Capstone Term I
05/12/2022

Mobile App Size, Rating & Installs Prediction

Group **#14**

Bindya Biju
ID: 100886575

Sanath Davis
ID: 100884693

Abstract

The final project report document takes the reader through the final design of various experiments on the dataset and their results. It reiterates the motivation behind the project and delves into the details of the dataset. The existing projects based on similar datasets are also explored. A concluding note is also attained.

Keywords: google, play store, price, size, category, installs, ratings

1. Introduction

In this day and age of mobile apps, every small decision made by mobile app developers will affect the sales of an app. There are many crucial decisions to be made, like the price of an app or the size of an app, which would lead to maximum profits, better user ratings, and the maximum number of installs. These decisions might vary depending on the category of the app.

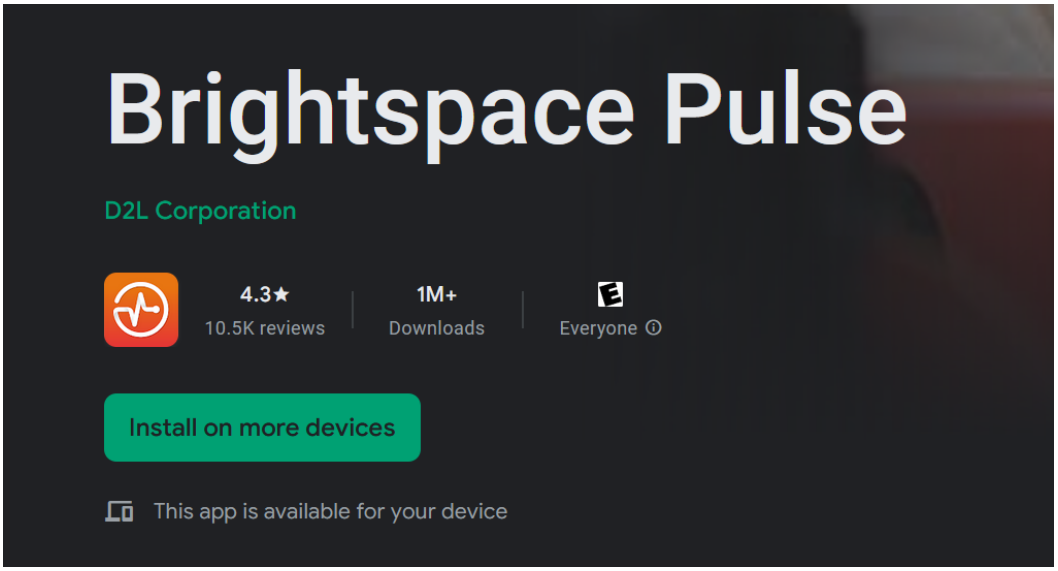

Figure 1. Details of an app in Google Play Store (Representative image)

After our machine learns this dataset, it would then be able to predict the ratings and number of installs an app would get given a particular size, category, and price. Also, we

* sanath.davis@dcmail.ca, bindia.biju@dcmail.ca

explore the preferred size of the app for which the app gets the maximum customer rating and the maximum number of installs. So, the machine would use the data to tell developers about what kind of apps and their size the app should be to maximize the number of installs. Most of the existing work focuses on analyzing either the app category or the price of the app. We want to focus on the effect of the size of the app too and the influence it has on the number of installs and ratings. Our work would cater to independent developers who lack the resources to do large-scale marketing.

2. Related Work

- The Impact of Price on Google Play Apps [1]

  This provides various guides on cleaning the dataset, especially the conversion of various datatypes. A very good visualization provides information about the most expensive apps in each category and the price trends.

  This focuses mainly on the price aspect and doesn't explore the size aspect and how it impacts the number of installs and customer ratings. Our work also compares various regression models and predicts the ratings and the number of installs an app would get.

- Mining and Analysis of Apps in Google Play [2]

  The paper is trying to find various intrinsic factors in the play store. It finds a good correlation between price and the number of downloads. They have used K means clustering.

  Our work also relies on various correlations. But we are focused on predicting the rating an app would get and we have used multiple regression models for it.

- Characteristics of successful 'free' apps [3]

  The author is trying to find the X factor which makes a free app successful. This is more in line with the aim of my project as it focuses on free apps which will be more useful for independent developers. This too focuses more on the category of the app. It concludes that apps with elements of communication or productivity tend to get more downloads on average.

  In our project, we are trying to find conclusions about how various factors will lead to better ratings and more installs. We are also predicting the user rating a particular app would get.

- Analysis of Google Play Store Data set and predict the popularity of an app on Google Play Store [4]

  Here the authors have divided each app into one of two categories: successful or unsuccessful, based on the number of installs. Then they have done it as a classification problem and obtained that the Decision Tree model gave the best accuracy.

  In our project, we are looking at it as a regression problem instead, and we are trying to predict the app rating and the number of installs. Even though we have created new features using feature engineering like the authors above, ours was not for classification.

3. Experimental Design

Since our dataset has the data in a labeled format, we decided to treat it with supervised learning techniques. By looking at the properties of the target variable(output), namely App Rating and Number of Installs, it falls in the continuous variable division and this makes it a regression problem.

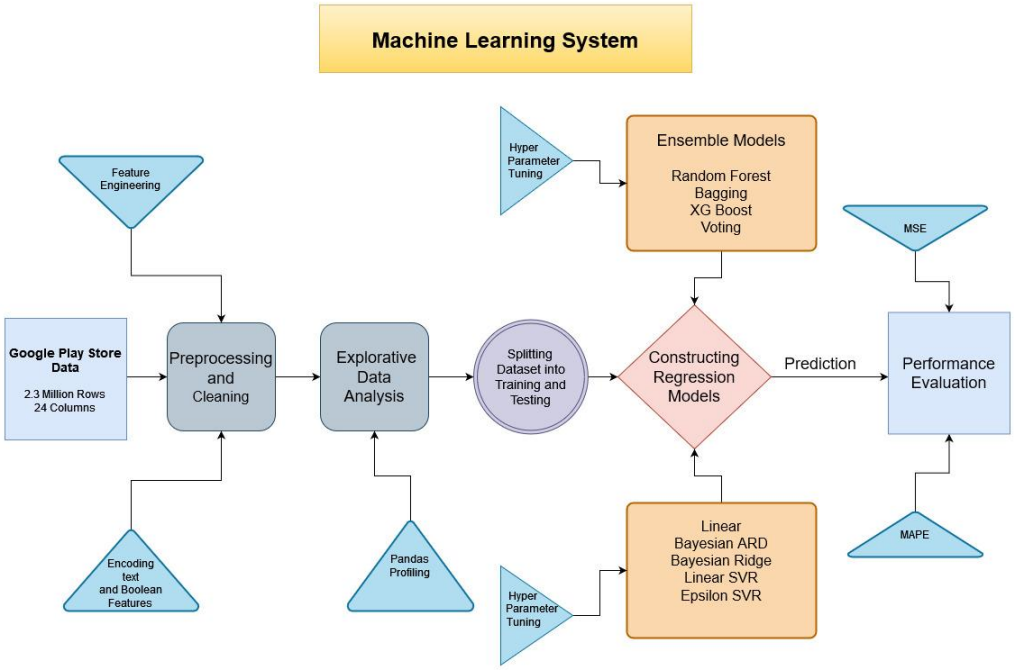

Figure 2. Flow chart of the machine learning system

We set up an environment with our dataset loaded and conducted various experiments on it. We cover the details about our setup and the tools and techniques used in our experiments here.

3.1. Main purpose

We want to look into three key areas at this stage:

- ➢ Find the size range in which the most numbers of installs are obtained for an app in different categories
- ➢ Predict the Average User Rating an app would get, based on the features like category, price, and size
- ➢ Predict the number of total installs an app would get, based on the features like category, price, and size

3.2. Dataset

We will be using the Google Play Store App data of 2.3 million+ applications. It contains almost all publicly available features of an app like its price, size, category, user rating, etc. This was collected by Gautham Prakash and pushed into Kaggle. The data was collected with the help of a Python script called Scrapy running on a cloud VM instance. GitHub tool Google Play Scraper [5] was also used for the scraping process. This dataset was updated as recently as last year.

Link:
https://www.kaggle.com/datasets/gauthamp10/google-playstore-apps [6]

3.3. Method

The methods used in the project include pre-processing, modeling, and performance evaluation. The flow chart items shown in Fig 2 are discussed in detail below:

3.3.1. Minimum Requirements

You need to have the following installed to run the notebook in a Python environment.

- Pandas
- Sklearn
- Numpy
- Seaborn
- Matplotlib
- Pandas Profiling
- XG Boost

To handle multiple environments, the Anaconda tool is recommended, but not compulsory.

3.3.2. Loading the Dataset

We read the dataset in CSV format using Pandas and converted it into a Pandas Dataframe.

3.3.3. Pre-processing the dataset and cleaning

This step is crucial for the success of the project. So we were well involved in cleaning the data and priming it for modeling and testing.

- We identified some features like 'Privacy Policy, 'Scraped Time', 'Developer Id', etc which were not very important for our experiments and removed them from the dataset
- We counted the number of rows that contain NULL and NAN values and used the 'dropna' functionality to remove these 29,429 rows
- The representative number of Installs which had terms like 100+ and 10,000+ were converted into normal integers. For such purposes, cleaner functions were written and applied all across the specific data frame column
- The app size column, which is of great importance to the project, had different units in it. Some apps had sizes in Kilo Bytes and Giga Bytes, while most apps had sizes in Mega Bytes. We converted all the sizes to Mega Bytes and dropped the units
- There were about 1,227 apps with their prices marked in a currency different from USD. Since this is a small number, we decided to remove these apps so that all apps would have prices in USD
- The minimum android version was found out from phrases like '4.1 and up' (4.1) and '6.0 - 7.1.1' (6.0) and converted into float datatype

3.3.4. Feature Engineering

We have identified some derived features that are likely to influence the App Rating and Number of Installs intuitively. These features are derived from existing features.

- Whether the App Name has the term 'Free' in it - '*has_free_in_app_name*'
- Whether the app was released in the last year - '*released_within_one_year*'
- Number of characters in the App Name - '*no_of_characters_in_app_name*'

3.3.5. Encoding Boolean Features

Since we are going for regression we want to convert meaningful text into number representations to preserve the context and relationships. Some columns like 'Free Column', 'Ad-Supported Column', 'In-App Purchases Column', and 'Editors Choice column' are all boolean. We replace all these boolean values with 1s and 0s.

3.3.6. Encoding Text Features

We encode text features like 'Content Rating' and 'Category' as a one-hot numeric array using the OneHotEncoder of Sklearn. [7]

3.3.7. Explorative Graphs and analysis

We have drawn multiple types of graphs to better analyze the data and draw inferences. The Box Whisker plot helps us to visualize the maximum installs of different categories. The Correlation graph which uses Kendell's rank coefficient helps us to identify various correlations between the features. The density graphs about the Ratings and

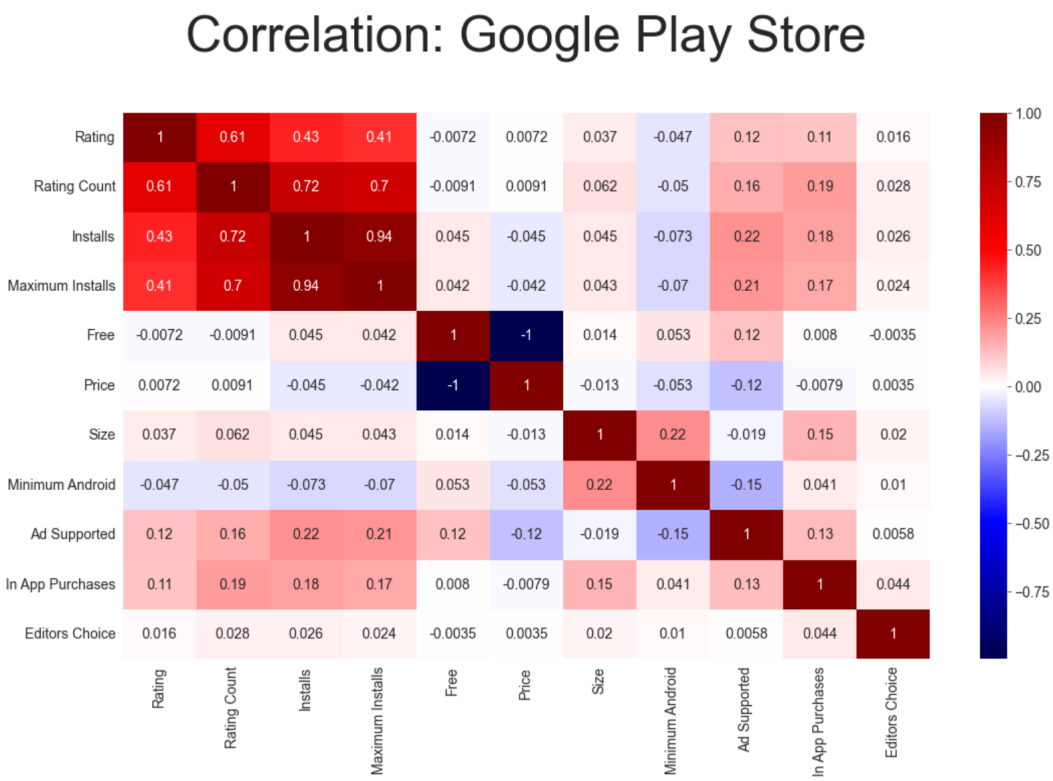

Figure 3. Correlations using Kendall's and 'seismic' color coding

Sizes help us identify the concentration of apps in various sizes and ratings. Then we draw various scatter plots to visualize our dataset further.

3.3.8. Pandas Profiling

We can do a full round of preliminary data explorations using the very handy Pandas Profiling tool. This generates a report which is downloaded when the notebook is run. The profile is also visible in a block inside the notebook. The profile contains various types of correlation matrices and interactions. Also, for easier access, we have hosted the profiling online. [8]

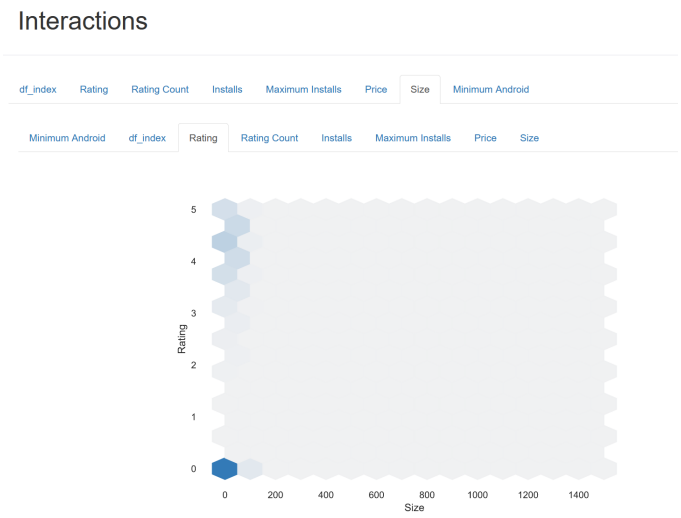

Figure 4. Interactions section in the Pandas Profile Report

3.3.9. Tackling Question 1: What should be the size of an app for it to get more installs?

We tackle this question as a mathematical problem. We look into each category and see which app size has gotten the most installs. For this, we find the maximum app size in each category and convert the app sizes to various ranges. Then we plot bar graphs that show the relationships between the number of installs and the size ranges.

3.3.10. Tackling Question 2: Predict the Rating of an app

We want to predict the rating an app would get given its features like app size, category, price, etc. For this, we first prepare the dataset for regression modeling. We use the Sklearn library to split the dataset into two parts. 60% of the cleaned dataset is now set aside for training, while the remaining 40% will be used later for testing and measuring performance. We use various regression models and try them out for predicting the rating of an app. Some of the regression models we use are

- ➢ Linear Regression
- ➢ Random Forest Regressor
- ➢ Bayesian ARD regression
- ➢ Bayesian ridge regression
- ➢ Linear SVR regression
- ➢ Epsilon SVR regression

3.3.11. Tackling Question 3: Predict the Number of Installs of an app

We follow the similar methods followed above for predicting the rating of an app. We just replace the target of the training and testing and do all the above regression analysis here too.

3.4. Baseline

As a baseline, we have chosen the Linear Regression model. We see that this would be the simplest regression method available. Also once we checked the R2 score for linear regression, it indicated a possible good fit. Linear regression tries to give us a best-fit-line using which we can make predictions. This model gave us quite a big mean square error in both cases and this served as a good baseline for reducing the error rates.

3.5. Evaluation Metrics

Since we are handling a regression problem, we are always trying to reduce the error in prediction. To calculate the error rates, we use the test dataset we had split

earlier. Some of the metrics we can use for the regression problem are Mean Absolute Error, Mean Squared Error, Root Mean Squared Error, R Squared, etc.

Mean squared error	$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\mathrm{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

Figure 5. Mathematical Formula for different types of Errors

We decided to mainly focus on the Mean Squared Error (MSE) and the Mean Absolute Percentage Error (MAE) for this project.
One important factor was that there the MAE matric strictly demand that there should not be any infinity or NAN or NULL values, and we have spent considerable resources on making sure this is the case. Another advantage is that this metric is robust to outliers. We also picked MSE metrics because, unlike MAE, it is more sensitive to outliers. One possible disadvantage is that we will not be able to infer the direction of the error to say whether we are undercutting or overfitting. [9]

4. Fine Tuning - Improving Performance

Once we have used various regression models on our dataset, we looked into ways of fine-tuning the models and improving performance. There were two focus areas:

4.1. Using Ensemble Models

The ensemble models try to combine the predictions from multiple models to produce more accurate predictions. There are generally three types of ensemble models

- ➢ Bagging
- ➢ Boosting
- ➢ Voting

In this category, we have used the Bagging Regressor, XG Boost Regressor, Random Forest Regressor, and Voting Regressor.

4.2. Hyperparameter Tuning

As we know each model has many parameters associated with them. Changing them could lead to many performance benefits. But it might not be always possible to do this manually. So we depend on two types of Hyper Parameter Searching

which automatically give us the best possible parameters for a model while running on our dataset:

- ➢ Grid Search
- ➢ Randomised Search

They find out the parameters which result in the least error and we can proceed to improve our existing models by plugging in these 'best' parameters. In the project, we have conducted the Grid Search on the Support Vector Regressor and the Randomised Search on the Random Forest Regressor and the XG Boost regressor.

5. Experimental Results

5.1. Main findings and Results

- ➢ Through the various experiments we found out how to properly clean the dataset, specifically convert the datatypes of important features and make them suitable for regression.
- ➢ We found out means for engineering new and derived features from existing features via feature engineering
- ➢ Through the various explorative analyses and the pandas profiling, we got many inferences about the data including correlations
- ➢ The first main finding is that we found out the best app size ranges for getting the maximum number of installs in different categories.

Category	**Minimum Size**	**Maximum Size**	**Best Size Range to get Maximum Installs**
Adventure	0.045 MB	1100 MB	800-900 MB
Maps & Navigation	0.016 MB	382 MB	150-200 MB
Role Playing	0.043 MB	1500 MB	1000-1200 MB

Figure 6. Best Size range in different categories

- ➢ Then using various regression models we were able to predict the rating and number of installs
- ➢ We chose the linear regression as the baseline and calculated the MSE and MAPE for it
- ➢ We then conducted various other regressions and found the model with the lowest error. We found that for predicting the Ratings, **the Random Forest Regressor provided the best predictions and the lowest error**

➢ Predicting App Ratings

Regression	MSE	MAPE
Linear (Baseline)	4.221	4354160987210370.0
Random Forest	0.4	0.07
Bayesian ARD	2.058	4369784657954465.5
Bayesian ridge	2.054	4354193448350276.5
Linear SVR	2.508	3135247702318329.5
Epsilon SVR	Taking too much time to fit	Taking too much time to fit

Figure 7. Results from multiple regression models

➢ While predicting the Number of Installs we found that the error rate was quite high in all the regressors
➢ We found that the Epsilon SVR is not suitable for our particular dataset
➢ Inspired from the success of the Random Forest model, an ensemble model, we used three different types of ensemble models to check the performance. XGB and Voting regressors gave good MSE rates, while overall, Random Forest still provided the best predictions. These were the results:

Ensemble Model	MSE	MAPE
Random Forest	0.4	0.07
Bagging Regressor	2.08	5121046823101658
XG Boost Regressor	0.45	10671825510858
Voting Regressor	0.45	10448593663074

Figure 8. Results from different ensemble regression models

➢ Using Randomised and Grid Searches, we were able to find the best parameters for different models which produce lower error rates. This lead to improvements in the prediction rates and gave us a valuable lesson on the importance of choosing the best parameters. These were the results:

Tuned Model	Best Parameters	MSE Improvement	MAPE Improvement
Random Forest (Randomised Search)	{'n_estimators': 200, 'min_samples_split': 5, 'min_samples_leaf': 4, 'max_features': 'auto', 'max_depth': 10, 'bootstrap': True}	3.2 %	4.08%
Support Vector Regressor (Grid Search)	{'C': 10.0, 'gamma': 0.01}	50%	70%
XG Boost Regressor (Randomised Search)	{'subsample': 0.8999999999999999, 'n_estimators': 1000, 'max_depth': 3, 'learning_rate': 0.01, 'colsample_bytree': 0.8999999999999999, 'colsample_bylevel': 0.7999999999999999}	.01 %	175%

Figure 9. Improvements from Hyper-parameter Tuning

5.2. General Trends and Insights

- Compared to the baseline, most models performed better.
- Some models are not suitable for the dataset
- Since we are predicting using regression we found that we should convert all the different features to numeric. Even the text fields should be encoded in this manner.
- When we use the best hyper-parameters in each model, there is a good improvement in the prediction rates. This is because Grid Search/ Randomised Search finds out the best parameters for that model which gives better accuracy for our Playstore dataset. So even if this is an expensive process, even if it provides a small improvement in the predictions, it would be worth it. But we noted that Grid Search takes a very large amount of time and we should consider using Randomised search instead, which itself takes a reasonably large time to complete running on a big dataset as ours.
- Usage of Ensemble methods resulted in good MSE, but MAPE rates didn't improve significantly in cases other than Random Forest when compared to the baseline. Ensemble models combine a variety of models together to help the prediction process. But we understood that Ensemble methods are less explainable and result in the predictions being less understandable to the users.

5.3. Defeats and Next Steps

- We can engineer other features from the existing features and find out their influence on the predictions
- While we were able to predict the App Rating with good error rates, our selected models weren't efficient enough when we attempted to predict the number of installs. We might conclude that it might not be possible to predict the number of installs using the given features
- The next future step is making our solution ready for production by
    - Polishing the code
    - Preparing Documentation
    - Writing Test Cases and resolving any bugs
    - Recreating the required environment in production
- After that, we will move on to deploy our hyper-tuned Random Forest model on to the production environment. This will be achieved by saving our trained Random Forest model, transferring the files to production, and loading it there.
- In this way we can make our model into an API that can be reached by any service.
- Also, we can create our own service which calls the API. This service would have a UI interface and will be hosted online using which the App Developers can predict how their app would be rated, what kind of app they should focus on making, etc
- In the future, we also need to keep our data and models updated, as this Google Play Store dataset is updated by the source on a yearly basis
- Also we can look at ways of extending a similar approach towards ***Apple Store*** Apps too.

Conclusion

The Random Forest Model was found to be the best suitable for the prediction of app user ratings. It is showing better performance due to it being an ensemble method. Also after conducting a Randomised Search and fine-tuning the model, we achieved better results. We were able to evaluate the performance of the models using MSE and MAPE. The results indicate that compared to the baseline, ensemble models are best suited for this dataset. Also, the results tell us the importance of hyperparameter tuning and proper data pre-processing. But we also understand that there might be features that affect the rating and number of installs an app gets, which might not be included in our dataset.

Ethical Implications

- The number of ***false app ratings*** is a crucial ethical problem to take into account. For instance, an app developer may encourage favorable ratings to alter customer perception in order to boost the possibility that a consumer will download an app. Such dishonest action is both immoral and against the law. Sadly, false reviews might go unreported, leading to bias in our model. [10]
- Allowing app developers to know what kind of apps they should make to gain more money might end up killing the creativity of developers and the natural enjoyment humans get when they create something unique.
- The use of this predictor might result in more and more apps looking very similar to each other

Acknowledgments

The Capstone Project is done under the guidance of Ruba Al Omari, Faculty of Science, Engineering & Information Technology at Durham College, Oshawa, Ontario.

Bibliography

[1] Omarmoneer, O. (2022, July 31). The impact of price on google play apps. Retrieved October 30, 2022, from https://www.kaggle.com/code/omarmoneer65/the-impact-of-price-on-google-play-apps

[2] Mokarizadeh, S., Rahman , M., & Matskin, M. (2013). Mining and analysis of apps in google play. *Proceedings of the 9th International Conference on Web Information Systems and Technologies*. https://doi.org/10.5220/0004502005270535

[3] Mattkirtley. (2021, June 12). Characteristics of successful 'free' apps. Retrieved October 30, 2022, from https://www.kaggle.com/code/mattkirtley/characteristics-of-successful-free-apps

[4] Maredia, R. (2020). *Analysis of Google Play Store Data set and predict the popularity of an app on Google Play Store*. Retrieved from https://www.researchgate.net/publication/343769728_Analysis_of_Google_Play_Store_Data_set_and_predict_the_popularity_of_an_app_on_Google_Play_Store

[5] Facundoolano. (2016). Facundoolano/Google-play-scraper: Node.js scraper to get data from Google Play. Retrieved October 30, 2022, from https://github.com/facundoolano/google-play-scraper

[6] Prakash, G. (2021, June 17). Google play store apps. Retrieved October 30, 2022, from https://www.kaggle.com/datasets/gauthamp10/google-playstore-apps

[7] Sklearn.preprocessing.onehotencoder. (n.d.). Retrieved October 30, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

[8] Davis, S. (2022, October 29). Pandas Profile of Google Playstore Data. Retrieved October 30, 2022, from https://sanathdavis.github.io/

[9] Swalin, A. (2018, July 10). *Choosing the right metric for evaluating machine learning models - part 1*. Medium. Retrieved December 5, 2022, from https://medium.com/usf-msds/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4

[10] Karakolis, E., Oikonomidis, P. F., & Askounis, D. (2022). Identifying and addressing ethical challenges in Recommender Systems. *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*. https://doi.org/10.1109/iisa56318.2022.9904386