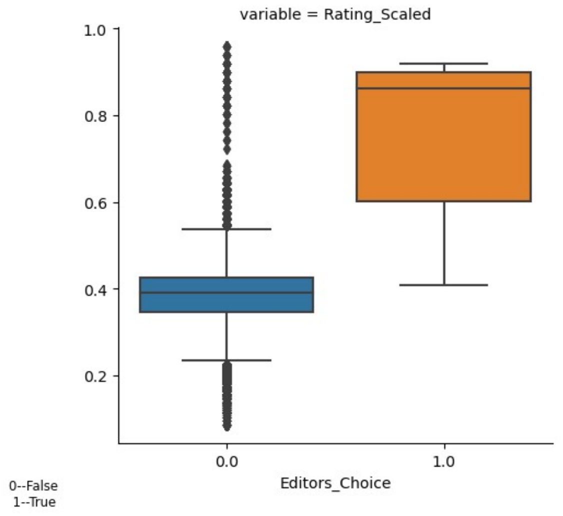
Figure 5: Scaled Rating by Editor Choice

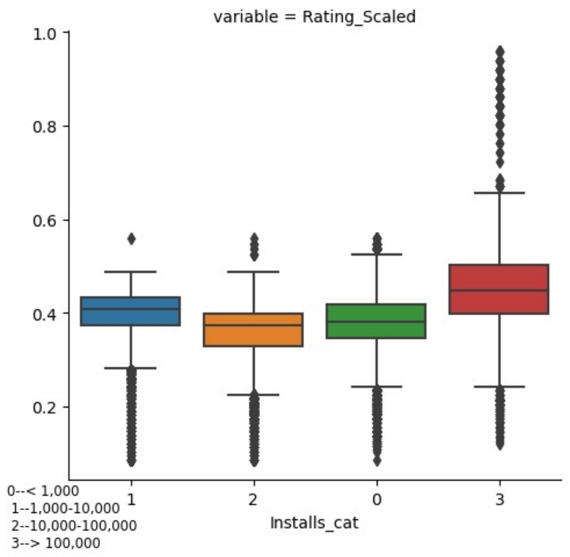
Figure 6: Scaled Rating by Installation Categories

Modeling

At modelling step, train test datasets are are split in a stratified fashion at a ratio of 9:1. There are 123,381 rows in training set and 14,043 in testing set. I first tried conducting multi-class classification on the calculated ratings (a ordinal categorical feature on the 5-point likert scale) . However, after various trials with different combinations of over-sampling and under-sampling techniques, with robust classifiers such as DecisionTreeClassfier, RandomForestClassfiier, GradientBoostingClassifier, and with hyper-parameter tuning, classification results were not desirable. The hyper-parameter tuning was done by RandomizedSearchCV with 5-fold cross-validation and 50 iterations. The performance metric macro-f1 scores stayed at around 0.30, and the accuracy scores stayed at around 0.80. The f1-scores for each classes, at their best, achieved the following values: (Class “1”~0.05, Class “2”~0.90, Class “3”~0.38, Class “4”~0.10, Class “5”~0.10). it seemed that the severe class imbalance problem could not simply be resolved by resampling, choice of classifier and/or hyper-parameter tuning.

To combat the severe class imbalance, I then decided to combine rating scores of “1”, “2” “3” into one class labeled as “Low”, and rating scores of “4”, “5” into another class labeled as “High”. After regrouping, I ended up having 123,507 (~88%) in Class “Low” and 16,917 (~12%) in Class “High”. There is still class imbalance but not as severe. The analysis was then proceeded as a binary classification. Combinations of over-sampling and under-sampling techniques once again were experimented and tuned for performance. The same group of classifiers (DecisionTreeClassfier, RandomForestClassfiier, GradientBoostingClassifier) were employed, hyper-parameters were tuned, and model performance metrics (a combination of the f1 scores for each class, the macro-f1 scores, the overall accuracy scores) were compared. The winner is GradientBoostingClassifier with n-estimators=100, learning_rate=0.1. Here is summary of findings , reported in Table 3 as well.
1) Over all, with binary classes the classification results have improved;
2) The macro-f1 scores is boosted up to 0.63, and the accuracy scores goes up to 0.74. The f1-scores is ~0.94 for class “Low” and ~0.53 for class “High”;
3) Installation is leading high in feature importance ranking, and its relative importance score (0.75) is way bigger than the rest. The next three are in-App purchase (0.09), the age of App (0.06) and updated age (0.06). All other are either close to or below 0.01.

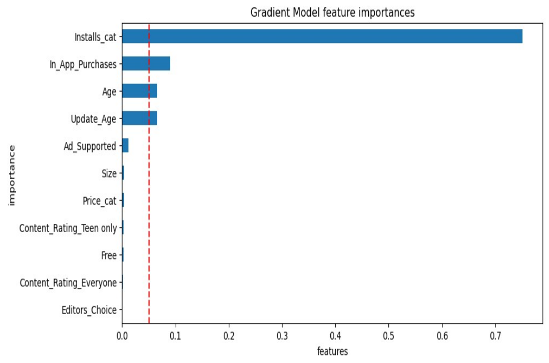

Figure 7: Feature Importance Plot

Class	precision	recall	f1-score
Low	0.93	0.94	0.94
High	0.54	0.51	0.52
macro avg	0.74	0.73	0.73
accuracy			0.89
weighted avg	0.89	0.89	0.89

Table 3: Final Model Classification Report

Conclusion

My study shows that given the Installations, Size, Price, Content Rating, information such as if the App supports ads, if it includes in-App purchases, if it is on the Editor’s Choice list, ect, the model could at best predict the App rating with only moderate accuracy. Though, class imbalance can not take all the blame for the undesirable classification results. There are a couple of ways to proceed for improvement

For one thing, I would look into Feature engineering. There might be important features about Apps that are missing and need to be incorporated. Another thing is that the target feature ratings are sentimental, subjective and are difficult to measure and it might not the best metric to measure App’s quality. We should look into alternative target features. This alternative target feature would have stronger correlations with other App features and can be modeled with better prediction/classification performance. For example, Installation might be a good alternative, given that it has stronger correlations with other features

(In_App_Purchase, Ad_Supported, Age, Updated_App, as can be seen in the correlation heatmap in Figure3.

One important conclusion we can draw is that Apps having in-App purchase feature and Apps supporting Ads have large Installations. App Developer can benefit from this finding and boost Apps' adoptions.

We could also investigate other genres of Apps, to see if our classification model would work differently across different genres. Id Educational App really rated differently compared with other genres?