Google Play Rating Prediction: Some Machine Learning Strategies and their Performances

葉軒*, 黃光輝†, 劉祖豪‡ and 陳韋翰§
Student ID: *105502035, †105502022, ‡105502511, §105502512
Department of Computer Science and Information Engineering, National Central University

***Abstract*—Google Play Store apps data has enormous potential to drive app-making businesses to success. With actionable insights, developers can build a more attractive product and capture the trends in the market. In this paper, three machine learning algorithms: SVR, Decision Tree, and KNN were applied to test for the best predictive model. The experiment results show that KNN with $K = 15$ has the R^2 score of 91.75% outperforming the other SVR of 91.57% and Decision Tree of 87.79%.**

I. Introduction

Recent years, with the development of information technology, more and more mobile applications are developed. However, reality is always cruel.

Although app designers want to create an app with their hobbies, the app often becomes a failure that no one wants to download it. Besides, many software engineers hope that they can make a popular app, such as ‘angry bird’ or ’ candy crash’ , so that they can earn lots of money.

As for user, we always want to download an useful and fascinating app. However, we often fall into a negative loop, we download an app and then the app is poor and awful, so we uninstall it. This process is time consuming. If we immediately distinguish whether it is good or not, it will be a blessing.

Hence, if there exists a good way to rating a App, users can save the time and developers can make the best profit from the product.

II. Methods

The research flow in this research is shown in Fig.1:

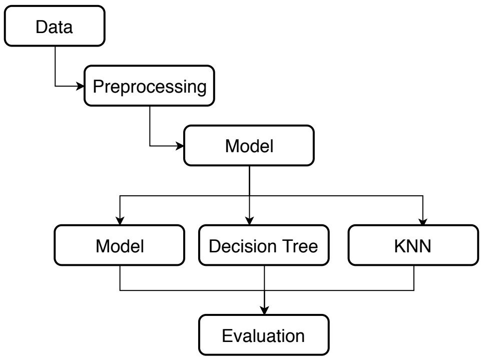

Fig. 1. Step of the experiment

A. Data Preprocessing

In this study, a dataset from Kaggle is used. The dataset contains about 11000 apps with 13 attributes such as name, category, rating, reviews, size, etc. And 5 of these attributes are used to observe the relationship between the corresponding rating and them:

- **Size:** Size of apps (as when scraped)
- **Type:** Paid or Free
- **Price:** Price of apps (as when scraped)
- **Content rating:** Age group the app is targeted at - Children / Mature 21+ / Adult
- **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.

To categorize the different classes in the training process, we need to specify different classes with some integer values. On the other hand, data which miss some attribute values or contains error data format are dropped. As for the *size* attribute, it's converted to the same unit, megabyte (MB).

B. Proposed Algorithms

In our scenario, the rating prediction is a regression problem, so some regression version of classical machine learning methods were applied.

1) Support Vector Regression (SVR): SVR is using the SVM algorithm on regression problem. The goal of SVM is to find the separation hyperplane; while SVR is to find the regression hyperplane. For the given training set $S = \{(x_2, z_1), ..., (x_i, z_i)\}$, where $x_i \in \mathbf{R}^n$ is a feature vector, and $z_i \in \mathbf{R}$ is the target output. In order to find the hyperplane, two parameters $C > 0$ and $\epsilon > 0$ must be given and the support vector regression can be defined in (1):

$$\begin{gathered}\min_{w_J b_J \xi_J \xi^*} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \\ \text{such that} \\ f(x_i) - z_i \leq \epsilon + \xi_i, \\ z_i - f(x_i) \leq \epsilon + \xi_i^*, \\ \xi_i, \xi_i^* \geq 0,\ \forall i \in [1, m]\end{gathered} \tag{1}$$

2) Decision Tree Regression: Select a feature vector X and cut point v to split the original dataset D into subgroups such that the target values in each group is as pure as possible. To prevent overfitting, pruning of the tree should be considered.

3) K-Nearest Neighbors (KNN): For a given sample with feature $X = (x_1, ..., x_n)$, find the nearest k training samples's type with Euclidean distance, and in KNN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors.

C. Performance Evaluation

In the evaluation phase, the coefficient of determination, denoted R^2 is used to score the performance of the trained model, which is defined in (2).

$$R^2(y, \hat{y}) = 1 - \frac{\sum(y_i - \hat{y_i})^2}{\sum(y_i - \overline{y})^2} \tag{2}$$

is the proportion of the variance in the dependent variable that is predictable from the independent variables.

III. Experiment Results

We use the ”scikit-learn”, which is a famous machine learning library in Python to build our experiments. In the training phase, we split the dataset with 75% and 25% for training and testing, respectively.

A. SVR

When we use $C = 1$ to train the data, we got the R^2 score of 91.57% on the SVR method.

B. Decision Tree

The trained model with depth of 2 is shown in Fig. 2. Some information shown in the node are listed below:

- *X:* The set of features
- *MSE:* Mean square error of that node
- *samples:* Amount of training data
- *value:* Average rate of samples

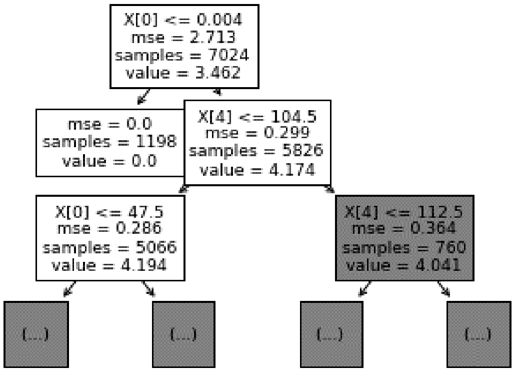

Fig. 2. The decision Tree with $depth = 2$

The experiment result shown that we got R^2 score of 87.79% on the trained Decision Tree model.

C. KNN

In the experiment, we tested for different K from 1 to 20 to find out the appropriate K. According to the results shown in Fig. 3, for $10 \leq K \leq 20$ we will have the similar R^2 scores, when $K = 15$ is selected, the score is 91.75%.

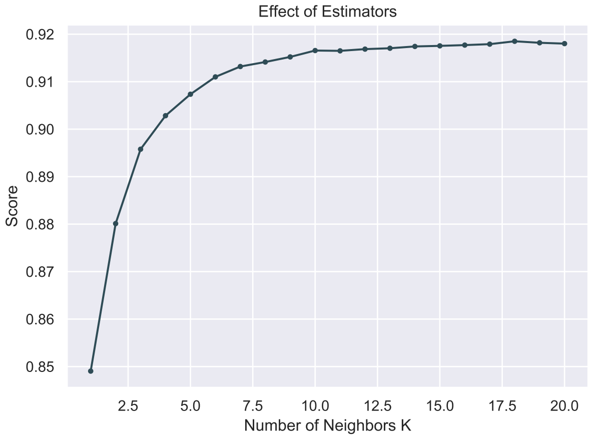

Fig. 3. Scores under different K.

IV. Conclusions

The comparison of the three proposed methods are shown in Table I. Among the three mothods, KNN with $K = 15$ has the best accuracy against the other two, and Decision Tree method had the worst performance. Hence, to do such rating prediction on Google Play apps, SVR and KNN algorithm is more acceptable. With the trained model, we can predict the rating when a app is given with the corresponding features, so that improve the user experience when surfing the Apps market and provide a early evaluation for developing the potential products.

TABLE I
Performance comparison of the proposed methods

Method	**SVR**	**Decision Tree**	**KNN ($K = 15$)**
Accuracy (R^2 score)	91.57%	87.79%	91.75%