SUCCESS PREDICTION OF AN APP IN PLAYSTORE

“THIS PROJECT IS DEDICATED TO ALL ASPIRING AND HARDWORKING MOBILE ANDROID APP DEVELOPERS AND TO THE DEVELOPER COMMUNITY OF ANY CORPORATION”

Acknowledgements and Preface

"I would like to express my thanks to the people who have helped me most throughout my project. I am grateful to my mentor Dr.Gnanavel Mutharasu for his nonstop support or attention who inspired me to successfully complete the project. A special thank of mine goes to my colleagues and peer group members who helped me out in completing the project, where they all exchanged their own interesting ideas, thoughts and made this possible to complete my project with all good results. At last but not the least I want to thank my friends who treasured me for my hard work and encouraged me and finally to God who made all the things possible for me till the end."

"Success Prediction of an app in playstore" is a data science project in which we predict some result for future enhancement of the process of making an app or to make a better progress from the start, to make it profitable.

This project was selected to be the most essential domain to work on because, being an android developer, I have developed many projects and published in playstore, but I don't get the expected revenue because, I don't meet the demand in the market. Also, my work goes unrecognised and unrewarded. This disappointment in me, gave the inspiration to make a prediction pipeline which predicts the success of any app in playstore, so that it could meet the demand in market. This system is expected to help and enhance the projects of every mobile app developers.

Concerning to go through a complete lifecycle of a data science project, this process took over a month to complete, which gave me a good time developing my skills on data science and adopting the lifecycle of a complete data science project. I felt completely engaged in the process, as I got to figure out a lot of stuffs which were out of my boundaries and there were many areas that I should focus particularly on the domain to improvise the project.

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 and highest accuracy. This report also gives detailed information on the data cleaning, data visualization and exploratory analysis done on this data set. Our project code can be found at ***https://github.com/VARUN7111/Success_prediction_of_app****.*

Contents

1. Introduction 10
2. Analysis Methodology 11
   2.1. Data cleaning 11
   2.2. Visualization and Exploratory Analysis of data 12
   2.3. Regression Models 13
      1.3.1 Cat Boost Regression Model 13
      1.3.2 Random Forest 14
      1.3.3 Gradient Boost Regression Model 16
      1.3.4 XG Boost Regression Model 17
3. Conclusion and future work 18
4. Task Assignment and Acknowledgement 19
   References 19

List of Figures

Figure 1: Histogram of apps available for download by category.

Figure 2: Pie-chart to define the percentage of free apps and paid apps.

Figure 3: Correlation matrix.

Figure 4: Bar chart – Average downloads vs categories.

Figure 5: Scatter plot – Average downloads vs size.

Figure 6: k-fold cross validation.

Figure 7: Structure of random forest regressor.

Figure 8: Hierarchy of Gradient boost regressor.

Figure 9: Architecture diagram

List of Tables

Table 1: Results of all algorithms.

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

Varun
NORDIC ACADEMY

1. 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 98.3% 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.

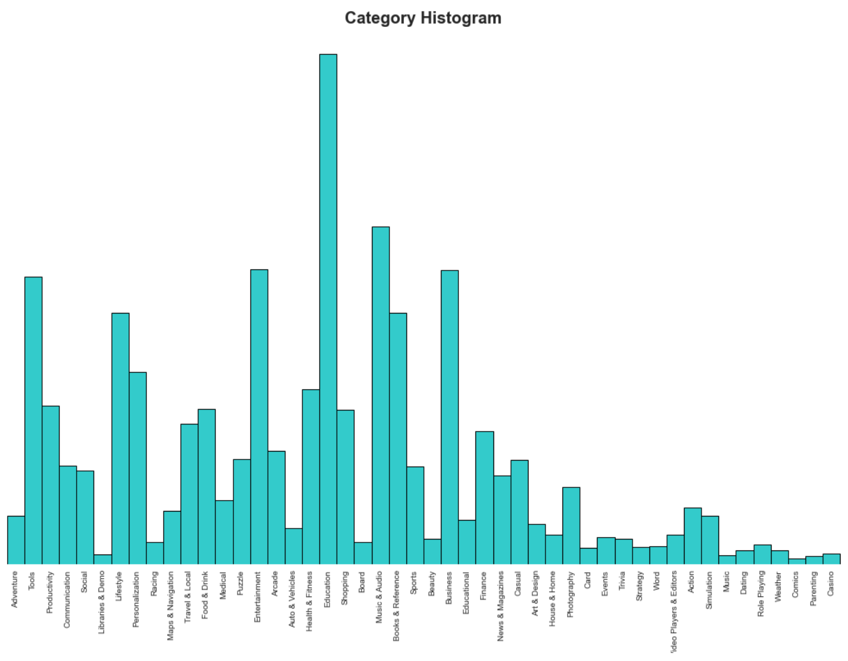
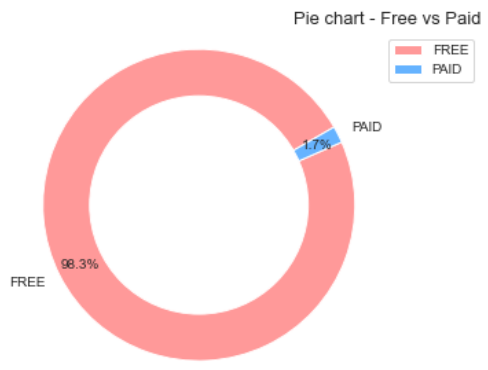


Figure 1. Histogram of apps available for download by category.

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 stuffs that are not needed in 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.

2. Analysis Methodology

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



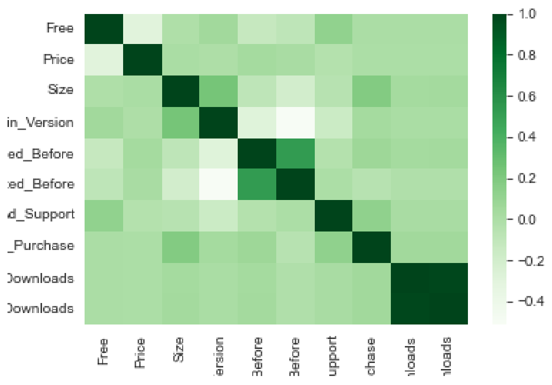


Figure: 2

Figure: 3

Figure 2. Pie-chart to define the percentage of free apps and paid apps.
Figure 3. Correlation matrix.

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 cleaning

We start by scraping our data directly from the kaggle API due to size constraints. After loading it, we start off by renaming all the columns for our convenience and then dropping all the rows where currency is not in terms of USD. Also, as the column containing the size of the app was not uniform (some apps being in megabytes, while some being in GB) so, we made that column uniform by converting all the app sizes into megabytes. Next, we format the "Released" and "Last Updated" column and calculate the number of months between the release date and the last updated date from the current date.

After this, other common processes like handling all the null and missing values from the dataset, changing the necessary datatypes etc. Our target variable being "Average Downloads" was calculated by finding the average of minimum downloads and maximum downloads.

2.2. Visualization and Exploratory Analysis of data

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 1.7% of paid apps in Google play store.
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. Comparing the histogram of categories (fig: 1) and the bar chart b/w Categories and average downloads (fig: 4), we can infer that the categories with the least frequency in fig: 1 has the abnormal number with average downloads in fig: 4.Hence, if these categories with least frequency(less than 1 percent of total data) are taken out, then we can expect a normal bar graph between categories and average downloads.

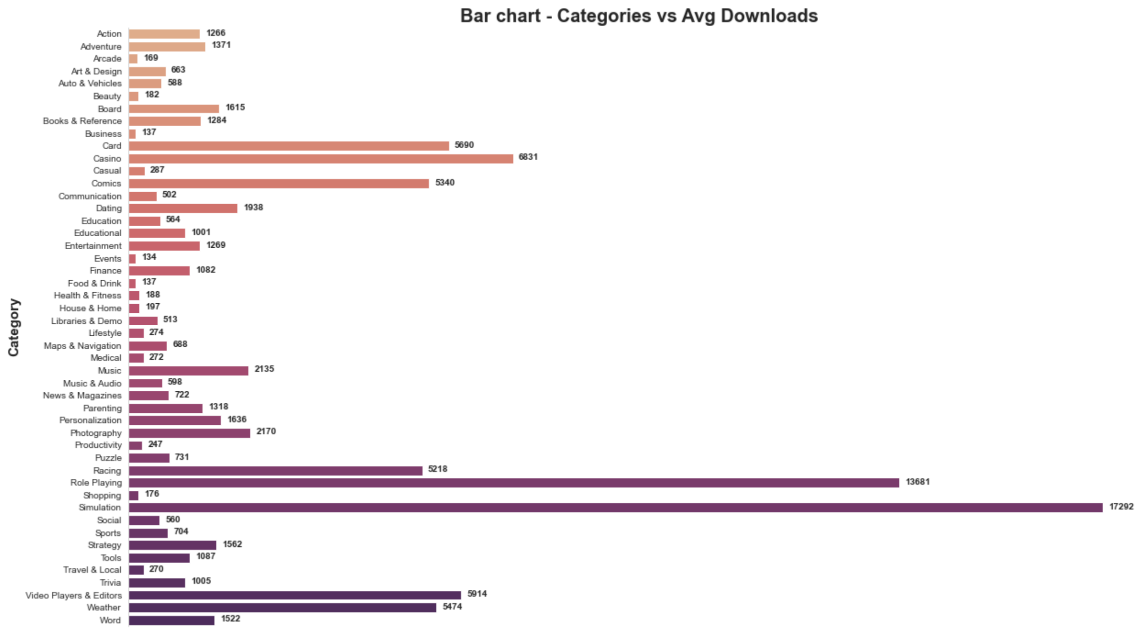


Figure: 4

Looking at the scatter plot between Average downloads and Size, we can infer that the apps with size less than 200MB has the greater chance of getting higher no of downloads and more than 200MB is considerably low.

Finally considering the distribution of every feature in dataset, they were highly skewed and the values are diminished in the right side due to its high frequency of numbers close to zero. This type of data required log 10 transformation to be in normal distribution. After that also it resulted with some skewedness, which was due to the presence of outliers. After the elimination of outliers the graph was close to which were expected.
Even after logarithmic transformation, the scale was having a difference between them. So, we scaled the whole dataset between 0 and 1 using min-max scaler. After all, the data looked pretty much ready for modelling and prediction.

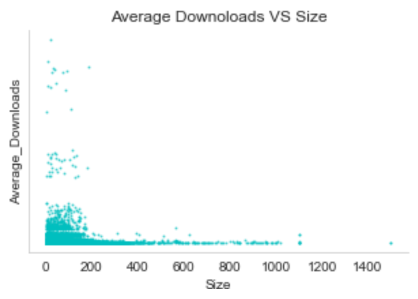


Figure: 5

2.3. Regression Models

1.3.1 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. We got the least mean absolute error of 11.9 percent which was bought down to 11.7 percent through manual hyper parameter tuning.

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*".

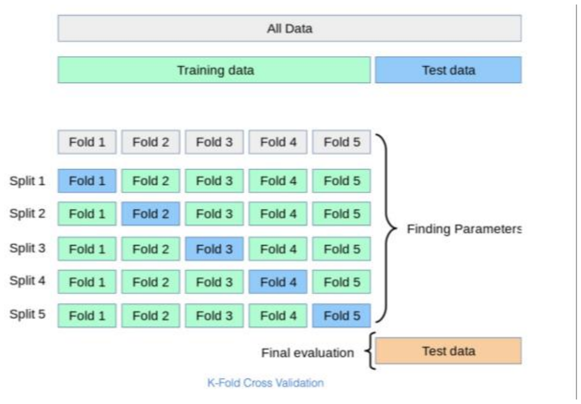


Figure 6. K-Fold Cross Validation

1.3.2 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 hyper parameters and sensible heuristics for configuring these hyper parameters. 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. The error we got for this model was a bit higher- 12.6 %, which was also not a bad score.

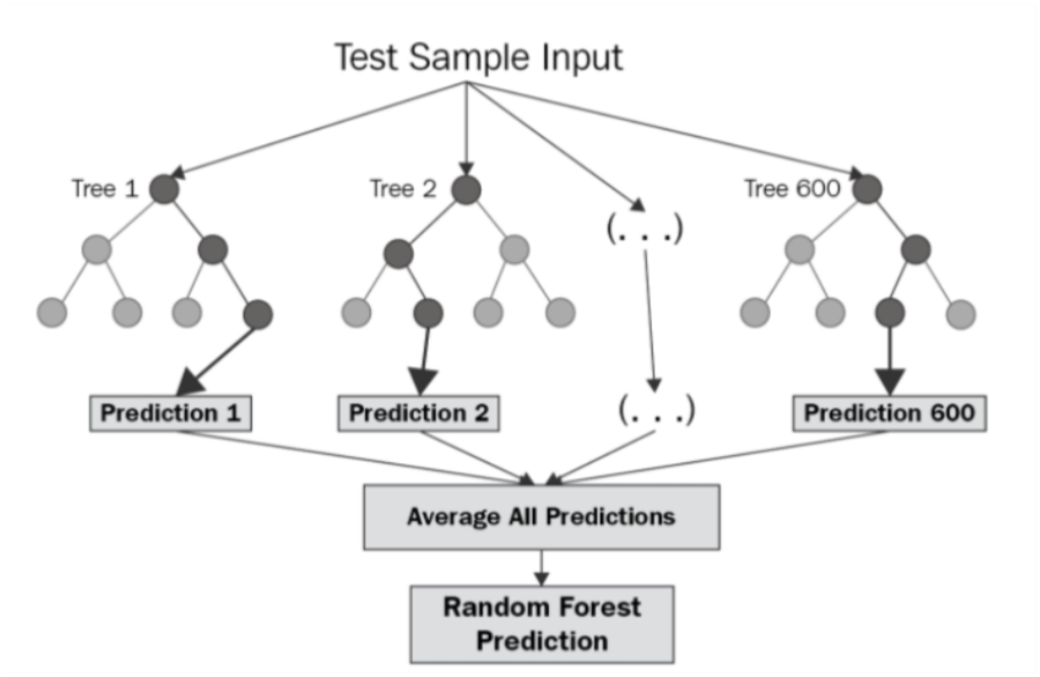


Figure 7: Structure of random forest

Advantages of Random Forest:

- It gives variable importance which helps in determining the variable which impacts positively.
- Often machine learning models are over fitted, random forest classifiers wouldn't get over fitted.
- 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.

1.3.3 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.

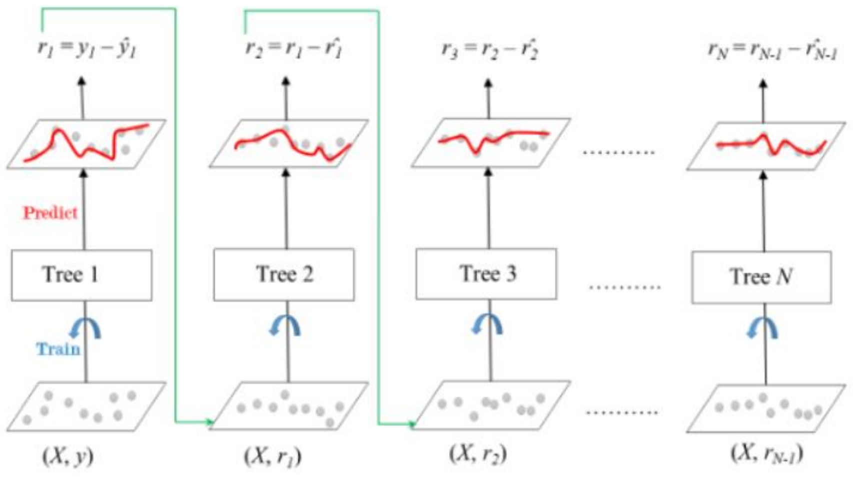


Figure 8: hierarchy of Gradient boost regressor

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,

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

On training this model with our data, gave a score of 12.8% which was lesser than the before.

1.3.4 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. This model on training with the dataset gave a very good result of only 12% error. We will also have to remember that the boosting algorithms are aggressive in nature.

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.

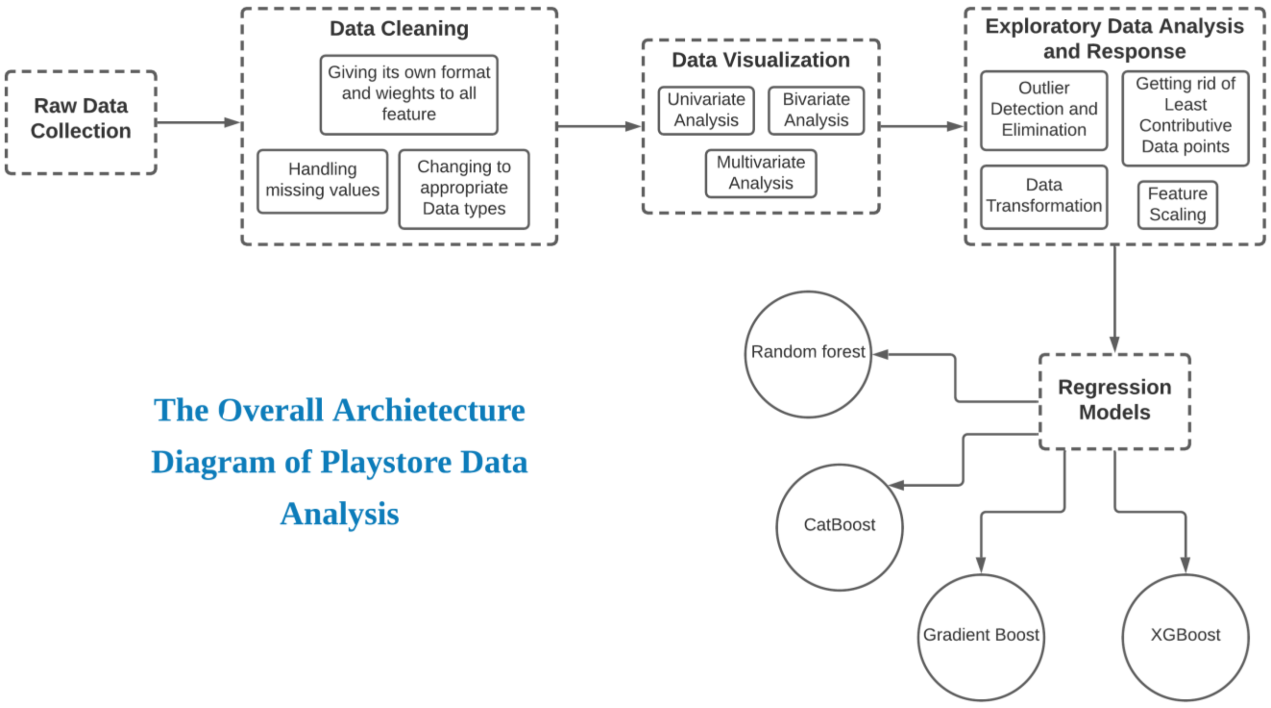

Raw Data Collection
Data Cleaning
Giving its own format and wieghts to all feature
Handling missing values
Changing to appropriate Data types
Data Visualization
Univariate Analysis
Bivariate Analysis
Multivariate Analysis
Exploratory Data Analysis and Response
Outlier Detection and Elimination
Getting rid of Least Contributive Data points
Data Transformation
Feature Scaling
Regression Models
Random forest
CatBoost
Gradient Boost
XGBoost
The Overall Archietecture Diagram of Playstore Data Analysis

Figure 9: Architecture Diagram

3. Conclusion and future work

This data set contains a large amount of data that can be used for various purposes. Currently, the Cat boost 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.

Table 1. Results of all algorithms.

Algorithm	Mean Absolute Error %
Cat boost	11.9%
XG-boost	12.0%
Random forest	12.6%
Gradient boost	12.8%

Using this data set, we applied various regression algorithms and found that Cat boost fits best for our problem statement. We also discovered how different algorithms work in different cases. We found that the cat boost 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 ratings and installs based on the classification 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 Minimum number of installs.

4. Task Assignment and Acknowledgement

This project was completed by Nithish Kumar, Aditi Mittal and Varun under the supervision and instruction from Dr. Gnanavel Muthurasu. The dataset was used from the Kaggle data store.

References

[1] Kaggle.com (2021). Google Play Store Apps. [Online] https://www.kaggle.com/gauthamp10/google-playstore-apps [Accessed Sept. 2021].

[2] Hands-On Machine Learning with Scikit-Learn - Aurelien Geron.

[3] Python Data Science Handbook – VanderPlas