A Exploratory Study of Google Play Store dataset and analysing its features

Report By:-

Sarath Patlolla (764816172)

Karthik Sama (463617949)

Divya Sai Sekhar Mullapudi (755658447)

1. Introduction

1.1 Overview

This report discusses and presents the results of the work done in analysing and performing various data mining algorithms on **Google Play Store Dataset,** to explore the various hidden patterns in the dataset.

1.2 Background and Motivation

Google play (previously Android Market) is a digital distribution service operated and developed by Google Inc. It serves as the official app store for the android operating system, allowing users to browse and download applications developed by the Android software development kit (SDK) and published through google. The major categories in which the apps are distributed are Family, Game etc.

Due to the presence of huge number of apps in the play store, analysing those apps and performing various data mining algorithms on them is a challenging task. The Play store apps data has enormous potential to drive app- making business to success. Actionable insights can be drawn for developers to work on and capture the Android market.

1.3 Dataset

The google play store dataset was scrapped from the android market by Lavanya Gupta. The data set has a collection of more than 10,000 play store, and there are 13 features for each app which are App, Category, Rating, Reviews, Size, Installs, Type, Price, Content Rating, Genres, Last Updated, Current Ver, Android Version. The dataset is in csv format.

2. Exploratory Data Analysis

2.1 Pre-processing

The data which was scrapped from the play store and stores is not in clean format. There were many feature values present in the dataset which were missing. During the pre-processing steps all those rows which had missing values were removed in-place. Thus removing the rows which had missing values resulted in the shape of dataset to 9360 * 13.

After the missing field containing rows are removed, integer encoding of all the columns need to perform for implementing various data mining algorithms.

2.1.1 Encoding of columns

The values in the category are nominal attributes and there is a total of 33 categories present in the dataset, therefore all the category names are assigned with a number

For the size column of the dataset, the apps are suffixed by “M” for mega bytes and “K” for kilo bytes. All the apps and their sizes are converted to kilo bytes i.e if the size of the app is in mega bytes it is multiplied by 1024.

The installs columns values are followed by a “+” sign, in the pre-processing steps the “+” sign is removed.

The values in the type column can only take 2 values i.e free or paid, these values are encoded in binary format i.e free are assigned with 0 and paid are assigned with 1.

In the price column of the csv file each price id appended by a “$” sign. The “$” sign from each of the row variable is removed.

The content rating column values are also converted to binaries of 0 and 1.

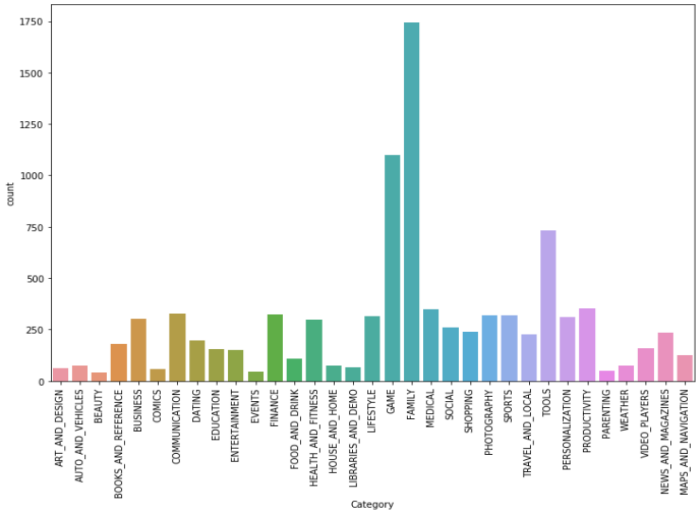
The genres values are also encoded with a number describing the genre value.

2.2 EDA plots

Individual and co-relational plotting of the features is done to perform the exploratory data analysis of the feature values and their distribution in the dataset.

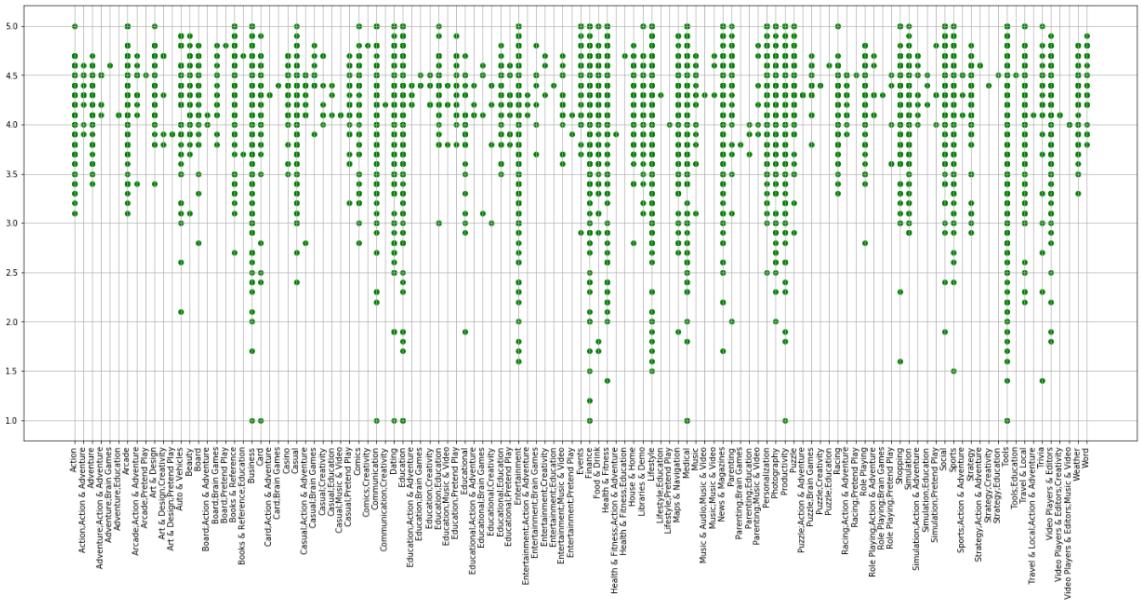
2.2.1 Category v/s Count

In this plot a bar graph is plotted against the categories on the x-axis and the count of each app belonging to a category on the y-axis. The plot is done using matplotlib and seaborn packages in python. Family category has the highest number of apps followed by game and so on.



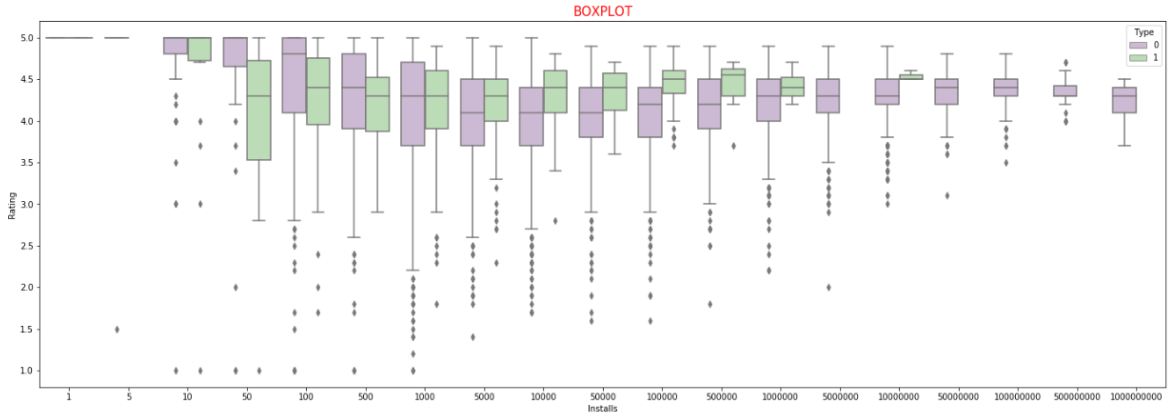
2.2 Genres v/s Rating

A graph is plotted against genres and the rating of a particular app. On x-axis the genres of a particular app is considerd and the rating of that app on y-axis. Each app is plotted accordingly on that graph.



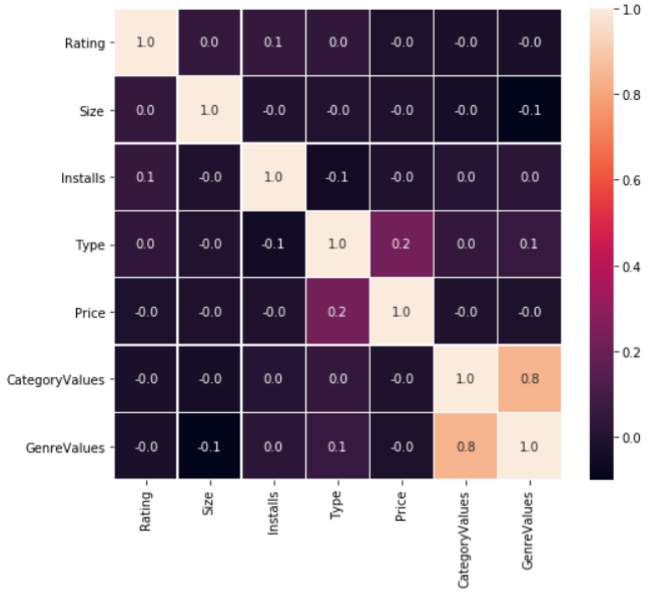
2.3 Rating v/s Installs

A box plot is plotted to represent the distribution of average rating when corrosponding to the number of installs. A box plot has been used tor represent this distribution with 90% values in the box and the remaining 5% are distributed above and below the box. The box is also divided based on type i.e paid or free.



2.4 Heat Map

A heat map is plotted by considering all the features in a square matrix on both x-axis and y-axis and heat map is plotted one co-relation with the other



3. Approach, Results and Related Work

The dataset is split into train data and test data in the ratio of 80:20. Before splitting the dataset into train set and test set the data is shuffled since all similar category apps are present at a single place and shuffling of the dataset supports efficient learning of model those are build.

3.1 Multivariate Regression for predicting App Rating

A multivariate regression for the prediction of app rating model is built. The independent variables for multivariate regression algorithm are Category, Installs, Size, Type, Price, Genre, and the dependent variable for regression model is Rating. Various combinations of features are considered for prediction of app rating, if “category” and “genre” features are not included in the features list, then the RMSE(Root Mean Square Error) of the test data is **11.2372** and if the category value and Genres are

included in the features for predicting of the app rating then the RMSE value for the test data obtained is **0.2453.**

3.2 Neural Network for Classification of the apps based on category

A 3-level deep learning neural network is built for the classification algorithm, The different features to train the neural network model are Installs, Size, Type, Price and Genre. The output labels are 32 Category value classes. If the Genre values are not included in building the model, the training accuracy is very low, If the genre values are included in building the model, then there is a considerable improvement in both train and test accuracy. Thus, we can infer that the genre values play a key role in the better classification of the model.

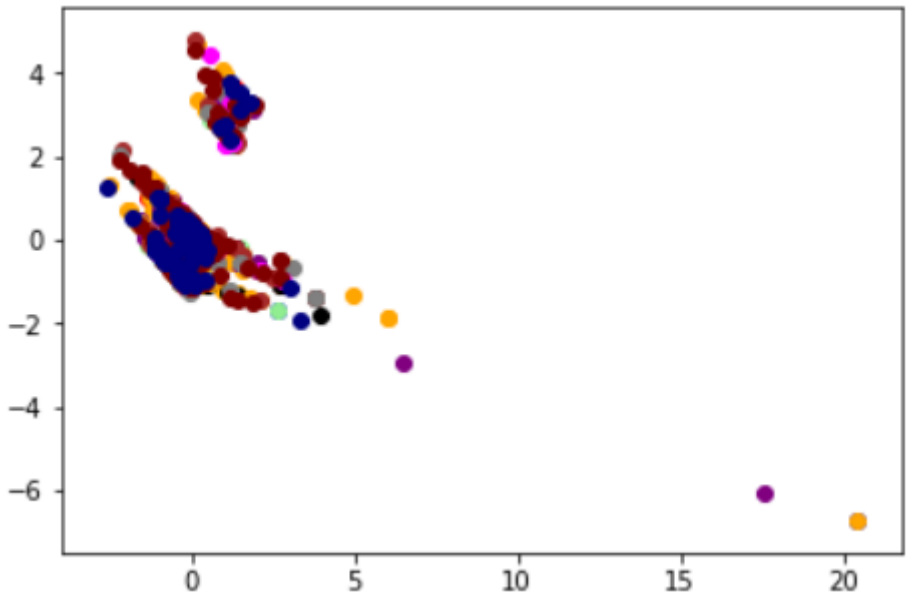
The neural network architecture which was built for this model had 6 neurons whose initial weights were all initialized to 1. Then a score function is computed which is a product of the input vector and the weight matrix after the score function is computed the score function is passed through relu activation function, to impose non-linearity in the learning curve. The score function after the activation function is sent as an input to the hidden layer in the neural network. The hidden layer had 6 neurons and relu is used as an activation function. The final layer is a fully connected layer and had cross entropy loss is computed. The loss is optimized by the adam optimizer in the back-propagation step.

Training Accuracy	45.5%
Test Accuracy	30%

The low value of accuracy was due to the fact that there was no direct co-relation between feature1 of app1 to feature2 of app1. Since all the feature value were independent of each other and change of one value in the feature had no influence on the other.

3.3 K-Means Clustering

An unsupervised clustering algorithm K means clustering is implemented for the category values without taking into consideration the Category values. The initial centroids were fixed by using K-means++ approach. Linear Discriminant Analysis (LDA) is used in to find linear combination of features that characterizes or separates two or more classes of objects and events. The LDA is closely related to regression analysis. The LDA explains categorical variable by the values of continuous independent variable. The clusters are formed by considering components values =2.



The Clusters thus formed are not clear and there is a lot of overlapping of the clusters thus formed and there are some points which belong to the same category but are depicted outside of clusters. A number of clusters analysis has been performed by considering various set of features, but none of the clusters formed was in accordance with the dataset. These irregular clusters were formed since the feature values of the apps were independent of each other and thus using clustering analysis was not satisfactory.

3.4 Naïve Bayes Algorithm based Classification:

Description of Naïve Bayes:

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Statement:

The main purpose the classification is to predict the category the app belongs based on the other features input.

Why this algorithm suits our data set?

Naïve Bayes algorithm works considering the feature parameter as the independent of each other. Here in our data set the features are all independent of each other.

Pre-processing of data:

Syllable Count: Created a new parameter of the based-on app name having number of syllables in it.
Size: convert size of the app into KiB
Price: Convert price to numbers
Review: Segregate review into 12 groups based on number of reviews
Content Rating, Category and Genres are converted to numbers
Selection of feature and that causes change output accuracy score of prediction:

Attributes	Model Accuracy
Rating, Review, Size, Price, Type, Content Rating, Syllables_count, GenreValues	68.53
Rating, Review, Size, Price, Type, Content Rating, Syllables_count,	17.094
Rating, Review, Price, Type, Content Rating, Syllables_count, GenreValues	80.23
Rating, Review, Price, Content Rating, Syllables_count, GenreValues	84.13
Rating, Review, Price, Content Rating, Syllables_count,	4.67
Review, Price, Content Rating, GenreValues	83.1

Analysis and finding:

The table above lists the accuracy of the classification model for different sets of attributes. The inclusion of one particular category namely ‘GenreValues’ drastically increases the accuracy of the model. Whereas the attribute ‘Rating’ does not have any effect on the accuracy. From this we understand that different attributes have different significance in the categorization of the apps.

3.5 Recommendation system for Play Store apps

Motive:

The motive is to build a recommendation system for a cold start and to understand how the recommendations change with change in attributes.

Description:

When a user visits an app in PlayStore, she is recommended similar apps based on the app visited. Based on this idea we have designed a recommendation system using collaborative filtering which recommends apps which are similar to the app visited based on features such as ‘Category, Genre, Paid/Free, Ratings and No. of installs’. These features are taken as attributes to compute similarity with other apps. Then the top most similar apps are returned.

Method:

First the app features in the dataset is pre-processed to make attributes for the system:

- The values in each attribute is checked for uniqueness.
- For continuous attributes like 'Ratings' and ‘No. of Installs’, the values are discretized into intervals.
- For attributes like Category or Genre, the results should be very much similar to the visited app. So the dimensionality of attribute is increased such that there is one column for each unique value of the attribute

Then a utility matrix is formed from these columns and apps as rows. A similarity matrix is formed by computing cosine similarity between each app. The top most similar apps are returned by sorting this list.

Results:

Table 1
Sketch - Draw & Paint
350 Diy Room Decor Ideas
Tattoo Name On My Photo Editor
Sad Poetry Photo Frames 2018
Anime Manga Coloring Book

Table 2
I Creative Idea
Smoke Effect Photo Maker - Smoke Editor
Cardi B Wallpaper
HD Mickey Minnie Wallpapers
2000 AD Comics and Judge Dredd

Table 3
Logo Maker - Small Business
Art Drawing Ideas
Harley Quinn wallpapers HD
HD Mickey Minnie Wallpapers
2000 AD Comics and Judge Dredd

Table 4
Gas Station
Fuelio: Gas log & costs
Best Car Wallpapers
Dictionary - Merriam-Webster
Coloring book moana

Analysis:

The attributes and its values have been tweaked to understand the effect of various changes on the recommendation results. Parameters like number of attributes, dimensionality of each feature, weightage of attributes have been changed.

The results above are recommendations for the app named ‘Sketch - Draw & Paint’.

Decreasing the number of attributes: Table 3 shows the results which have been obtained by removing the attributes like ‘Rating’ and ‘No. of installs’. It can be seen that the recommendations are not very popular apps which the user may not be interested in, though they are similar to the visited app.

Weightage of attributes: Table 1 lists the results obtained when all columns are given equal weights. The apps listed in Table 2 are obtained by increasing the weightage for attributes: ‘Category’ and ‘Genre’ by a factor of 5. It can be seen that the recommendations change drastically. Though recommendations in both cases look relevant, but a closer analysis shows that the apps in table 1 fall in multiple genres like ‘Art & Design’ and ‘photography’, whereas the apps in table 2 fall only under ‘Art & Design’. So it is understood that to get a more tighter fit results, the weightage of the corresponding attributes should be given higher priority.

Dimensionality of each feature: Results in Table 4 are obtained when each feature is considered as a single attribute of continuous values instead of multiple dimensions for each unique value. It can be seen from the results that the recommended apps belong to completely different categories i.e recommendations for a app belonging to ‘Art and Design’ include results from ‘Auto and Vehicles’, ‘Books’ and ‘Beauty’. This happens because the difference in values for each category is significantly small compared to differences in attributes such as ‘Paid/ Free’. So when a paid app is visited, if there are not many paid apps in the similar category, paid apps from other categories which are numbered closer are displayed. This is irrelevant to the user.