Predicting the average number of downloads of an app on Google Play Store by analysing Google Play Store Data set

Aditi Mittal

SRM University

Kattankulathur, Chennai

NORDIC ACADEMY

# Abstract

*The google play store is one of the largest and most popular Android app stores. It has an enormous amount of data that can be used to make an optimal model. We have used a raw data set of Google Play Store from the Kaggle website. This data set contains 12 different features that can be used for predicting whether an app will be successful or not using different features. This data set is scraped from the Google Play Store. This report talks about different regression models that we used for prediction purposes and finding which one gives the least mean absolute error. This report also gives detailed information on the scaling of features and Data visualization done on this data set. Our project code can be found at https://github.com/Rimshamaredia/CSCE421-Project.*

# Introduction

Mobile applications are one of the fastest-growing segments of downloadable software application markets. Out of all of the markets we choose Google Playstore due to its increasing popularity and recent fast growth. One of the main reasons for this popularity is the fact that about 81% of the apps are free of cost. The market has increased to over 845900 Apps and 226,500 unique sellers in April 2013. This rapid market has, in turn, led to over 500 million users downloading around 40 billion Apps all over the world. Developers and users play key roles in determining the impact that market interactions have on future technology. However, the lack of a clear understanding of the inner working and dynamic of popular app markets impacts both the developers and users. In this report, we seek to shed light on the dynamics of the Google Play Store and how we can use different features from this data set for prediction purposes.

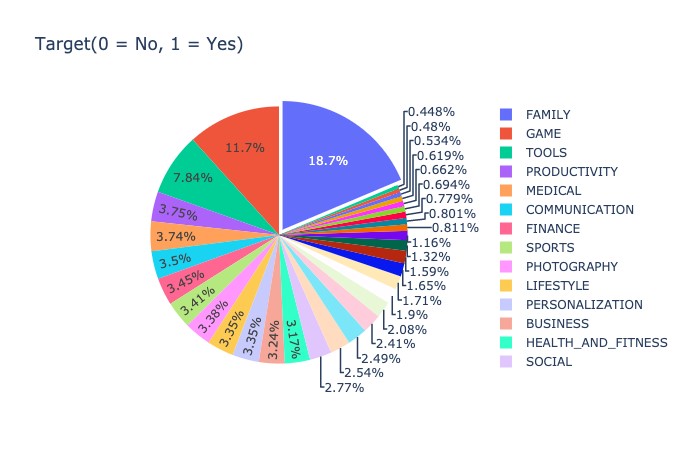


Figure 1. Number of apps available for download by category.

In this report, we will provide a longitudinal study of Google Play store dataset which will help in predicting the average number of downloads of an app on Google Playstore. Our Analysis is divided into the following phases: data extraction, data cleaning, data visualization, exploratory data analysis and applying different models. First, we collect the data from the Kaggle website. In the next step, we try to do data cleaning on the data set to reduce the error percentage. After the data set is ready, we try to analyse the data set using different plots and remove the stuff not needed from the data set. The last step includes using different regression algorithms on the data set to see which one gives the lowest percentage of mean absolute error. Finally, we narrate the analysis results to provide a clear vision of the relationship among the areas of interest.

# Analysis Methodology

Our analysis approach is divided into the following phases: data extraction, data cleansing, data visualization, EDA and data modelling.

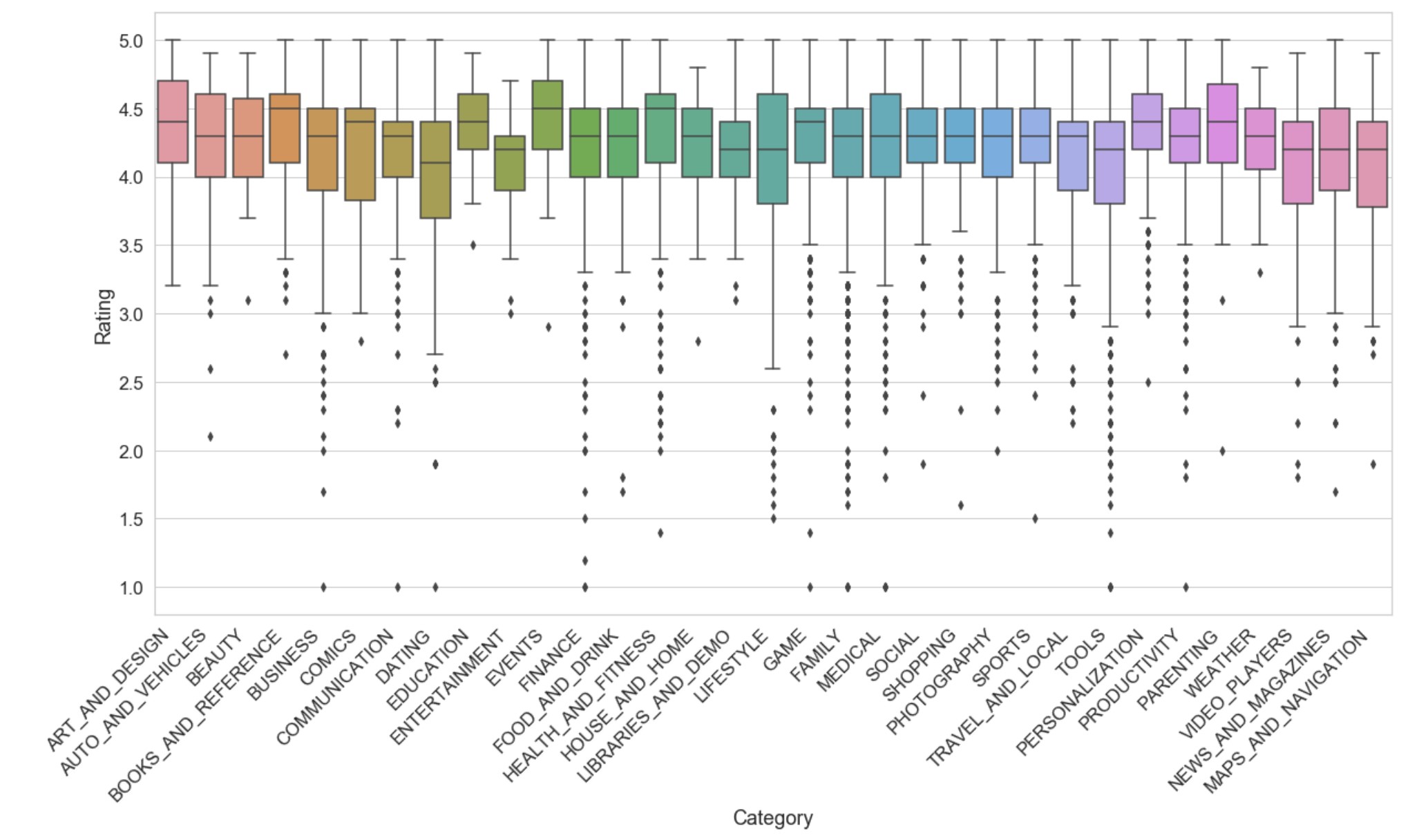


Figure 2. Box-plot to find average ratings.

In the first step, we collected the raw data from Kaggle. Then we did basic data cleaning and data visualization. After visualizing the data set, we removed some unnecessary features and made it ready for data modelling. Next, we conduct data modelling by using various regression models like Cat Boost, XG Boost, Random Forest and Gradient Boost.

2.1. Data Visualization

In this data set there are various features that can be used to analyse the data set. In this section we will be analysing different features to find which feature determines the average number of downloads of an app. The top five categories of the google play store include Education, Entertainment, Music and Audio, Business and Tools. This shows that apps that belong to these categories have high chances of being downloaded frequently. There are only 1.75% of paid apps in Google play store. The majority of paid apps cost about $1. So, the average downloads don’t depend on this factor.

While looking at the average download feature, we can see that the number of rows where average number of downloads is around 1000 to 2,200 are quite high. To further define which category has the highest number of downloads, we will only look at the data for each category that has more than or equal to 1000 in the average number of downloads column. This will help us in making more optimal model. To analyse the apps that would produce the most revenue, we will look at the correlation table between downloads and other parameters. From figure 3 we can see that installs and reviews have the strongest inverse correlation. This is reasonable because popular apps tends to get more number of reviews. There is no correlation found between installs and other features like size,rating,number of installs and price. There is no correlation between rating and price also. Since installs parameter is independent and not correlated to any other parameters, we must only use installs to show the popularity of an app. For our analysis we will be only using number of installs for classifying whether the

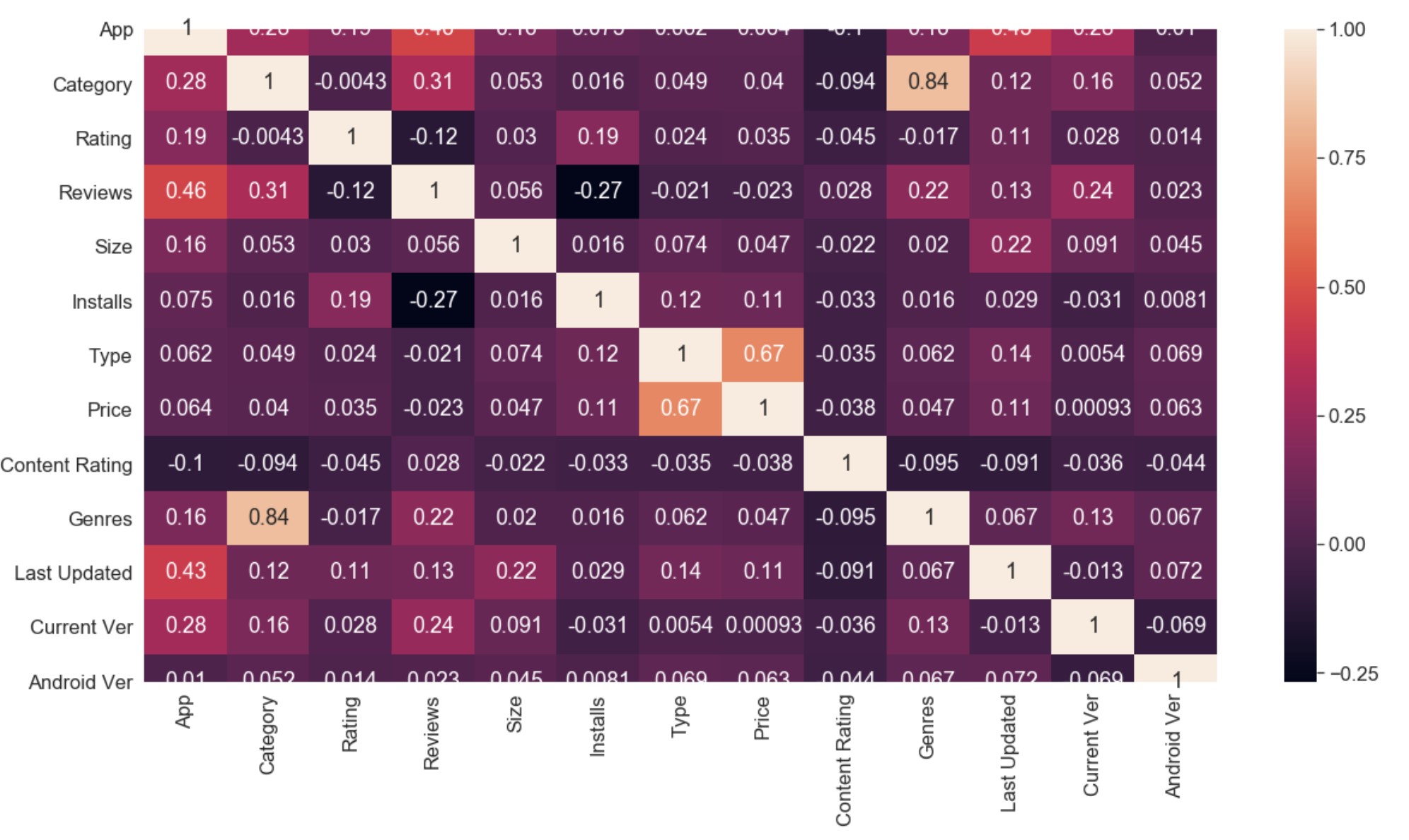


Figure 3. Correlation between installs and other parameters

|  |  |
| --- | --- |
| Algorithm | Accuracy % |
| Decision Tree | 95.32% |
| K-nearest neighbor | 90.59% |
| Gaussian Naive Bayes | 88.45% |
| Logistic Regression | 89.86% |

Table 1. Results.

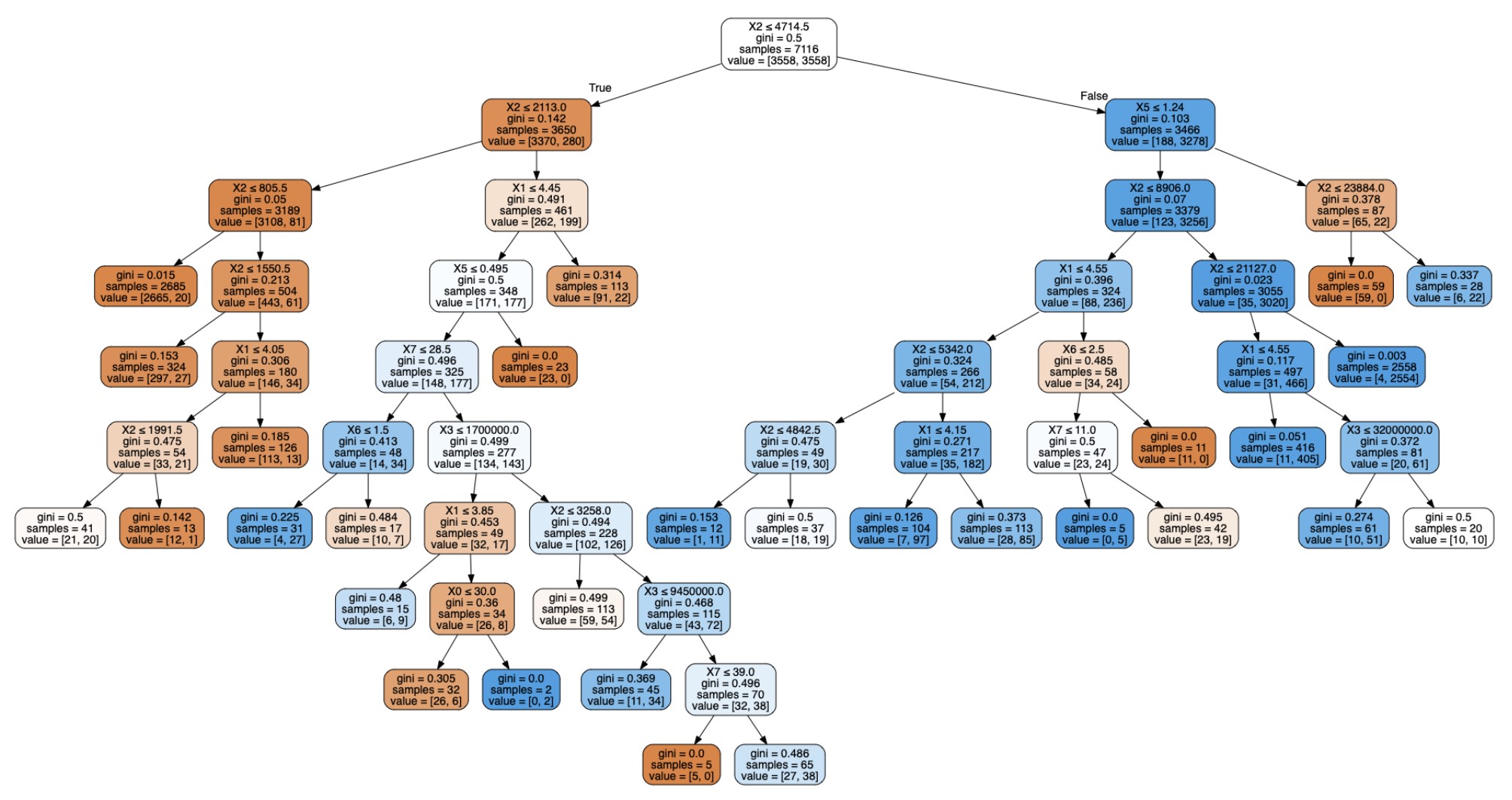


Figure 4. Decision tree chart

app is successful or not. If the number of installs for a particular app is greater than 1000, it will go under the class of successful and if it is less than 1000 we will classify it as unsuccessful.

2.2. Regression Models

## 2.2.1 Cat Boost Regression Model

CatBoost is a recently open-sourced machine learning algorithm from Yandex. It can easily integrate with deep learning frameworks like Google’s TensorFlow and Apple’s Core ML. It can work with diverse data types to help solve a wide range of problems that businesses face today. To top it up, it provides best-in-class accuracy.

It is especially powerful in two ways:

* It yields state-of-the-art results without extensive data training typically required by other machine learning methods, and
* Provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems.

“CatBoost” name comes from two words “**Cat**egory” and “**Boost**ing”. This library works well with multiple **Cat**egories of data, such as audio, text, image including historical data. “**Boost**” comes from gradient boosting machine learning algorithm as this library is based on gradient boosting library.

Advantages of CatBoost Library

* **Performance:**CatBoost provides state of the art results and it is competitive with any leading machine learning algorithm on the performance front.
* **Handling Categorical features automatically:**We can use CatBoost without any explicit pre-processing to convert categories into numbers. CatBoost converts categorical values into numbers using various statistics on combinations of categorical features and combinations of categorical and numerical features.
* **Robust:**It reduces the need for extensive hyper-parameter tuning and lower the chances of overfitting also which leads to more generalized models. Although, CatBoost has multiple parameters to tune and it contains parameters like the number of trees, learning rate, regularization, tree depth, fold size, bagging temperature and others.
* **Easy-to-use:**You can use CatBoost from the command line, using a user-friendly API for both Python and R.

The CatBoost library can be used to solve both classification and regression challenge. For classification, you can use “***CatBoostClassifier***” and for regression, “***CatBoostRegressor***“.

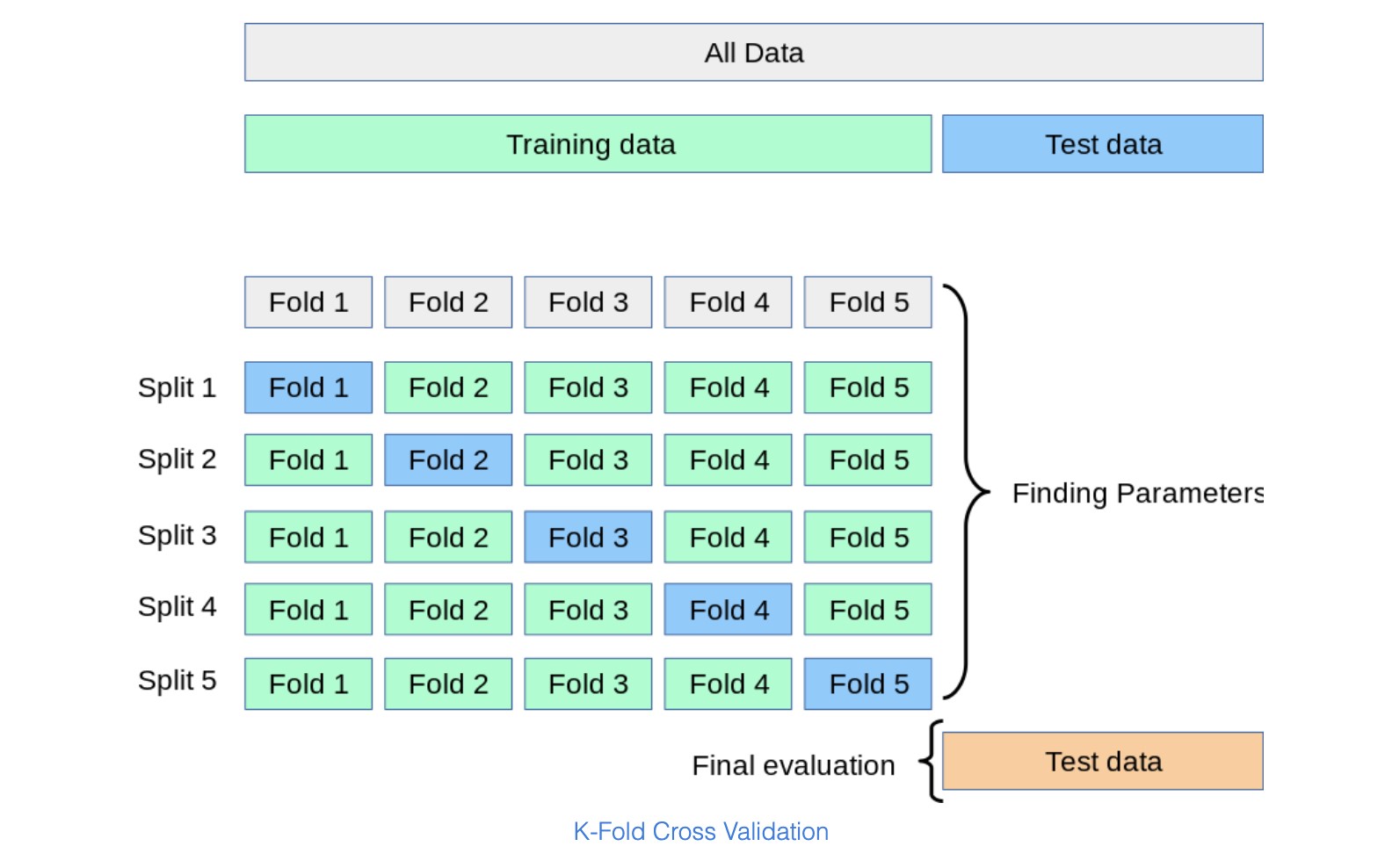


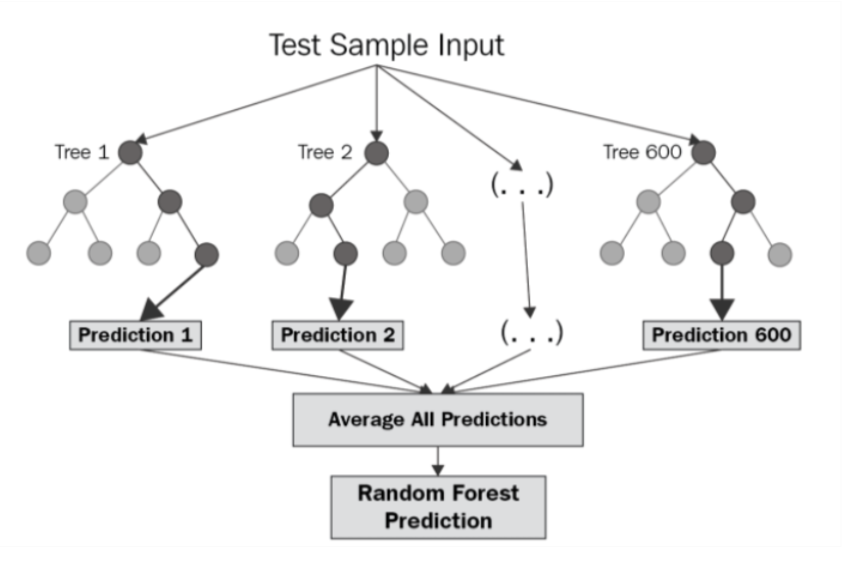
Figure 5. K-Fold Cross Validation

## 2.2.2 Random Forest

**Random Forest Regression** is a supervised learning algorithm that uses **ensemble learning** method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. It is perhaps the most popular and widely used machine learning algorithm given its good or excellent performance across a wide range of classification and regression predictive modelling problems.

It is also easy to use given that it has few key hyperparameters and sensible heuristics for configuring these hyperparameters. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.

A Random Forest Regression model is powerful and accurate. It usually performs great on many problems, including features with non-linear relationships. However, the disadvantages of random forest include the following: there is no interpretability, overfitting may easily occur, we must choose the number of trees to include in the model.



Advantages of Random Forest:

* It gives variable importance which helps in determining the variable which impacts positively.
* Often machine learning models are overfitted, random forest classifiers wouldn't get overfitted.
* It can be used as a regression as well as classification model.
* It takes care of null values
* It can automatically balance data sets when a class is more infrequent than other classes in the data.
* The method also handles variables fast, making it suitable for complicated tasks.

Disadvantages of Random Forest:

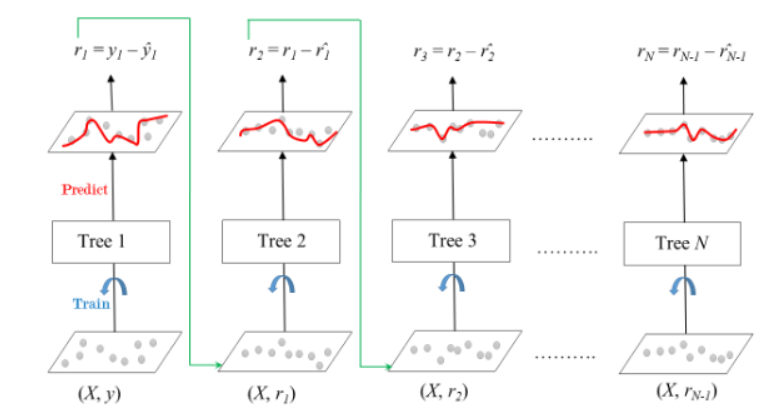
* The main limitation of random forest is that a large number of trees can make the algorithm too slow and ineffective for real-time predictions.
* In general, these algorithms are fast to train, but quite slow to create predictions once they are trained.
* A more accurate prediction requires more trees, which results in a slower model. In most real-world applications, the random forest algorithm is fast enough but there can certainly be situations where run-time performance is important and other approaches would be preferred.
* Random forest is a predictive modelling tool and not a descriptive tool, meaning if you're looking for a description of the relationships in your data, other approaches would be better.

## 2.2.3 Gradient Boost Regression Model

Gradient boosting is a boosting technique, consisting of two sub-terms, gradient and boosting. Gradient boosting re-defines boosting as a numerical optimisation problem where the objective is to minimise the loss function of the model by adding weak learners using gradient descent. Gradient descent is a first-order iterative optimisation algorithm for finding a local minimum of a differentiable function. As gradient boosting is based on minimising a loss function, different types of loss functions can be used resulting in a flexible technique that can be applied to regression, multi-class classification, etc.

Intuitively, gradient boosting is a stage-wise additive model that generates learners during the learning process (i.e., trees are added one at a time, and existing trees in the model are not changed). The contribution of the weak learner to the ensemble is based on the gradient descent optimisation process. The calculated contribution of each tree is based on minimising the overall error of the strong learner.

Gradient boosting does not modify the sample distribution as weak learners train on the remaining residual errors of a strong learner (i.e, pseudo-residuals). By training on the residuals of the model, this is an alternative means to give more importance to misclassified observations. Intuitively, new weak learners are being added to concentrate on the areas where the existing learners are performing poorly. The contribution of each weak learner to the final prediction is based on a gradient optimisation process to minimise the overall error of the strong learner.



The ensemble consists of *N* trees. Tree1 is trained using the feature matrix *X* and the labels *y*. The predictions labelled *y1(hat)* are used to determine the training set residual errors *r1*. Tree2 is then trained using the feature matrix *X* and the residual errors *r1* of Tree1 as labels. The predicted results *r1(hat)* are then used to determine the residual *r2*. The process is repeated until all the *N* trees forming the ensemble are trained.

Each tree predicts a label and final prediction is given by the formula,

y(pred) = y1 + (eta \* r1) + (eta \* r2) + ....... + (eta \* rN)

## 2.2.4 XG Boost Regression Model

XG Boost is a powerful approach for building supervised regression models. The validity of this statement can be inferred by knowing about its (XGBoost) objective function and base learners.

The objective function contains loss function and a regularization term. It tells about the difference between actual values and predicted values, i.e., how far the model results are from the real values. The most common loss functions in XGBoost for regression problems is reg: linear, and that for binary classification is reg: logistics. Ensemble learning involves training and combining individual models (known as base learners) to get a single prediction, and XGBoost is one of the ensemble learning methods. XGBoost expects to have the base learners which are uniformly bad at the remainder so that when all the predictions are combined, bad predictions cancel out and better one sums up to form final good predictions. XGB minimises a regularised objective function that merges a convex loss function, which is based on the variation between the target outputs and the predicted outputs. The training then proceeds iteratively, adding new trees with the capability to predict the residuals as well as errors of prior trees that are then coupled with the previous trees to make the final prediction.

Advantages of XG Boost Library:

* XGB consists of a number of hyper-parameters that can be tuned — a primary advantage over gradient boosting machines.
* XG Boost has an in-built capability to handle missing values.
* It provides various intuitive features, such as parallelisation, distributed computing, cache optimisation, and more.

Disadvantages of XG Boost Library:

* Like any other boosting method, XGB is sensitive to outliers.
* Unlike LightGBM, in XGB, one has to manually create dummy variable/ label encoding for categorical features before feeding them into the models.

# Experimental Results

From this analysis, we found that there was no correlation between app features like size, rating, number of installs, and price and even between price and ratings. There was a strong negative correlation between the number of installs and the number of reviews. Most numbers of apps belonged under genres of tools, Entertainment, Education, Business, and Medical. On the base of the number of installs, we divided the apps into two categories: successful and unsuccessful. Decision Tree gave the highest accuracy percentage of 95.32% and the Gaussian Naive Bayes model gave the lowest accuracy of 88.45%.

The decision tree worked well because we had a simple model and we had a really important feature to take the decision which was the number of installs. On the other hand, we got the lowest accuracy using the Gaussian Naive Bayes model because it has strong feature independence assumptions. In Naive Bayes, the assumption that all features are independent is not usually the case in real

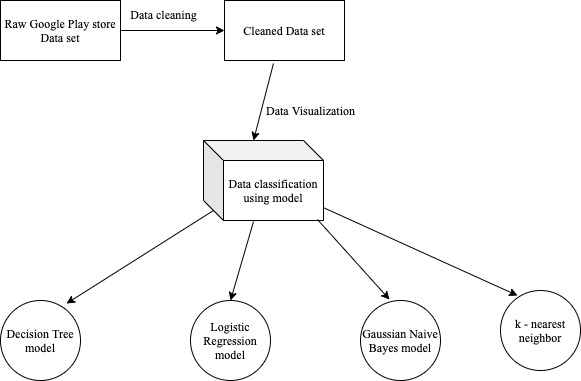


Figure 8. The overall architecture for App analysis

life so it makes it less accurate [11]. There were total 33 different categories in our data set We also found that the average rating for all the apps was 4.0 which means that most people rated if they like the apps. The apps with most importance belonged to categories of ”Entertainment” and

”AUTO AND VEHICLE”.

# Conclusion and future work

This data set contains a large amount of data that can be used for various purposes. Currently, the decision tree model made using this data set can be used for future developers and Google plays store team to glance at the google play store market and what categories of the apps should be made to keep google play store popular in the future. It can be used to improve business values and google play store in general. It is not just limited to the problem we solved.

Using this data set, we applied various classification algorithms and found that Decision tress fits best for our problem statement. We also discovered how different algorithms work in different cases. We found that the Decision tree is easy to visualize and explain the model implementation and it also saves computational power. Using this data set the future work includes the prediction of other parameters such as the number of reviews and installs based on the regression model, identifying the categories and statistics of the most installed apps, exploring the correlation between the size of the app and its version of Android, etc on the number of installs.

# Task Assignment and Acknowledgement

This project was completed by Rimsha Maredia under the supervision and instruction from Dr. Zhangyang (Atlas) Wang and Zhenyu Wu. The dataset was used from the Kaggle data store.

# References

1. Kaggle.com.(2018). Google Play Store Apps.[online] https://www.kaggle.com/lava18/google-play-storeapps [Accessed 3 Mar. 2020].
2. “Mining and Analysis of Apps in Google Play,” Proceedings of the 9th International Conference on Web Information Systems and Technologies, 2013.
3. Google play store: number of apps

2018(2018). [online] https://www.statista.com/ statistics/266210/number-of-available-applications-inthe-google-play-store/ [Accessed 3 Mar. 2020].

1. Amit Chile, Dr. P. R. Gundalwar.(2019). Analysis of Google Play Store Application.[online] http://ijraset.com/fileserve.php?FID=24134 [Accessed 3 Mar. 2020]
2. T. Denoeux, M. Skarstein-Bjanger, Induction of decision trees from partially classified data, in: Proceedings of the 2000 IEEE International Conference on Systems, Man and Cybernetics (SMC’00), IEEE, Nashville, TN, 2000, pp. 2923–2928.
3. Harman, M., Jia, Y., and Zhang, Y. (2012). App store mining and analysis: Msr for app stores. In 2012 9th IEEE Working Conference on Mining Software Repositories (MSR),pages 108–111.
4. R. P. Rajeswari, K. Juliet, and Aradhana, “Text

Classification for Student Data Set using Naive Bayes Classifier and KNN Classifier,” Int. J. Comput. Trends Technol., vol. 43, no. 1, pp. 8–12, 2017.

https://doi.org/10.14445/22312803/ijctt-v43p103

1. Jong, J. (2011). Predicting rating with sentiment analysis. [online] http://cs229.stanford. edu/proj2011/JongPredictingRatingwithSentimentAnalysis.pdf.
2. [2015].Grover, S. 3 apps that failed (and what they teach us about app marketing). [online] https://blog.placeit.net/apps-fail-teach-us-appmarketing/.
3. H. G. Schnack, M. Nieuwenhuis, N. E. van Haren, L. Abramovic, T. W. Scheewe, R. M. Brouwer, H. E. Hulshoff Pol, and R. S. Kahn, “Can structural MRI aid in clinical classification? A machine learning study in two independent samples of patients with schizophrenia, bipolar disorder and healthy subjects,” NeuroImage, vol. 84, pp. 299–306, jan 2014.