Project Proposal

Jiamin Tang

Problem Description

As app development is an increasingly innovative industry nowadays, ratings in this competitive market has become a significant index to measure the success of an app. It not only has the potential to give developers a hint how to make a more popular product, but also become a determinant for users' purchase decisions. High-rating apps are more likely to be attractive to users, thus can drive business success in app-developing business. Besides, the vast availability of the features of existing apps published by app stores provides the possibility to predict apps' ratings.

In this case, we mainly focus on predicting the rating of an app in Android market based on the collected features of published apps. The target variable will be the app ratings which is an average of user ratings out of 5 stars. Also, this is a problem of regression.

Dataset Description

The dataset for Google play Store Apps is the web scraped data of about 10k Play Store apps for analyzing the Android market. There are two pieces of dataset, one of them is the objective statistics of the app including size, installs etc., another one is user reviews containing the most relevant 100 reviews of each app and attributed with some features. We suppose the first dataset will be of more contributions, so we mainly focus on the objective statistics to excavate high-rating features.

The dataset contains 10840 entries and 11 features with each app, accompanied with the target variable. The description of each feature is given as follows with the preprocessing steps.

Data Preprocessing

At first, we noticed that there are some missing entries in some features and we check the fraction of missing values per column. We can see that there are missing entries in Rating, Type, Content Rating, Current Ver and Android Ver. The overall percentage of missing values is about 13%, which is mostly due to the missing ratings. We try to fill in the feature of Types since it can be implied from Price, and we treat the missing in categorical feature Content Rating as another category. For the rest of features with missing entries, we will just remove the NaN entries since we have no such simple way to recover them that can remove bias. Finally we have a dataset with 9360 rows and 13 columns with no missing entries.

For each feature, we consider different transformations as well as some preprocessings before it can be applied with transformers.

- App: Application name
  We drop the column App in the preprocessed dataset since we cannot imply anything predictive from the app names.

- Category: Category the app belongs to
  We apply LabelEncoder with integer values from 0 to n_classes-1 since the categories are unordered and there is no need applying OneHotEncoder with no intersections.

- Rating: Overall user rating of the app out of 5 stars(as when scraped)
  We retain Rating in the preprocessed dataset as original since it’s the target variable.

- Reviews: Number of user reviews for the app (as when scraped)
  For the feature Reviews, we can see the type of the feature is object. We should first convert it into numeric, and then apply StandardScaler because it varies a lot and has outlier cases.

- Size: Size of the app (as when scraped)
  For the feature Size, except for "Varies with device", the rest either contain the unit "M" or "k". For the thing "Varies with device", we try to add a categorical feature called "size varies by device" (1 if it does, 0 if it doesn't). And for the rest, we replace the unit "k" and "M" respectively by "e+3" and "e+6", and then convert to numeric. After that, for the continuous feature, we apply StandardScaler.

- Installs: Number of user downloads/installs for the app (as when scraped)
  For the feature Installs, since it contains "," and "+" in the string and is hard to deal with, we consider to treat the 19 unique values in it as categorial and apply OrdinalEncoder to it.

- Type: Paid or Free
  We apply OrdinalEncoder to the feature Type according to whether it is paid.

- Price: Price of the app (as when scraped)
  For the feature Price, "$" should be removed and then convert to numeric. We apply MinMaxEncoder to it.

- Content Rating: Age group the app is targeted at -- Children / Mature 21+ / Adult
  We apply OrdinalEncoder to the feature since it can be arranged in the age order.

- Genres: An app can belong to multiple genres (apart from its main category). e.g. a musical family game will belong to Music, Game, Family genres
  For the feature Genres, we can see there are some transactions. The thing we should do first is to split the categories in each genre, and we have to genres called "First Genre" and "Second Genre", which are the same when the feature only belongs to one genre. Then we apply OneHotEncoder to the two new columns.

- Last Updated: Date when the app was last updated on Play Store (as when scraped)
  The feature appears the exact date like "January 7, 2018", which is not informative. It is a good way to convert it to the number of days till now with Python package such as datetime to get more inspirations. For the continuous thing, we apply StandardScaler.

- Current Ver: Current version of the app available on Play Store (as when scraped)
  The feature appears like "2.0.0", "1.2.2.4". We think about changing the form into the format of number.number and then save the unique values as a list and write a function to order them. Then we could apply OrdinalEncoder. For the thing "Varies with device", we have the same handling as in the feature "Size"(add a categorical feature).

- Android Ver: Min required Android version (as when scraped)
  We find that the usual form of this feature looks like "4.1 – 7.1.1". We split the beginning version and the ending version into two new columns and apply OrdinalEncoder which is the same as Current Ver.

Github link for reviewing code: https://github.com/jiamin-tang/Data-1030-Project.git