Visualizing Playstore App Data

by

Avirup Das (MDS202013)
Prosenjit Dey (MDS202029)
Shiuli Subhra Ghosh (MDS202035)
Suman Roy (MDS202041)

MSc Data Science

A project on scraping, cleaning and visualising the data on play-store apps

Instructor:
Dr Venkatesh Vinayakarao

Lecturer, Chennai Mathematical Institute
venkateshv@cmi.ac.in
vvtesh.co.in

June, 2021

1. Introduction

Apps have transformed the way consumers handle their mobile phones and, as a result, how companies handle their business. Whether using for communication, shopping or transportation applications, consumers today prefer to solve their problems on their mobile phones. Given these new consumer habits, companies need to reinvent themselves to meet their customers' needs and be present in the best way possible. When well designed and developed with user-focused methodologies, an application can be very beneficial to a company's marketing and sales strategies.

Also, there are millions of applications uploaded by the developers on the daily basis. Without any check and balance millions of users download these applications. Theses duplicated applications damage the users trust on Google play store and can grab the confidential information of user. There is no more information provided by developers on the front end of the application that can define the legitimacy of the application.

Often times developers do not have the slightest idea of what the users are actually looking for and thus end up designing something which fails miserably in the market. An 'incompetent-app' can also cause damage to the business of the company. Thus it is important to analyse the trends of apps in order to study the factors determining whether users are going to be satisfied with that app or not.

2. Why this is a Big-Data project?

The Google play store was launched in 2008. Earlier till 2020, the total number of apps on the Google Play Store was 2.87 million which has now risen to nearly 5 million. Fetching, cleaning and analysing the data of all these apps is not an easy job. The scale of the data is estimated to be of a few hundred gigabytes. This amount of data cannot be handled using conventional data handling systems and commercially available devices. Big-Data solutions come to the rescue in these scenarios!

3. Methodology

3.1. **Crawling and Scraping.** We designed a code which will crawl over the Playstore using the links provided and collect all relevant data (Downloads, Size, App permissions, Release Date, Version Support, etc.) The crawler visits each of the links, waits for 5 seconds for the page to load and scrolls down to the bottom of the page. The Google play store doesn't allow its users to view all apps at once, new apps are loaded once the user scrolls down to the bottom of the page. Our bot handles this without breaking a sweat, it will wait for 3 seconds for the new apps to show up and will continue doing this until there are no more apps to show.

At this stage, it has the app-id's of all the apps it has come across and will visit the app-page to collect the relevant data. All this data is then transferred to a CSV file.

3.2. **Data Cleaning.** The data that we have needs a lot of cleaning in order to use it for further use (Visualisation or even building predictive models). We have used Py-Spark in a single node setup to perform all data cleaning tasks.

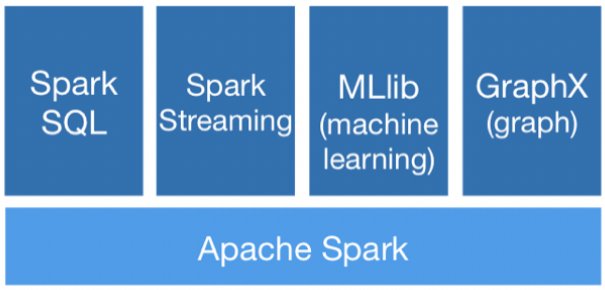
The first task of cleaning would be to read all the csv files, combine them into a single schema and remove duplicate apps. A few apps might appear on multiple search results for various reasons like popularity, Editor's preference or maybe the developer has opted for a paid promotion. Also, play store provides different information for the same app when referenced through different links, so we have to remove duplicates based solely on App-ID.

Next we fix the size of the apps. In reality we need this to be a numerical data but the scraped data comes of as '102.4M' (102.4 megabytes) or '1.2G' (1.2 gigabytes). So we convert the size of all the apps to kilobytes in order to get a continuous data. To execute this using Spark, we write a UDF (User-Defined Function) which takes the size as String input and returns the size in kilobytes. This function is then applied to the 'size'-column.

For the In-App-Purchases column we either get a range of price (₹ 7,000-₹ 9,000 per item) or a single price (₹ 2,500 per item). We need to find the average price and get a continuous variable from this data. Also if there is a missing value, there are no in-app purchases involved and hence the average price is 0. We treat this column in a similar way by writing a UDF and applying it to the column.

The 'Content Rating' column needs a similar treatment. We need to convert inputs like 'Rated for 3+' to 3 (continuous variable) and is done by writing a UDF. Finally for app-permissions we have missing values for those apps which do not require that particular permission. So, we need to fill in 'False' for missing values and 'True' if the description of the permission is present for that app.

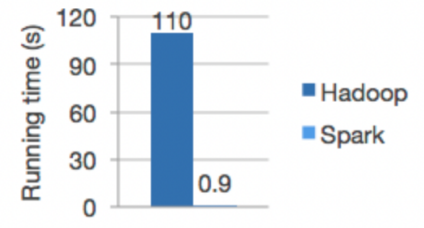
3.3. **Why use Spark?** PySpark is an interface for Apache Spark in Python. It not only allows us to write Spark applications using Python APIs, but also provides the PySpark shell for interactively analyzing the data in a distributed environment. PySpark supports most of Spark's features such as Spark SQL, DataFrame, Streaming, MLlib (Machine Learning) and Spark Core.



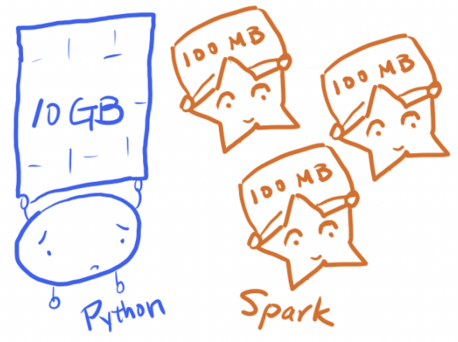
Spark SQL is a Spark module for structured data processing. It provides a popular programming abstraction called DataFrame which allows storage of variables of

multiple data-types in a single unit and can also act as distributed SQL query engine.

Apache Spark achieves high performance for both batch and streaming data resulting in running workloads 100 times faster than Hadoop.



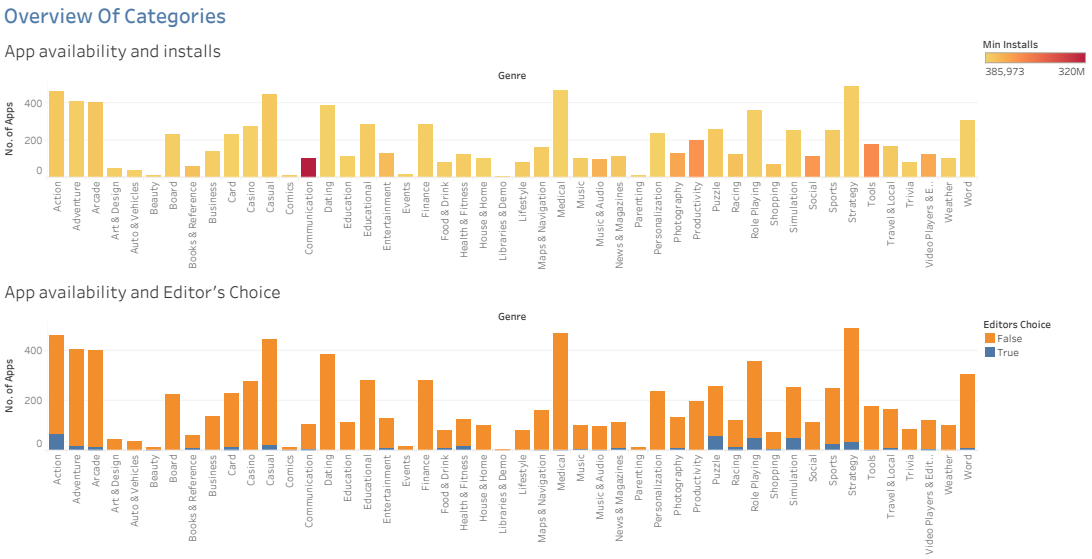
Logistic regression in Hadoop and Spark



The major problem with Python is that it will load the data into the pool all at once and will also perform all operations on the whole data. So when we have a data-set of say 100 GB and the primary memory of the system is 64 GB, even loading this data into a Pandas dataframe is as daring as to prove the Riemann hypothesis! Spark re-partitions the data and picks it up one at a time. This way we can handle huge data even with comparatively less resources.

4. Visualisation

4.1. **Overview of Categories:**



For detailed dashboard, please click here.

4.1.1. *App availability and installs.*

- Here we tried to show an overview of the count of apps in different categories.
- We have also mapped the count of apps with respect to the average minimum installs.
- **Observation:** Communication category has lesser apps but it has maximum average installs. The category where we find more number of apps, no. of installs are eventually distributed among all the available apps. So, we can't visualize significant minimum installs in case of all the game categories.
- We found that there are many sub categories of games which can't be identified from here. It was appearing as different genre.
- So, we created a new calculated filed in Tableau and combined all the games sub categories under "Games".

4.1.2. *App availability and Editor's Choice.*

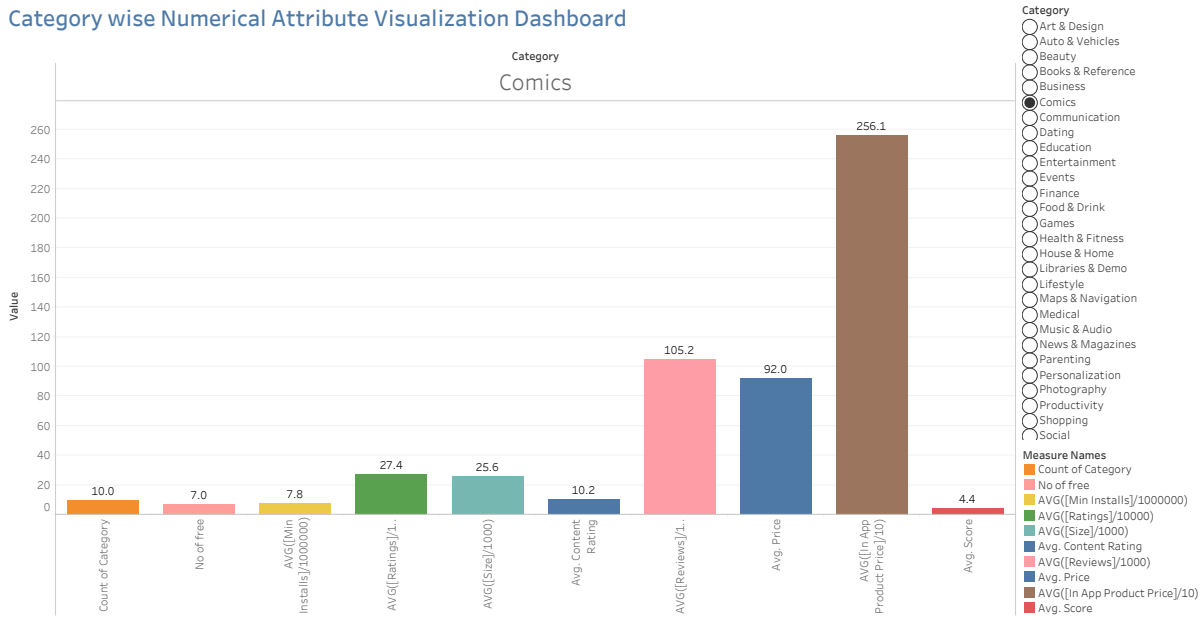
- Here we tried to visualize how many apps in the different categories were under Editor's choice.
- Mostly Game category Apps had the highest editor's preference.
- One table is shown with the Editor's Choice statistics.

Genre	App Count	Editor's Choice(True)
Action	460	64
Puzzle	256	56
Role Playing	358	50
Simulation	252	49
Strategy	488	34

Genre	App Count	Editor's Choice(True)
Sports	252	26
Casual	447	25
Adventure	407	20
Health & Fitness	124	17
Card	231	14
Arcade	400	13
Racing	120	13
Travel	165	10
Food & Drink	82	10

TABLE 1. App Count vs. Editor's Choice

4.2. **Category wise Overview of Data.**

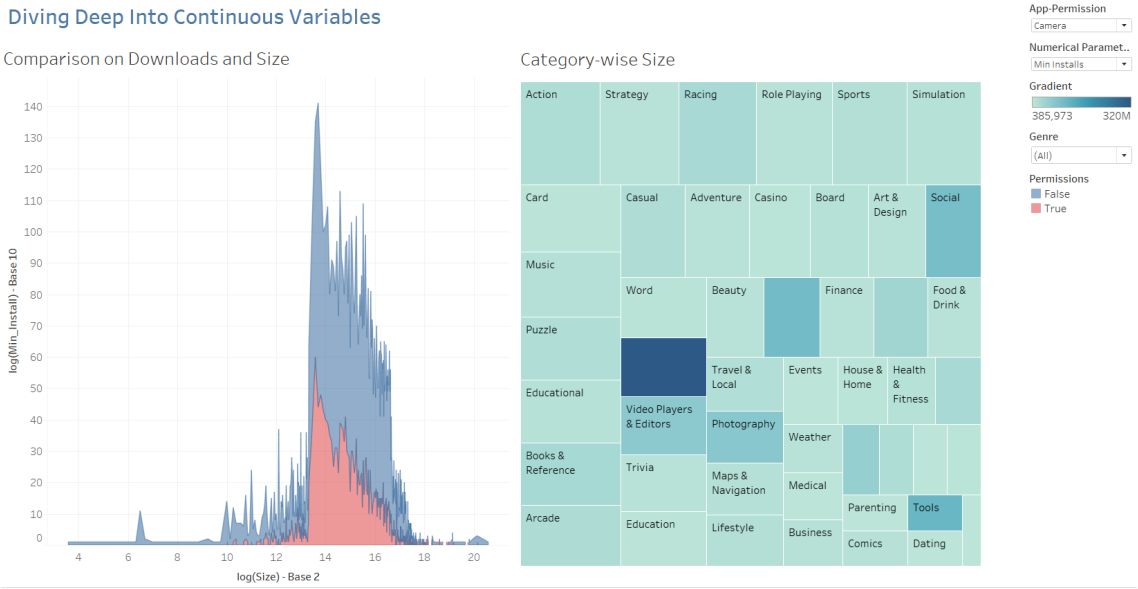


For detailed dashboard, please click here.

- This is an interactive dashboard for category wise numerical attribute description.
- We have shown following attributes below..
  - Count of Category
  - No. of Free Apps
  - Average Min Installs (in Millions)
  - Average Ratings (in Thousands)
  - Average Size (in Thousands)
  - Average Content Rating (+ Years)
  - Average Number of Reviews (in Thousands)
  - Average Price
  - Average In App Product Price (in Hundreds)
  - Average Score

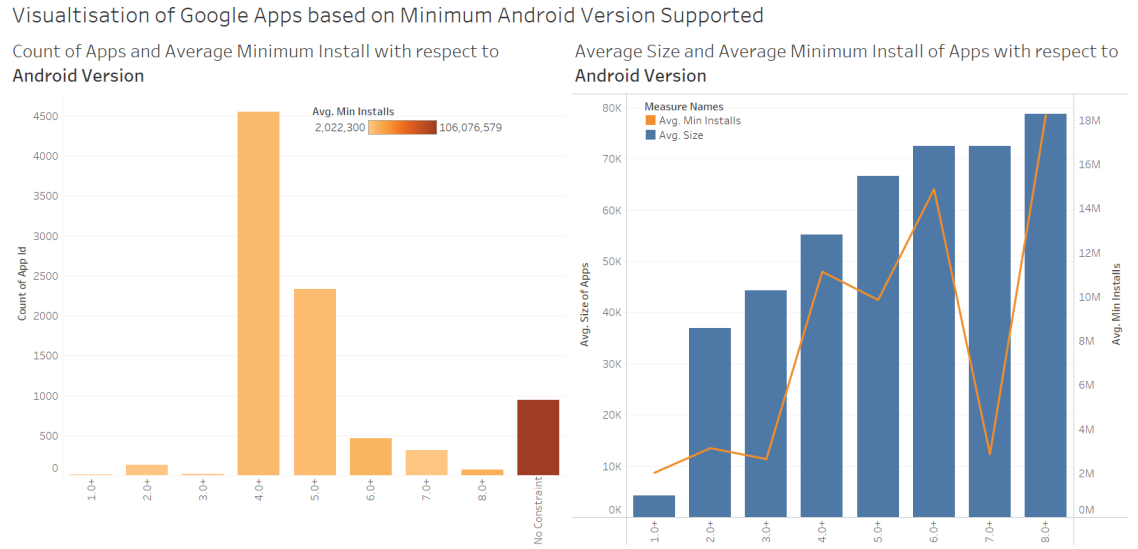
- Minimum installs are in millions range, where content rating or average score is in 10's range. So, we scaled each attributes so that it can be visualized with in a suitable range.
- **Observations:**
  - Google play contains huge number of Gaming apps (Nearly 5000).
  - Beauty and Dating apps are mostly free.
  - Medical category apps are most costly.
  - Dating apps have average content rating of 17+
  - Communication category apps contain maximum reviews and maximum average min installs.

4.3. **Diving Deep into Continuous Variables.**



For detailed dashboard, please click here.

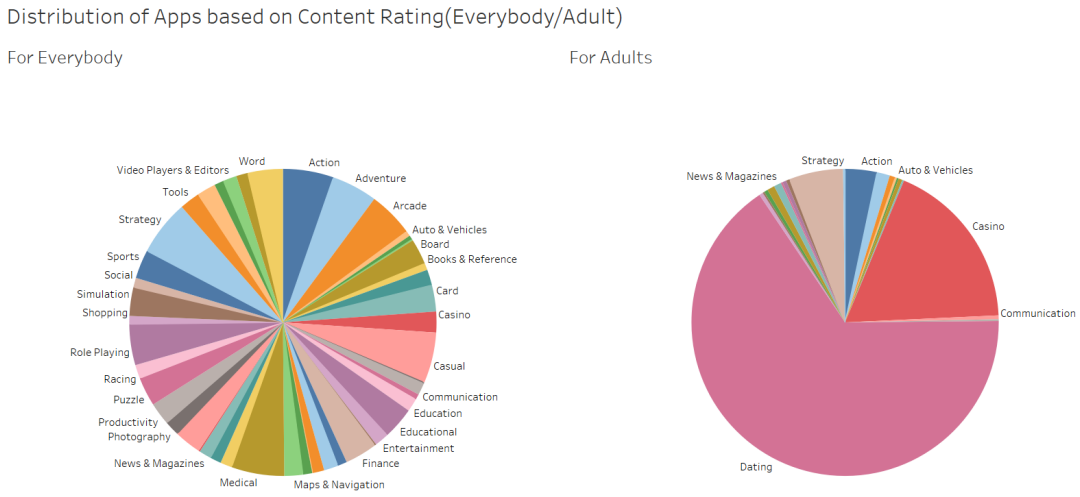
4.4. **Visualization of Google Apps based on Minimum Android Version Supported.**



For detailed dashboard, please click here.

- We have taken count of Apps for different 'Minimum Android Version Supported' attribute. It shows that minimum version of Android 4.0 has most number of apps in its bag. It is to be noted that Android 4.0 (Ice Cream Sandwich) is one of the most major updates of Android till date and it was out in 2011.
- There are also apps with no constraints/undefined as minimum android version supported. We have taken into belief that these apps support all the Android Versions.
- While the average minimum installs of apps for different android version remains more or less same , There is a sharp rise in minimum average installs for apps which have no constraint in Android versions.
- The Average size of the apps have also gone up with the Minimum Android Version supported.

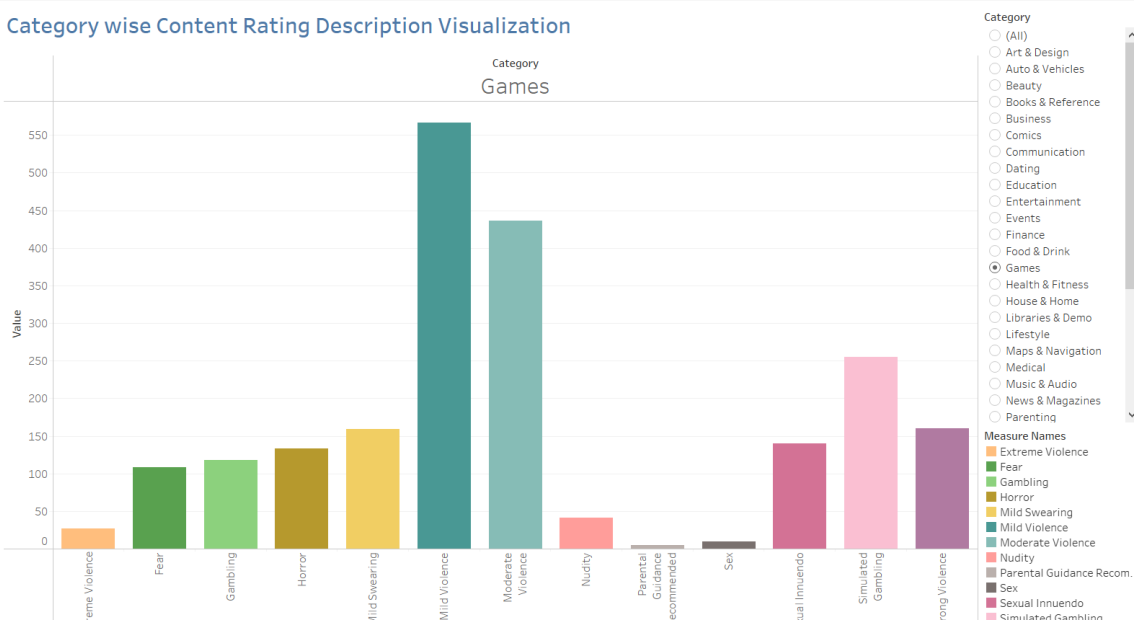
4.5. **Distribution of Apps based on Content Rating.**



For detailed dashboard, please click here.

- The apps in the Google playstore are rated for different ages(like for ages 3+,7+,12+,18+ etc.)
- for this visualisation we have grouped all the other ages except 18+ as "everybody" and 18+ as "Adult".
- While the distribution of genres is more or less even for apps rated for "Everybody", in "Adult" section Mainly three Genres of apps dominate. These are Dating,Casino and Strategy.
- Dating and Casino are rightfully rated for Adult people but there are few genres which have a good amount shares in the Adult category like Strategy or Adventure. With proper modification of the content in the app these genres can have less number of apps in the adult category.

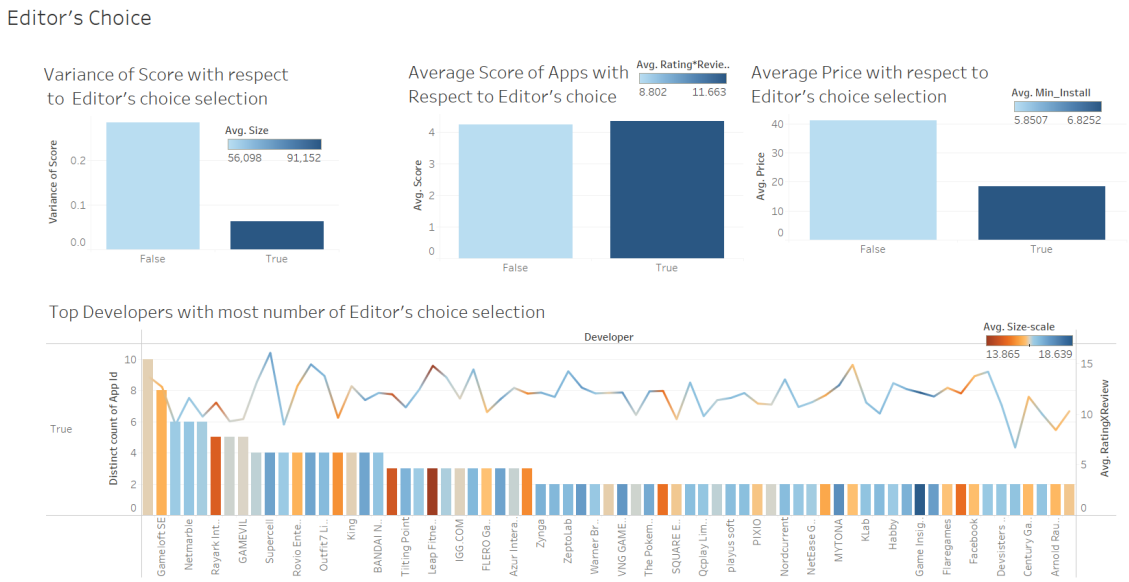
4.6. **Category wise Content Rating Visualization Description.**



For detailed dashboard, please click here.

- This is a dashboard for visualizing Content Rating Description

4.7. **Editor's Choice.**



- A number of the apps are selected as "Editor's Choice" at a specific interval of time determined by Google. Getting selected as "Editor's Choice" means that the App is very popular or it is of great utility.
- The Variance of Score(Rating out of 5) of apps selected as "Editor's Choice" is significantly lower than the apps not selected as "Editor's Choice". It means "Editor's Choice" apps are very consistent with its scores.
- The Average score of "Editor's Choice" apps are also higher than that of not selected as "Editor's choice" apps.
- The Average price of "Editor's Choice" apps lower than that of not selected as "Editor's choice" apps.It means those apps which are more reachable to masses, have a higher probability of picked up in "Editor's Choice".

- The developer named "Electronic Arts" has most number of apps selected in Editor's Choice(10).The developer "Supercell" has most number for rating/reviews.

5. Github Link:

https://github.com/avirupdas55/Big-Data-Visualisation-