**CAPSTONE PROJECT**

**INTRODUCTION TO DATA SCIENCE REPORT**

App Recommender System

*Recommend Google Play app based on user ratings*

**Group 5:**

Nguyễn Trung Thành – 20163727

Bùi Việt Dũng – 20160637

Nguyễn Lê Ngọc Linh – 20162431

Lê Thế Chỉnh – 20160436

Hoàng Minh Giáp – 20161198

**CAPSTONE PROJECT**

**INTRODUCTION TO DATA SCIENCE REPORT**

1. **Overview**

Recommender systems are an important applications of machine learning algorithms that offer items suggestions to potential users. Recommendation system has the two major paradigms: collaborative and content based methods. In our report, we approach to bring a recommendation system using collaborative methods on user ratings

1. **Introduction**

In the general sense, collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc.

One of the most essential criteria is the ratings that users give after their experience with the products. For example, IMDB is the world's most popular and authoritative source for movie with high reliable, user rate after experience of the movies they watched. In general, IMDB is an enormous datasets of movies ratings. As an efficient algorithms of collaborative methods, recommendation engines in our project use dataset of user rating from Google Play app to build a system to recommend most rating apps.

1. **Aim of the project**

The aim of this project is to train a machine learning algorithm model to predict ratings of apps from the association between users and Google Play’s apps

Two main algorithms of collaborative filtering:

* Nearest Neighborhood: the process is to calculate the similarities between target user/items and all other users/items, select the top X similar users/items, and take the weighted average of ratings from these X users with similarities as weights.
* Matrix Factorization: Since sparsity and scalability are the two biggest challenges for standard CF method, it comes a more advanced method that decompose the original sparse matrix to low-dimensional matrices with latent factors/features and less sparsity

In our project we use Matrix Factorization based on Singular Value Decomposition (SVD) to perform principle component analysis (PCA) that aims to decompose a matrix (usually a set of observations) in order to find the directions (called principal directions or principal axes) in which the observations have the largest variance.

**3. Dataset**

- The dataset used in this project has been obtained from Google Play app. **data.csv** is a dataset that contains 23027 ratings from 22351 users for all 298 particular apps. Data points include app identity, app name, username, user identity, score given by user.

- **Problem:** Due to crawling process, we only get reviews of 298 applications so we get into a “real situation” imbalanced data. In detail, the dataset contains over 17000 values of rating 5, and the rest values only from 100 – 400 values.

- **Solution**: We have researched some resampling techniques such as: Undersampling, Oversampling, combination of undersampling and oversampling. In this project, for simplicity we approach undersampling built-in scikit-learn. We randomly drop values of rating 5 to make the dataset more balanced

A screenshot of a cell phone

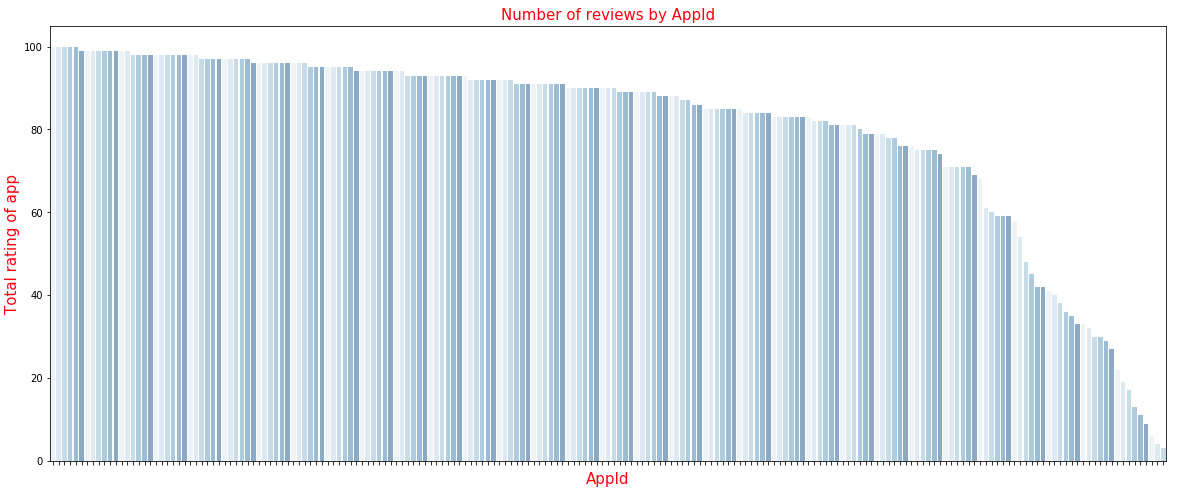
Description automatically generated

Preprocessing

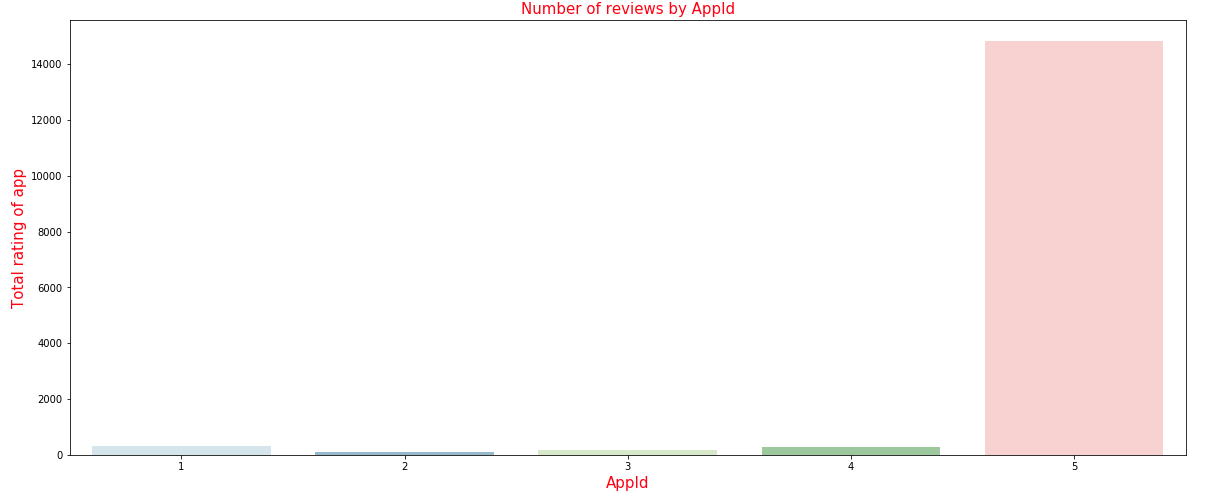
* Drop Duplicate
* Drop NA
* Random Under-sampling

- Some analysis:

**Before Preprocessing:**



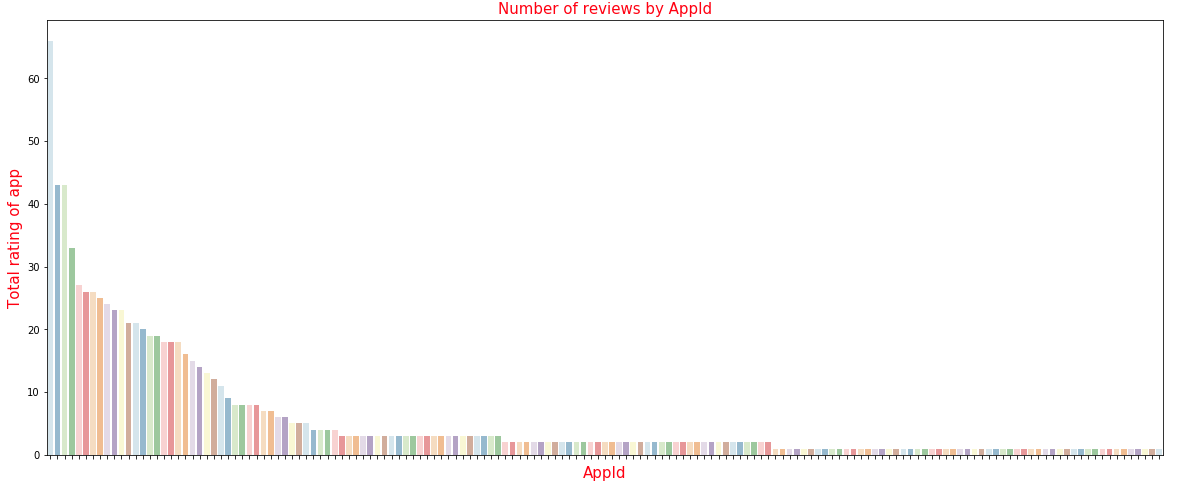
Number of application rattings craw by us (max 100)

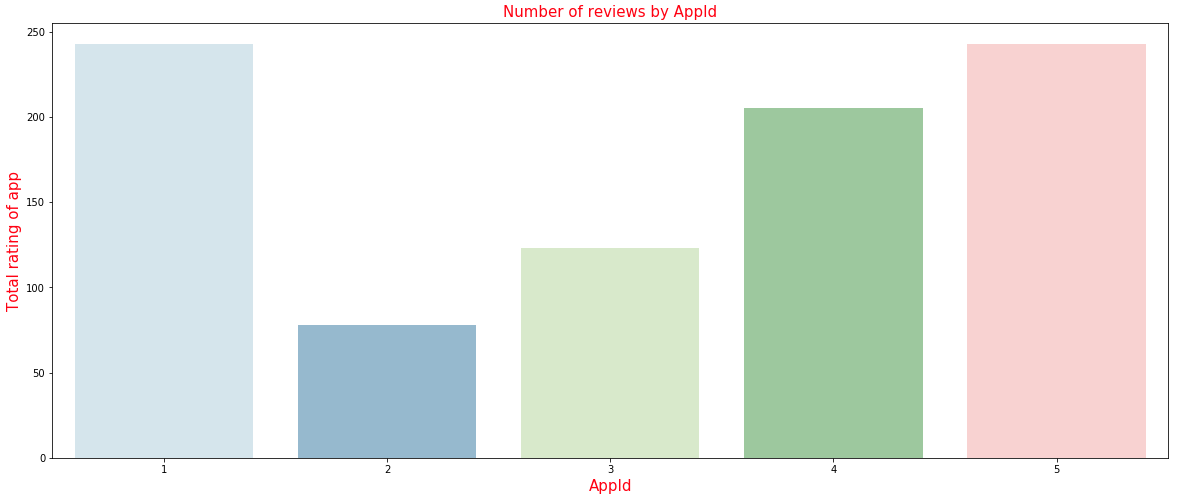


Number of ratings from 1 to 5 stars

\*For now, we have imbalanced data with 5 stars rating apps. The number of 5 staring apps dominate other ratings.

**After preprocessing by under-sampling**





\*As we can see, the data is more balance after we used under-sampling method

1. **Methods and Analysis**

**Content Filtering** → creates a profile for each user or product to characterize its nature

**Collaborative Filtering** → analyzes relationships between users and inter-dependencies among products to identify new user-item associations

Collaborative filtering is generally more accurate then content filtering however, it suffers from cold start problem. (If new user exists and does not have any inter-dependencies among others, we can’t recommend anything

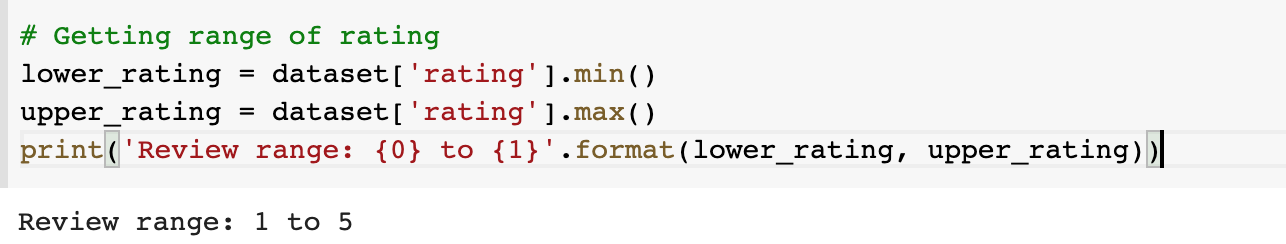
In this project we used matrix factorization technique to build a sophisticated recommender system in which outperformed nearest-neighbor techniques.

Most of the MF models are based on the latent factor model. Matrix Factorization approach is found to be most accurate approach to reduce the problem from high levels of sparsity in RS database, certain studies have used dimensionality reduction techniques.

1. **Training the model**

We use matrix factorization with approach Singular Value Decomposition (SVD) which requires some parameters to optimize the performance of model such as learning rate, regularization term for all parameter

First step, we will check the range of rating to read into the dataframe



Due to this range, we will tune our model to specify the parameters above

A screenshot of a cell phone

Description automatically generated

These parameters are the best parameter we will use for SVD model from surprise library

Second step, we will train the model with the above parameters

A screenshot of a cell phone

Description automatically generated

1. **Recommending section**

For the recommendation process, with the user id we filter out the list of applications that user didn’t rate from the list of all applications

A screenshot of a cell phone

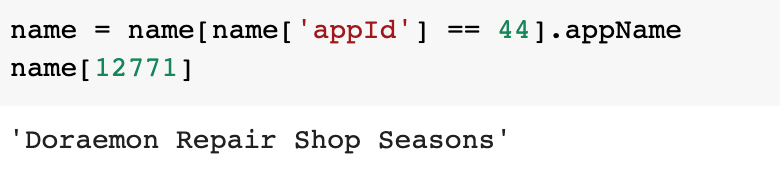
Description automatically generated

Next. We get the app id with highest predicted rating of this user

A screenshot of a cell phone

Description automatically generated

From here, we can find the name of application with this id



1. **Evaluation**

To evaluate we will test the algorithm using the test\_set obtained from train\_test\_split the dataset by the RMSE

The **root-mean-square deviation (RMSD)** or **root-mean-square error (RMSE)** is a frequently used measure of the differences between values (sample or population values) predicted by a model or a estimator and the values observed

A screenshot of a cell phone

Description automatically generated

The RMSE is quite high. This might be caused by our preprocessing data

We also use cross\_validate to see the performance between each fold

1. **More**

Our project needs to improve the accuracy, performance

We will research more in:

* Handling imbalanced data with SMOTE, TomekLink
* Try to apply and evaluate other method of surprise as: Non-negative matrix factorization, Basic algorithms, k-NN algorithms, slope one, co-clustering
* Try to crawl more data

**Resources:**

* *We write our own crawler using NodeJS*

[*https://github.com/petixiuxx/Google-play-store-crawler*](https://github.com/petixiuxx/Google-play-store-crawler)

* *Research about Recommender system: Course Machine learning in Coursera*
* *Ref surprise documentation:*

[*https://surprise.readthedocs.io/en/stable/*](https://surprise.readthedocs.io/en/stable/)

* *Handling imbalanced data*

[*https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets*](https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets)