ABSTRACT

Google play store is engulfed with a few thousands of new applications regularly with a progressively huge number of designers working freely or on the other hand in a group to make them successful, with the enormous challenge from everywhere throughout the globe. Since most Play Store applications are free, the income model is very obscure and inaccessible regarding how the in-application buys, adverts and memberships add to the achievement of an application. In this way, an application's prosperity is normally dictated by the quantity of installation of the application and the client appraisals that it has gotten over its lifetime instead of the income is created. Application (App) ratings are feedback provided voluntarily by users and function important evaluation criteria for apps. However, these ratings can often be biased due to insufficient or missing votes. Additionally, significant differences are observed between numeric ratings and user reviews. This Project aims to predict the ratings of Google Play Store apps using machine learning Algorithms. We have tried to perform Data Analysis and prediction into the Google Play store application dataset that We have collected from Kaggle. Using Machine Learning Algorithms, We have tried to discover the relationships among various attributes present in my dataset such as which application is free or paid, about the user reviews, rating of the application.

Key Words: Google Play Store Apps, Ratings Prediction, Exploratory Data Analysis, Machine Learning

1.INTRODUCTION

Machine learning approaches are essential for us to take care of numerous issues. In this paper, we present machine learning models and structures in detail. Machine learning has numerous applications innumerous perspectives and has incredible advancement potential.

In future, it is predictable that machine learning could set up ideal speculations to clarify its exhibitions. In the meantime, its capacities of unsupervised learning will be improved since there is much information on the planet however it isn't relevant to add names to every one of them. It is additionally anticipated that neural system structures will turn out to be increasingly unpredictable with the goal that they can separate all the more semantically important highlights. In addition, profound learning will consolidate with support adapting better and we can utilize this points of interest to achieve more assignments.

1.1 Analysis and Prediction

In today's scenario we can see that mobile apps playing an important role in any individual's life. It has been seen that the development of the mobile application advertise has an incredible effect on advanced innovation. Having said that, with the consistently developing versatile application showcase there is additionally an eminent ascent of portable application designers inevitably bringing about high as can be income by the worldwide portable application industry.

With enormous challenge from everywhere throughout the globe, it is basic for a designer to realize that he is continuing in the right heading. To hold this income and their place in the market the application designers may need to figure out how to stick into their present position. The Google Play Store is observed to be the biggest application platform. It has been seen that in spite of the fact that it creates more than two-fold the downloads than the Apple App Store yet makes just a large portion of the cash contrasted with the App Store. In this way, I scratched information from the Play Store to direct our examination on it.

With the fast development of advanced cells, portable applications (Mobile Apps) have turned out to be basic pieces of our lives. Be that as it may, it is troublesome for us to follow along the fact and to understand everything about the apps as new applications are entering market each day. It is accounted for that Android1market achieved a large portion of a million applications in September 2011. Starting at now, 0.675 million Android applications are accessible on Google Play App Store. Such a lot of applications are by all accounts an extraordinary open door for clients to purchase from a wide determination extend. We trust versatile application clients consider online application surveys as a noteworthy impact for paid applications. It is trying for a potential client to peruse all the literary remarks and rating to settle on a choice. Additionally, application engineers experience issues in discovering how to improve the application execution dependent on generally speaking evaluations alone and would profit by understanding a huge number of printed remarks.

1.2 Google Play store Dataset

The dataset consists of Google play store application and is taken from Kaggle, which is the world's largest community for data scientists to explore, analyze and share data.

This dataset is for Web scratched information of 10k Play Store applications to analyze the market of android. Here itis a downloaded dataset which a user can use to examine the Android market of different use of classifications music, camera etc. With the assistance of this, client can predict see whether any given application will get lower or higher rating level. This dataset can be more over used for future references for the proposal of any application. Additionally, the disconnected dataset is picked so as to choose the estimate exactly as online data gets revived all around a great part of the time. With the assistance of this dataset. I will examine various qualities like rating, free or paid and so forth utilizing Hive and after that I will likewise do forecast of various traits like client surveys, rating etc.

1.3 Data Mining

Data mining is that the process of rummaging through a knowledge set and finding correlations, anomalies and or patterns which will be of usefulness. In other words, it's having an outsized dataset filled with scattered information and trying to form sense of it by finding meaningfulness.

1.4 Python

Most of the info scientist use python due to the good built-in library functions and therefore the decent community. Python now has 70,000 libraries. Python is simplest programing language to select up compared to other language. That’s the most reason data scientists use python more often, for machine learning and data processing data analyst want to use some language which is straightforward to use. That’s one among the most reasons to use python. Specifically, for data scientist the foremost popular data inbuilt open-source library is named panda. As we've seen earlier in our previous assignment once we got to plot scatterplot, heat maps, graphs, 3-dimensional data python built-in library comes very helpful.

1.5 Machine Learning

Machine learning is an application of AI (AI) that gives systems the power to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the event of computer programs which will access data and use it learn for themselves.

1.5.1 Supervised Learning

It is defined as a learning in which we train a machine as per our dataset or input. From that point forward, the machine is furnished with another arrangement of examples (data) so supervised learning analyses the provided data (set of preparing models) and creates a right result from given input.

1.5.2 Unsupervised Learning

In the Unsupervised learning we do not train our machine according to the present data or input. It means there is not any supervisor as a teacher in this learning. In this we allow algorithm to work on their own without any training or guidance. Here the main working of the machine is that it works on some definite patterns, similarities in the given dataset without any training or proper guidance. Therefore, machine is restricted to find out the structure which is hidden in the given dataset.

2.LITERATURE REVIEW

There has been a constant growth in the public and private information stored within the internet. This includes textual data expressing people's opinions on review sites, forums, blogs, and other social media platforms. Review based prediction systems allow this unstructured information to be automatically transformed into structured data reflecting public opinion. These structured data can be used subsequently as a measure of users' sentiments about specific applications, products, services, and brands. They can hence provide important information for product and services refinement. This kind of sentiment analysis was conducted in the following studies.

- Kumari and other researchers used the Naïve Bayes (NB) classifier to classify opinions as positive, negative, or neutral.

- Wang and others argued that a rating is not entirely determined by a review content. For example, a user may well intend to give a positive review by employing positive words, and yet issue a comparatively lower rating.

-Dave and others proposed a method for extracting the polarity in user reviews of products, expressed as poor, mixed, or good. The classifier used was Naïve Bayes (NB).

- A recent study investigated the application of a machine learning algorithm to a dataset covering, for example, the app category, the numbers of reviews and downloads, the size, type, and Android version of an app, and the content rating, to predict a Google app ranking. Decision trees, linear regression, logistic regression, support-vector machine, NB classifiers, k-means clustering, k-nearest neighbours, and artificial neural networks were studied for that purpose.

- App ratings have been predicted based on the features provided for app. Experiments were performed on the BlackBerry World and Samsung Android stores to collect the raw features provided for the apps, including their price, rank of downloads, ratings, and textual descriptions. The features were then encoded into a numerical vector to be used in case-based reasoning and to predict the app rating.

However, the above studies are unsatisfactory in various respects and are unsuitable for predicting numeric ratings of Google apps. First, text-mining techniques are ineffective when applied to app reviews, as it has Unicode supported language with a limited number of words. Second, those studies are based either on rating predictions made using inherent app features or on external features (e.g. price, bug report, etc.). None of those studies investigated the possible discrepancies between users' numeric ratings and reviews. To our knowledge, this study is the first to investigate such discrepancies and to base numeric-rating predictions for Google apps.

3.PROBLEM STATEMENT

3.1 Objective

The Main objective of our project is to predict the ratings of Google Play Store apps using machine learning algorithms**.**

3.2 Abstract of the project

Google play store is engulfed with a few thousands of new applications regularly with a progressively huge number of designers working freely or on the other hand in a group to make them successful, with the enormous challenge from everywhere throughout the globe. Since most Play Store applications are free, the income model is very obscure and inaccessible regarding how the in-application buys, adverts and memberships add to the achievement of an application. In this way, an application's prosperity is normally dictated by the quantity of installation of the application and the client appraisals that it has gotten over its lifetime instead of the income is created. Application (App) ratings are feedback provided voluntarily by users and function important evaluation criteria for apps. However, these ratings can often be biased due to insufficient or missing votes. Additionally, significant differences are observed between numeric ratings and user reviews. This Study aims to predict the ratings of Google Play Store apps using machine learning Algorithms. I have tried to perform Data Analysis and prediction into the Google Play store application dataset that I have collected from Kaggle. Using Machine Learning Algorithms, I have tried to discover the relationships among various attributes present in my dataset such as which application is free or paid, about the user reviews, rating of the application.

4. PROGRAMMING LANGUAGE USED

Python 3.8

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. The version of python used in this project is 3.8 64-bit. A lot many packages were used too for the completion of the said project.

Figure 4.1 Python

5.TECHNOLOGY AND TOOLS

5.1 Jupiter Notebook

The Jupiter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupiter Notebook is maintained by the people at Project Jupyter.

Jupiter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupiter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupiter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Figure 5.1 Jupyter Notebook Logo

5.2 Kaggle

Kaggle, a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

Kaggle got its start in 2010 by offering machine learning competitions and now also offers a public data platform, a cloud-based workbench for data science, and Artificial Intelligence education. Its key personnel were Anthony Gold bloom and Jeremy Howard. Nicholas Gruen was founding chair succeeded by Max Levchin. Equity was raised in 2011 valuing the company at $25.2 million. On 8 March 2017, Google announced that they were acquiring Kaggle.

5.3 Pandas

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

Pandas is one of the tools in Machine Learning which is used for data cleaning and analysis. It has features which are used for exploring, cleaning, transforming and visualizing from data.

Figure 5.2 Pandas Library Logo

5.4 NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

5.5 Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Before we start using scikit-learn latest release, we require the following −

- Python (>=3.5)
- NumPy (>= 1.11.0)
- Scipy (>= 0.17.0)li
- Joblib (>= 0.11)
- Matplotlib (>= 1.5.1) is required for Sklearn plotting capabilities.
- Pandas (>= 0.18.0) is required for some of the scikit-learn examples using data structure and analysis.

Figure 5.3 Sklearn Logo

5.6 Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Figure 5.4 Matplotlib Library Logo

5.7 Seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

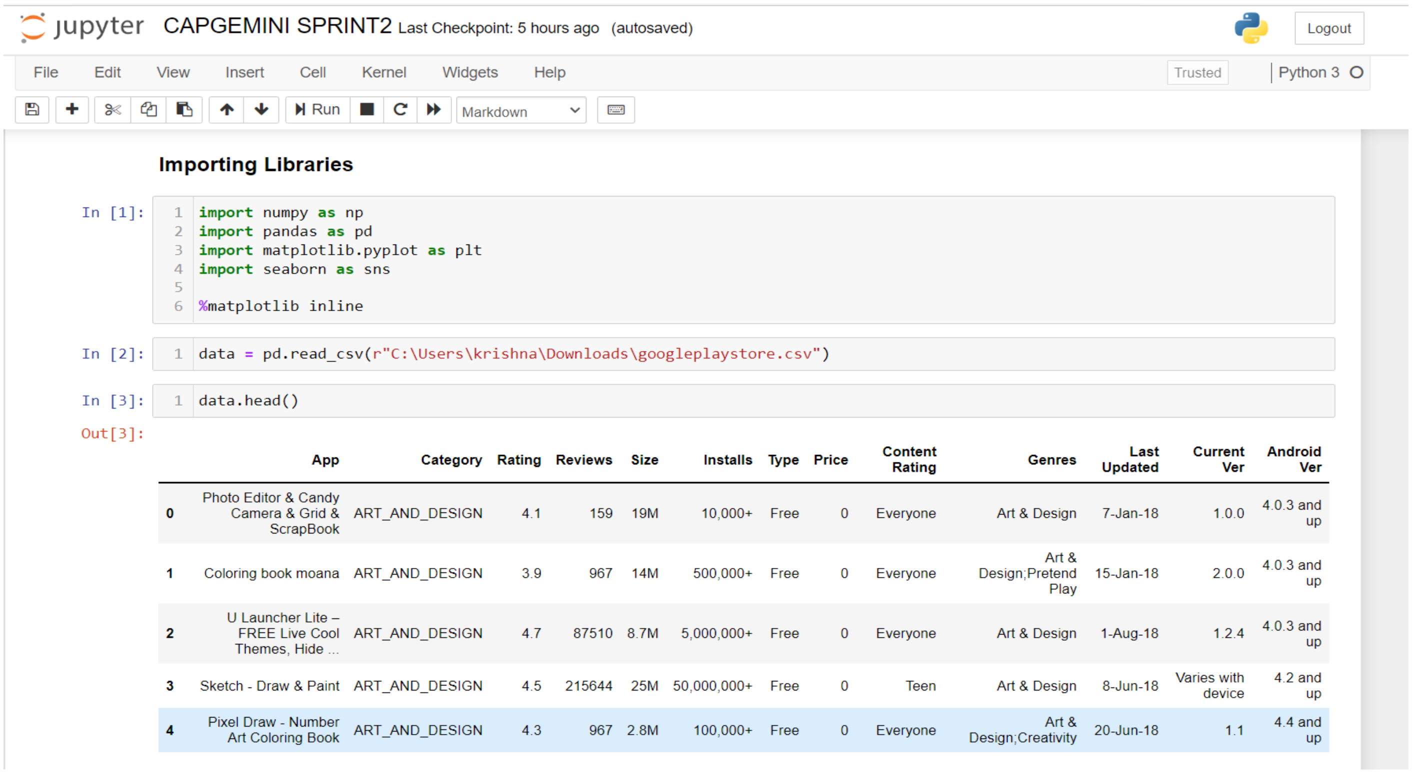
Important Features of Seaborn

Seaborn is built on top of Python’s core visualization library Matplotlib. It is meant to serve as a complement, and not a replacement. However, Seaborn comes with some very important features. Let us see a few of them here. The features help in −

- Built in themes for styling matplotlib graphics
- Visualizing univariate and bivariate data
- Fitting in and visualizing linear regression model
- Seaborn works well with NumPy and Pandas data structures
- It comes with built in themes for styling Matplotlib graphics

Figure 5.5 Seaborn Logo

6. IMPLEMENTATION


jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

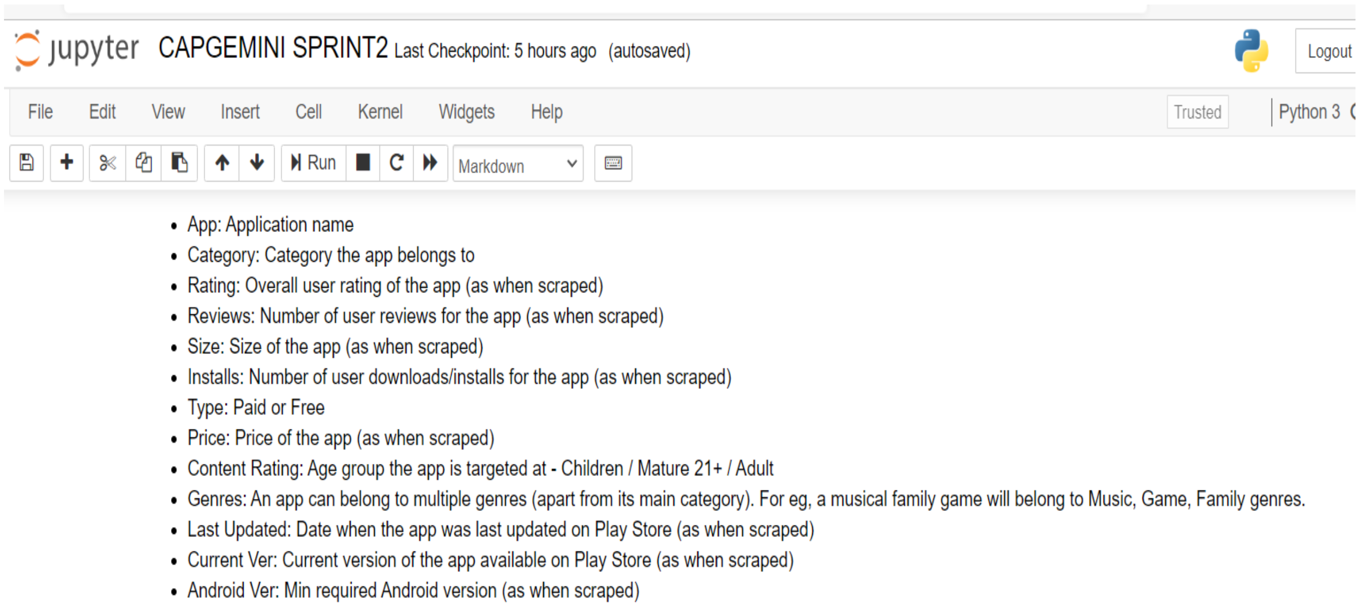
In [2]:
```
data = pd.read_csv(r"C:\Users\krishna\Downloads\googleplaystore.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	7-Jan-18	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	15-Jan-18	2.0.0	4.0.3 and up
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	1-Aug-18	1.2.4	4.0.3 and up
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	8-Jun-18	Varies with device	4.2 and up
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	20-Jun-18	1.1	4.4 and up

Attributes of the Dataset:


jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

- App: Application name
- Category: Category the app belongs to
- Rating: Overall user rating of the app (as when scraped)
- Reviews: Number of user reviews for the app (as when scraped)
- Size: Size of the app (as when scraped)
- Installs: Number of user downloads/installs for the app (as when scraped)
- Type: Paid or Free
- Price: Price of the app (as when scraped)
- Content Rating: Age group the app is targeted at - Children / Mature 21+ / Adult
- Genres: An app can belong to multiple genres (apart from its main category). For eg, a musical family game will belong to Music, Game, Family genres.
- Last Updated: Date when the app was last updated on Play Store (as when scraped)
- Current Ver: Current version of the app available on Play Store (as when scraped)
- Android Ver: Min required Android version (as when scraped)

6.1 DATA PREPROCESSING

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

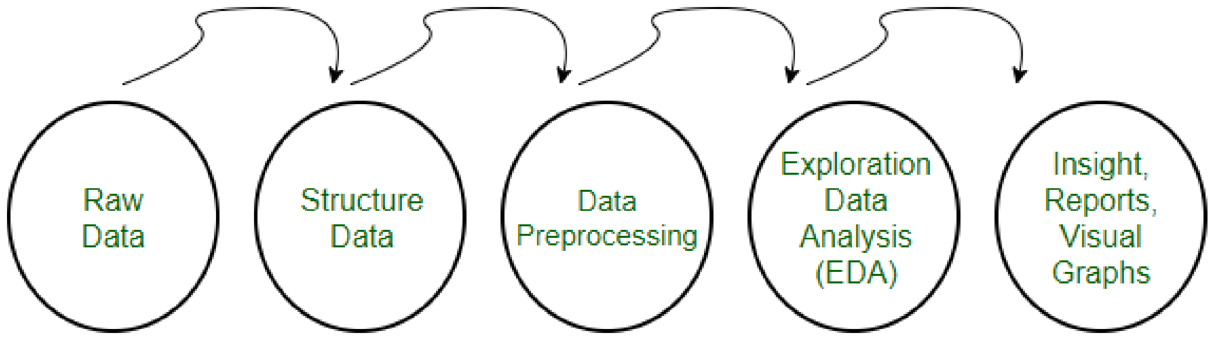
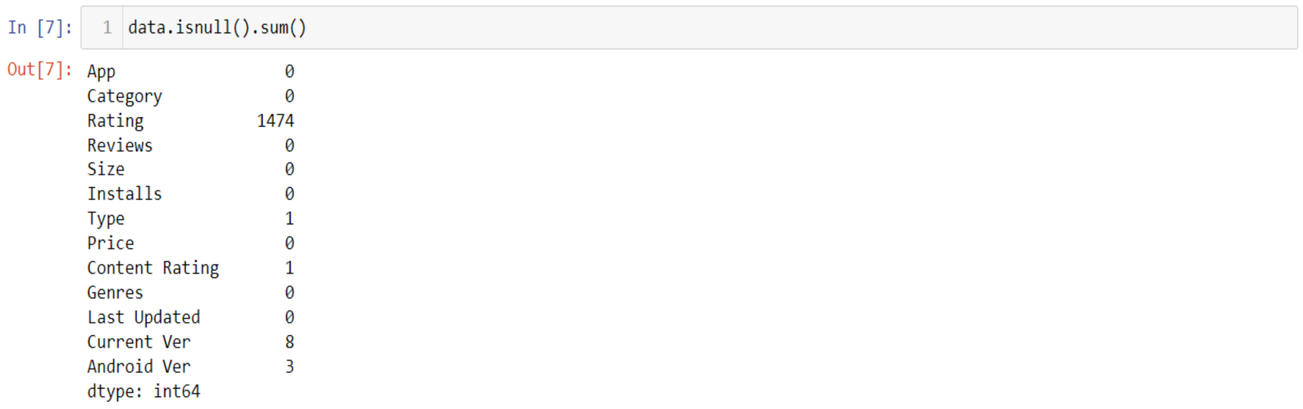
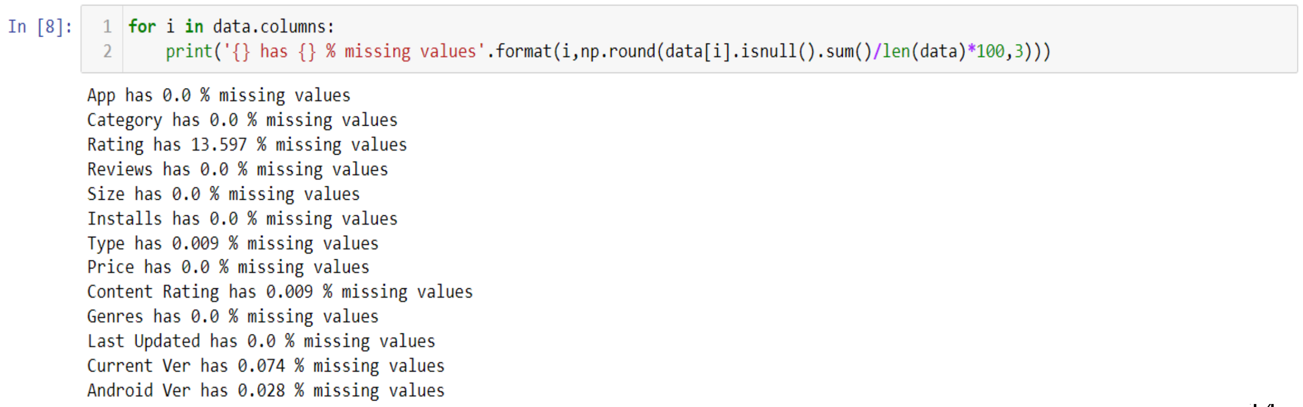

Raw Data
Structure Data
Data Preprocessing
Exploration Data Analysis (EDA)
Insight, Reports, Visual Graphs

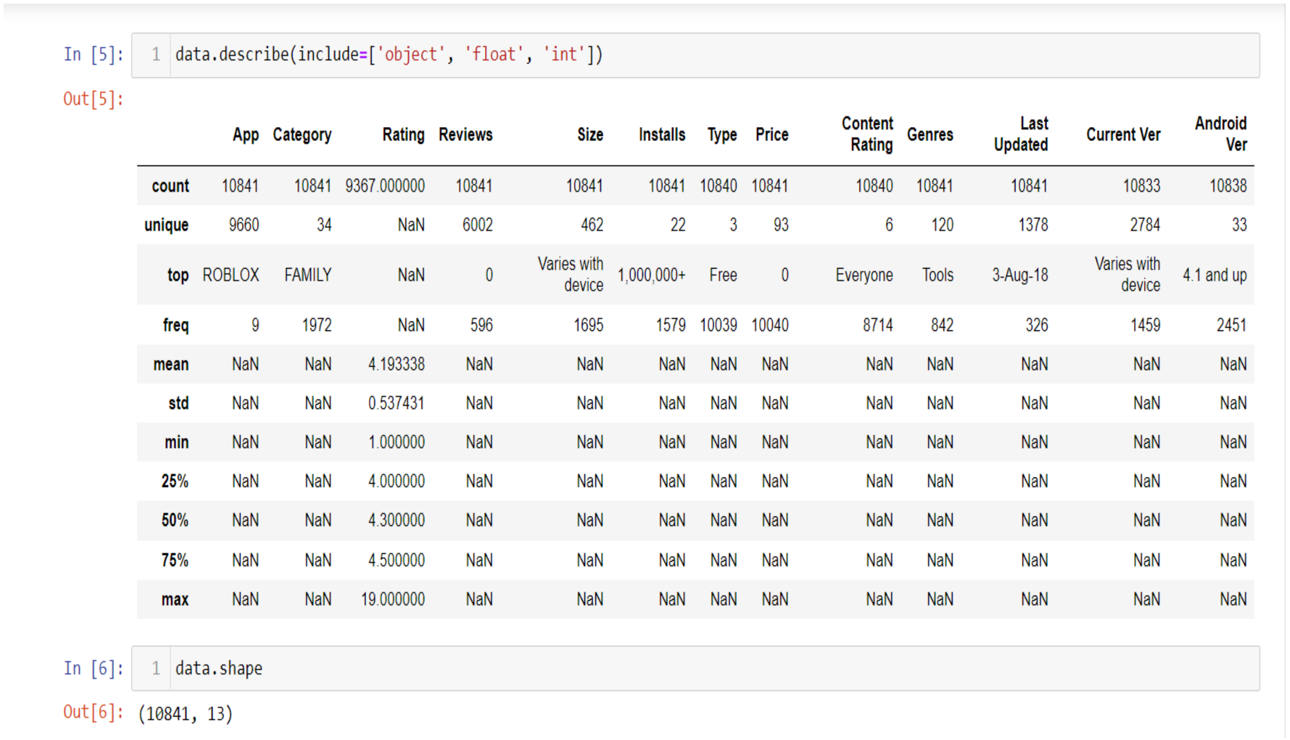
Figure 6.1 Steps involved in ML project cycle



```
In [7]:  data.isnull().sum()

Out[7]: App                  0
        Category             0
        Rating            1474
        Reviews              0
        Size                 0
        Installs             0
        Type                 1
        Price                0
        Content Rating       1
        Genres               0
        Last Updated         0
        Current Ver          8
        Android Ver          3
        dtype: int64
```



```
In [8]:  for i in data.columns:
             print('{} has {} % missing values'.format(i,np.round(data[i].isnull().sum()/len(data)*100,3)))

        App has 0.0 % missing values
        Category has 0.0 % missing values
        Rating has 13.597 % missing values
        Reviews has 0.0 % missing values
        Size has 0.0 % missing values
        Installs has 0.0 % missing values
        Type has 0.009 % missing values
        Price has 0.0 % missing values
        Content Rating has 0.009 % missing values
        Genres has 0.0 % missing values
        Last Updated has 0.0 % missing values
        Current Ver has 0.074 % missing values
        Android Ver has 0.028 % missing values
```



In [5]:
```
data.describe(include=['object', 'float', 'int'])
```

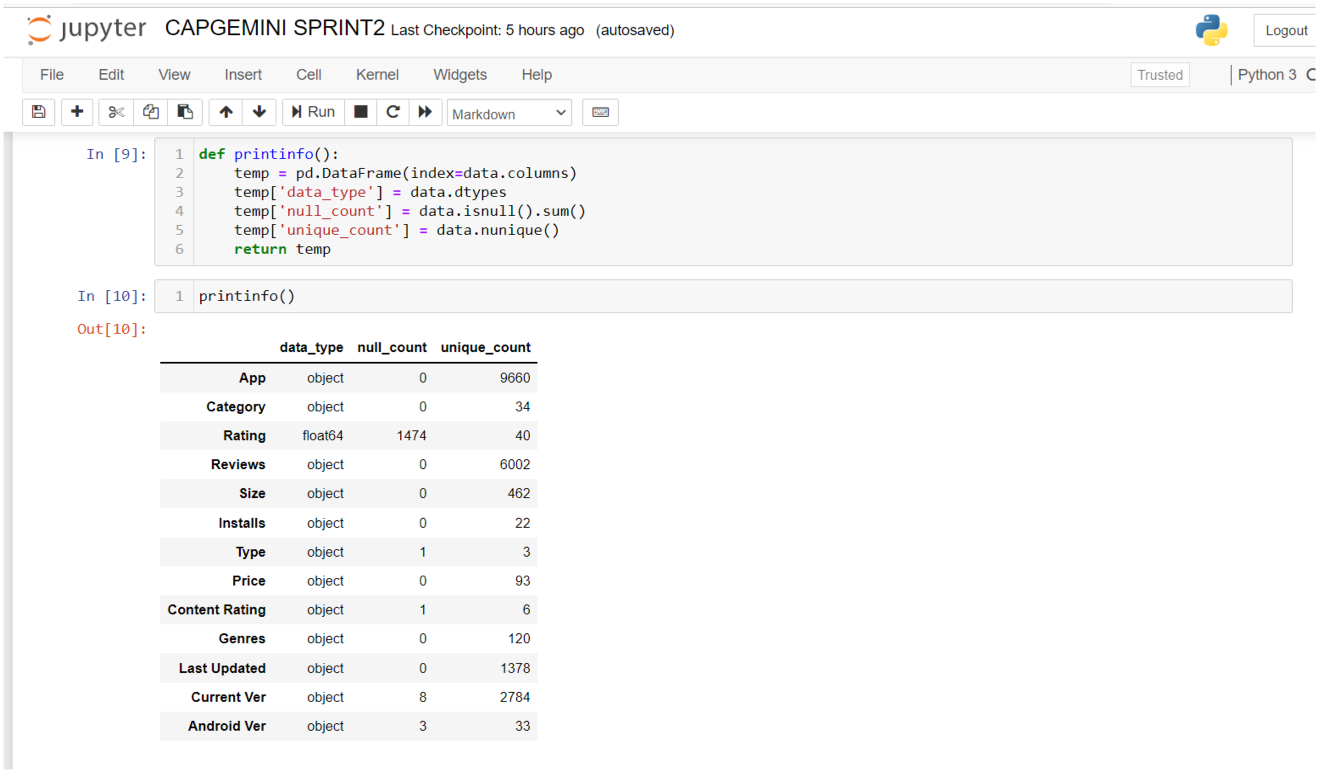
Out[5]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
count	10841	10841	9367.000000	10841	10841	10841	10840	10841	10840	10841	10841	10833	10838
unique	9660	34	NaN	6002	462	22	3	93	6	120	1378	2784	33
top	ROBLOX	FAMILY	NaN	0	Varies with device	1,000,000+	Free	0	Everyone	Tools	3-Aug-18	Varies with device	4.1 and up
freq	9	1972	NaN	596	1695	1579	10039	10040	8714	842	326	1459	2451
mean	NaN	NaN	4.193338	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	0.537431	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	4.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	4.300000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	4.500000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	19.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [6]:
```
data.shape
```

Out[6]:
```
(10841, 13)
```

The dataset contains 10841 rows and 13 columns



jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

In [9]:
```
def printinfo():
    temp = pd.DataFrame(index=data.columns)
    temp['data_type'] = data.dtypes
    temp['null_count'] = data.isnull().sum()
    temp['unique_count'] = data.nunique()
    return temp
```

In [10]:
```
printinfo()
```

Out[10]:

	data_type	null_count	unique_count
App	object	0	9660
Category	object	0	34
Rating	float64	1474	40
Reviews	object	0	6002
Size	object	0	462
Installs	object	0	22
Type	object	1	3
Price	object	0	93
Content Rating	object	1	6
Genres	object	0	120
Last Updated	object	0	1378
Current Ver	object	8	2784
Android Ver	object	3	33

We have some useful information about the dataset. i.e., we can now see the missing number of values of any attribute, its unique count, and its respective data types.

6.1.1 DATA CLEANING:

Data cleaning is one of the important parts of machine learning. It plays a significant part in building a model. It surely isn't the fanciest part of machine learning and at the same time, there aren't any hidden tricks or secrets to uncover. However, the success or failure of a project relies on proper data cleaning. Professional data scientists usually invest a very large portion of their time in this step because of the belief that **"Better data beats fancier algorithms"**.

If we have a well-cleaned dataset, there are chances that we can get achieve good results with simple algorithms also, which can prove very beneficial at times especially in terms of computation when the dataset size is large.

Obviously, different types of data will require different types of cleaning. However, this systematic approach can always serve as a good starting point.

Steps involved in Data Cleaning:

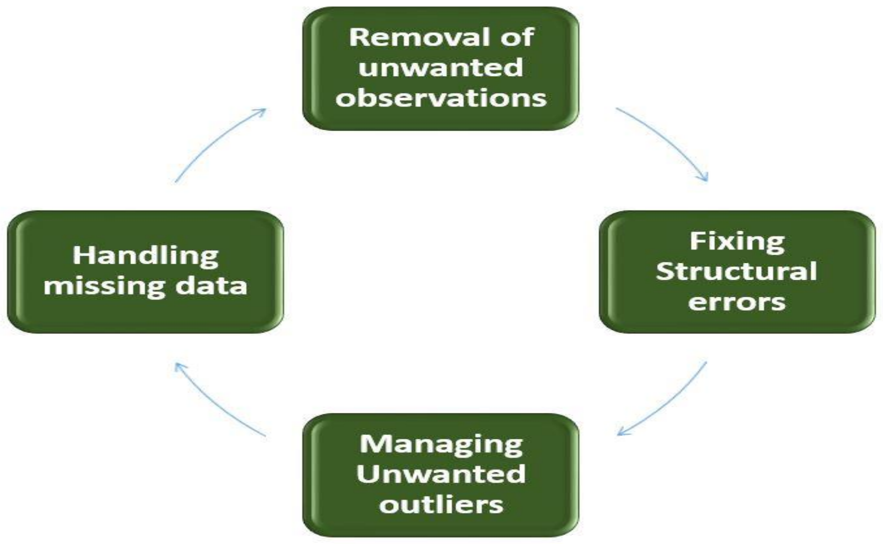
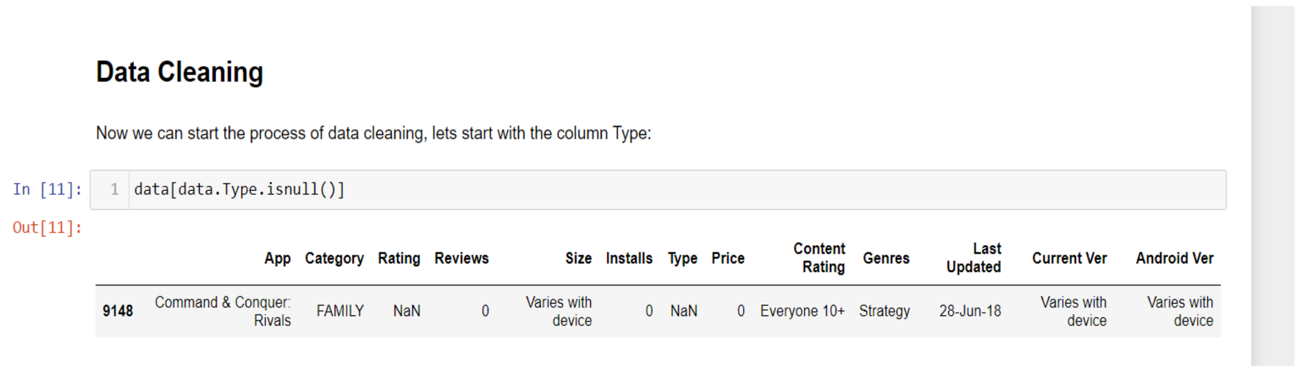


Figure 6.2 Steps involved in Data Cleaning



Data Cleaning

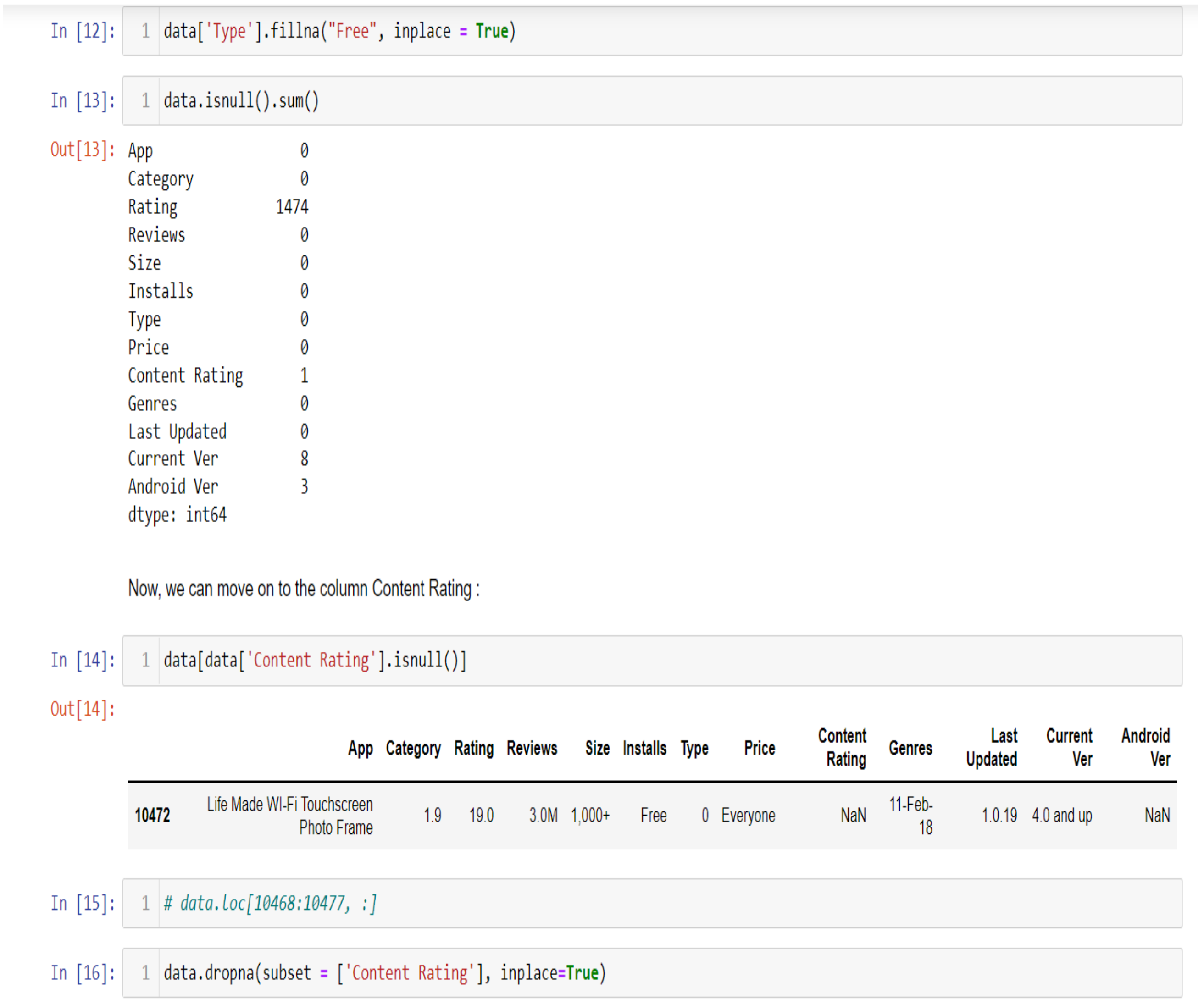
Now we can start the process of data cleaning, lets start with the column Type:

```
In [11]: data[data.Type.isnull()]
```

Out[11]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
9148	Command & Conquer: Rivals	FAMILY	NaN	0	Varies with device	0	NaN	0	Everyone 10+	Strategy	28-Jun-18	Varies with device	Varies with device

Since there is only one missing value in this column, So, let's fill the missing value. After cross-checking in the play-store the missing value is found to be Free, So now we can fill the missing value with-free.



```
In [12]: data['Type'].fillna("Free", inplace = True)
```

```
In [13]: data.isnull().sum()
```

```
Out[13]: App                  0
         Category             0
         Rating            1474
         Reviews              0
         Size                 0
         Installs             0
         Type                 0
         Price                0
         Content Rating       1
         Genres               0
         Last Updated         0
         Current Ver          8
         Android Ver          3
         dtype: int64
```

Now, we can move on to the column Content Rating :

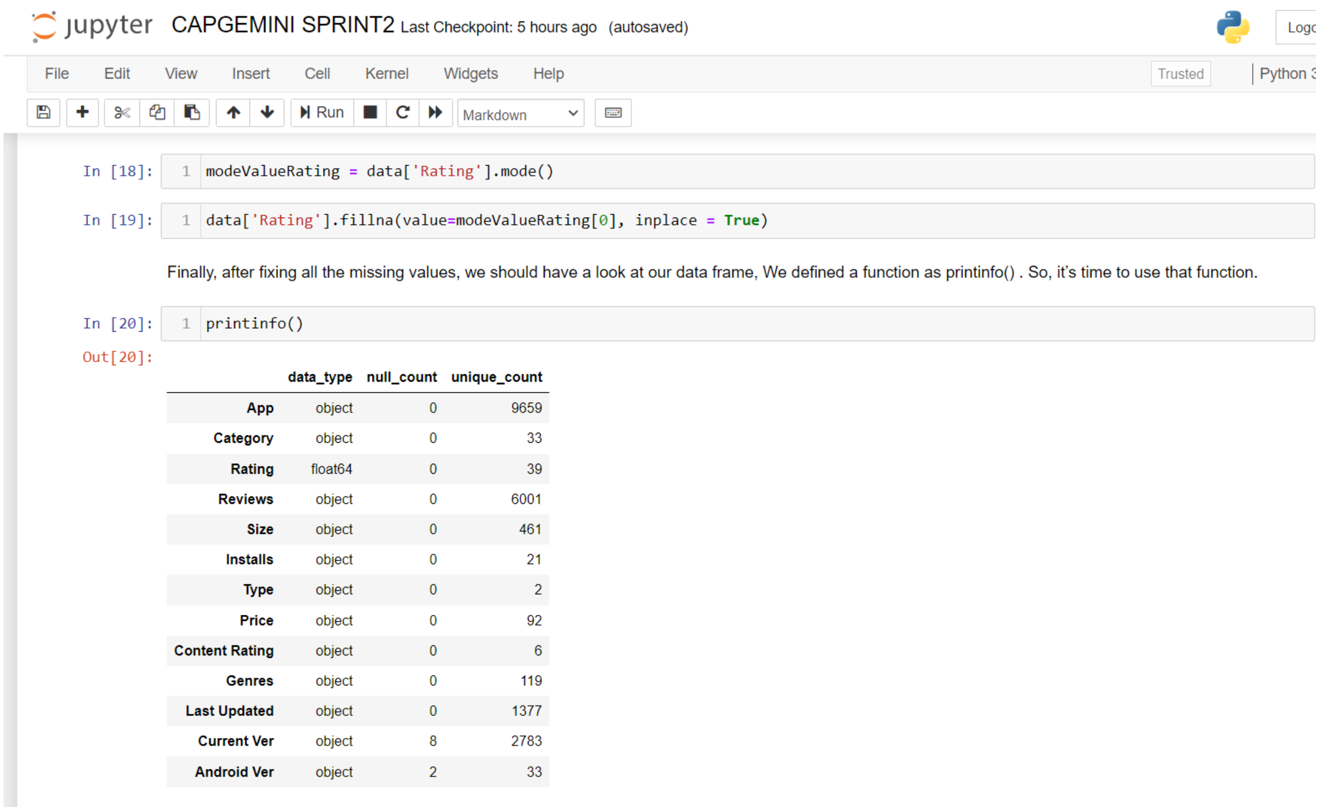
```
In [14]: data[data['Content Rating'].isnull()]
```

Out[14]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
10472	Life Made WI-Fi Touchscreen Photo Frame	1.9	19.0	3.0M	1,000+	Free	0	Everyone	NaN	11-Feb-18	1.0.19	4.0 and up	NaN

```
In [15]: # data.loc[10468:10477, :]
```

```
In [16]: data.dropna(subset = ['Content Rating'], inplace=True)
```


jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (autosaved)
Logo
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

```
In [18]: modeValueRating = data['Rating'].mode()
```

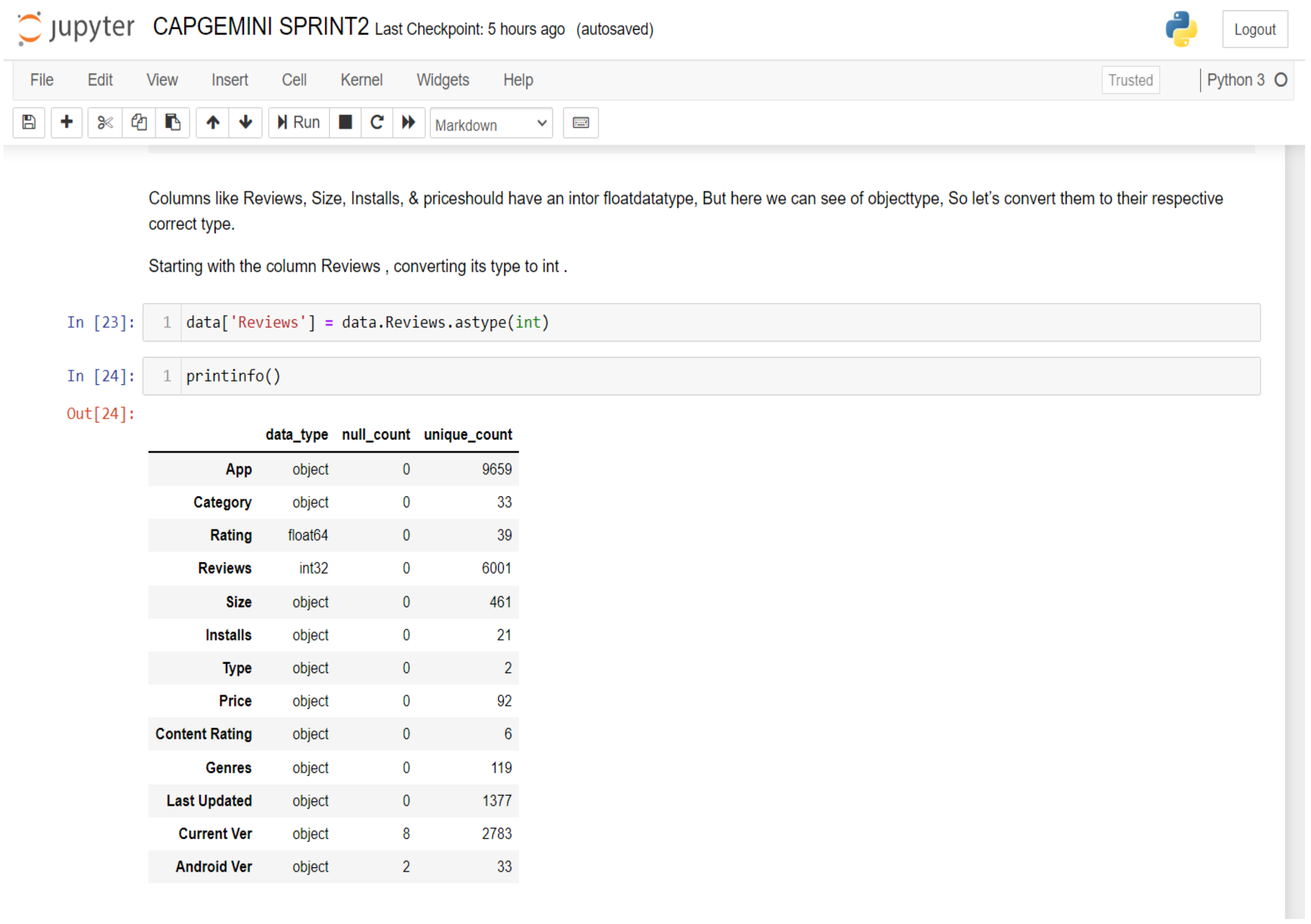
```
In [19]: data['Rating'].fillna(value=modeValueRating[0], inplace = True)
```

Finally, after fixing all the missing values, we should have a look at our data frame, We defined a function as printinfo() . So, it's time to use that function.

```
In [20]: printinfo()
```

Out[20]:

	data_type	null_count	unique_count
App	object	0	9659
Category	object	0	33
Rating	float64	0	39
Reviews	object	0	6001
Size	object	0	461
Installs	object	0	21
Type	object	0	2
Price	object	0	92
Content Rating	object	0	6
Genres	object	0	119
Last Updated	object	0	1377
Current Ver	object	8	2783
Android Ver	object	2	33


jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

Columns like Reviews, Size, Installs, & priceshould have an intor floatdatatype, But here we can see of objecttype, So let's convert them to their respective correct type.

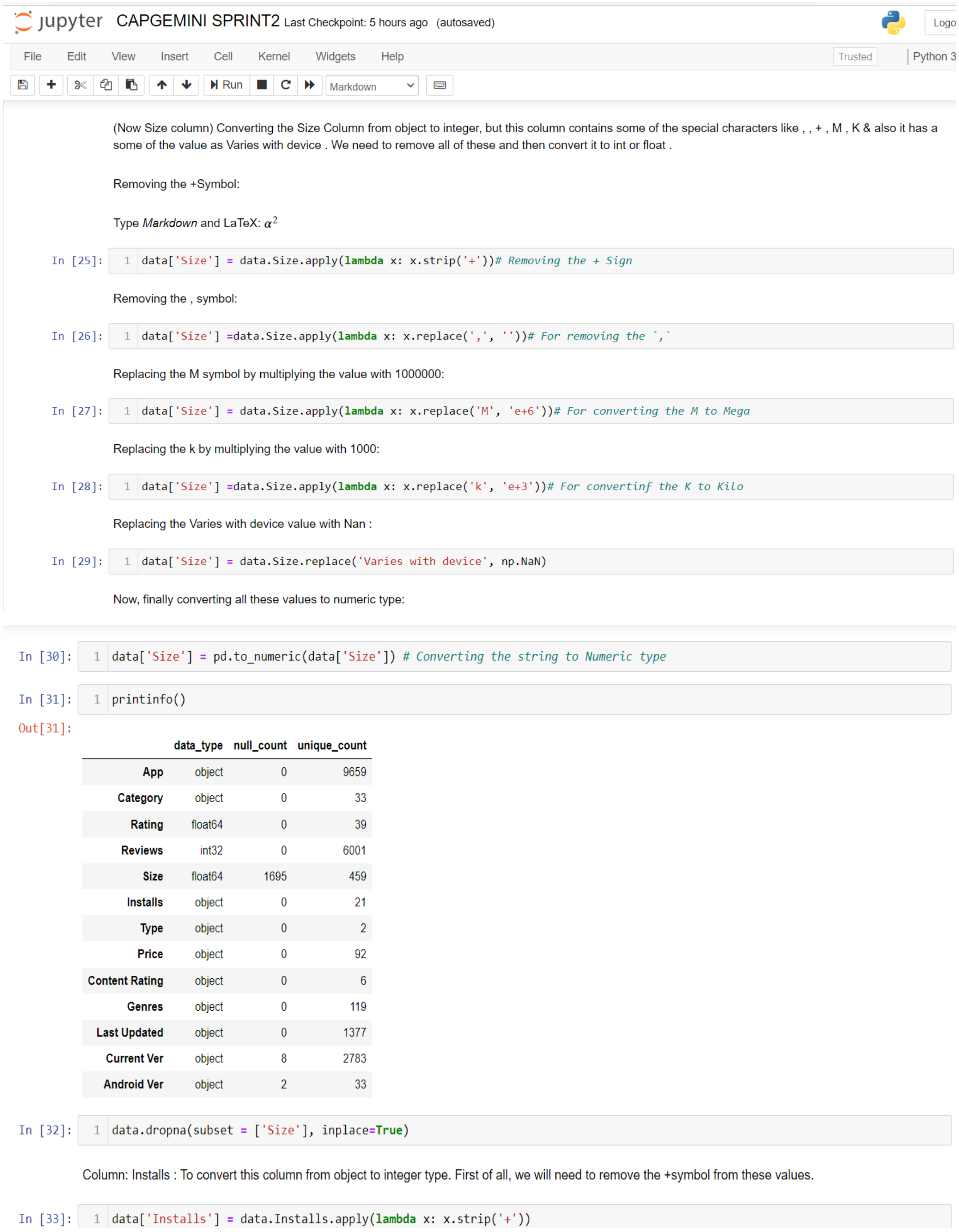
Starting with the column Reviews , converting its type to int .

```
In [23]: data['Reviews'] = data.Reviews.astype(int)
```

```
In [24]: printinfo()
```

Out[24]:

	data_type	null_count	unique_count
App	object	0	9659
Category	object	0	33
Rating	float64	0	39
Reviews	int32	0	6001
Size	object	0	461
Installs	object	0	21
Type	object	0	2
Price	object	0	92
Content Rating	object	0	6
Genres	object	0	119
Last Updated	object	0	1377
Current Ver	object	8	2783
Android Ver	object	2	33



CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (autosaved)

(Now Size column) Converting the Size Column from object to integer, but this column contains some of the special characters like , , + , M , K & also it has a some of the value as Varies with device . We need to remove all of these and then convert it to int or float .

Removing the +Symbol:

Type *Markdown* and LaTeX: α^2

In [25]:
```
data['Size'] = data.Size.apply(lambda x: x.strip('+'))# Removing the + Sign
```

Removing the , symbol:

In [26]:
```
data['Size'] =data.Size.apply(lambda x: x.replace(',', ''))# For removing the `,`
```

Replacing the M symbol by multiplying the value with 1000000:

In [27]:
```
data['Size'] = data.Size.apply(lambda x: x.replace('M', 'e+6'))# For converting the M to Mega
```

Replacing the k by multiplying the value with 1000:

In [28]:
```
data['Size'] =data.Size.apply(lambda x: x.replace('k', 'e+3'))# For convertinf the K to Kilo
```

Replacing the Varies with device value with Nan :

In [29]:
```
data['Size'] = data.Size.replace('Varies with device', np.NaN)
```

Now, finally converting all these values to numeric type:

In [30]:
```
data['Size'] = pd.to_numeric(data['Size']) # Converting the string to Numeric type
```

In [31]:
```
printinfo()
```

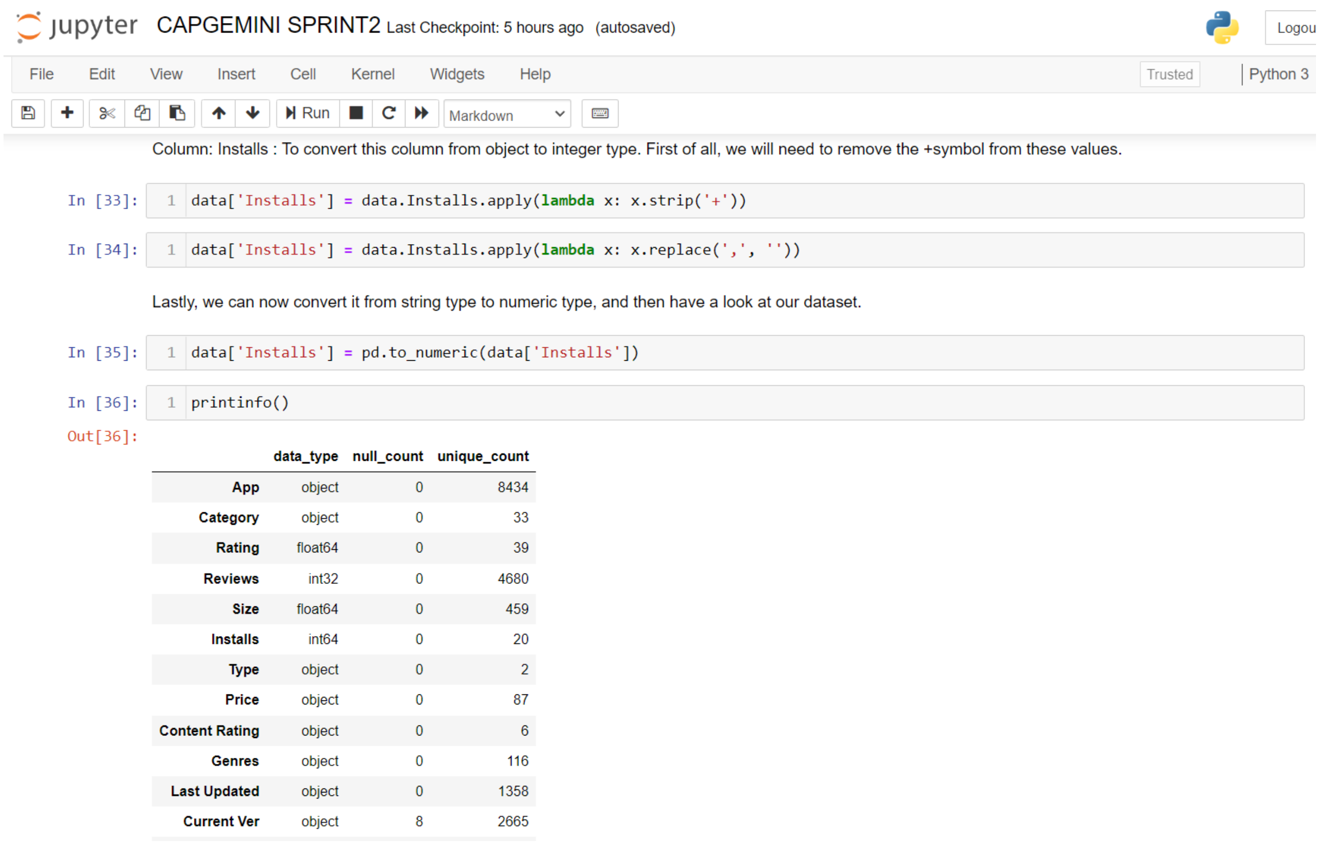
Out[31]:

	data_type	null_count	unique_count
App	object	0	9659
Category	object	0	33
Rating	float64	0	39
Reviews	int32	0	6001
Size	float64	1695	459
Installs	object	0	21
Type	object	0	2
Price	object	0	92
Content Rating	object	0	6
Genres	object	0	119
Last Updated	object	0	1377
Current Ver	object	8	2783
Android Ver	object	2	33

In [32]:
```
data.dropna(subset = ['Size'], inplace=True)
```

Column: Installs : To convert this column from object to integer type. First of all, we will need to remove the +symbol from these values.

In [33]:
```
data['Installs'] = data.Installs.apply(lambda x: x.strip('+'))
```


jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (autosaved)
Logou
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

Column: Installs : To convert this column from object to integer type. First of all, we will need to remove the +symbol from these values.

In [33]:
```
data['Installs'] = data.Installs.apply(lambda x: x.strip('+'))
```

In [34]:
```
data['Installs'] = data.Installs.apply(lambda x: x.replace(',', ''))
```

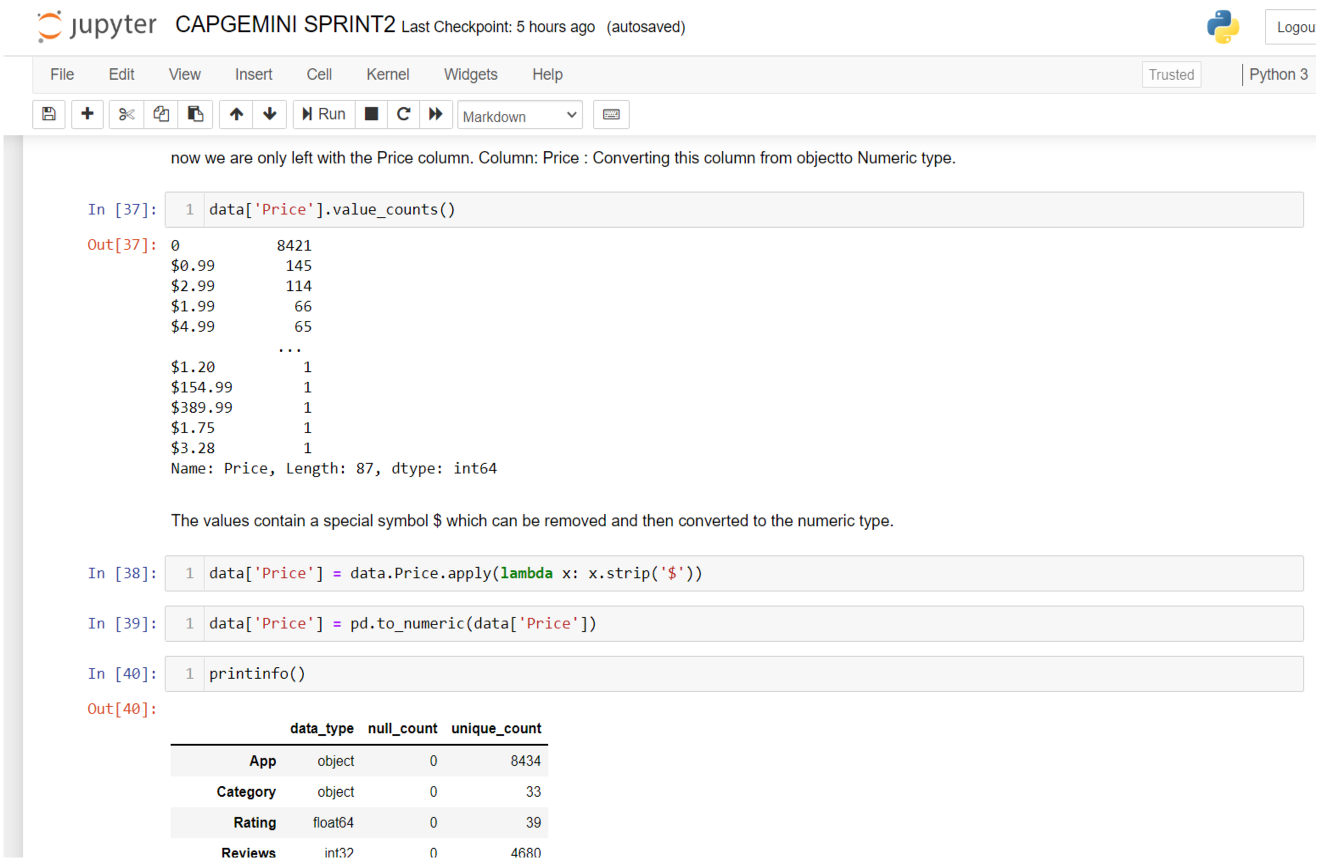
Lastly, we can now convert it from string type to numeric type, and then have a look at our dataset.

In [35]:
```
data['Installs'] = pd.to_numeric(data['Installs'])
```

In [36]:
```
printinfo()
```

Out[36]:

	data_type	null_count	unique_count
App	object	0	8434
Category	object	0	33
Rating	float64	0	39
Reviews	int32	0	4680
Size	float64	0	459
Installs	int64	0	20
Type	object	0	2
Price	object	0	87
Content Rating	object	0	6
Genres	object	0	116
Last Updated	object	0	1358
Current Ver	object	8	2665


jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (autosaved)
Logou
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

now we are only left with the Price column. Column: Price : Converting this column from objectto Numeric type.

In [37]:
```
data['Price'].value_counts()
```

Out[37]:
```
0          8421
$0.99       145
$2.99       114
$1.99        66
$4.99        65
           ...
$1.20         1
$154.99       1
$389.99       1
$1.75         1
$3.28         1
Name: Price, Length: 87, dtype: int64
```

The values contain a special symbol $ which can be removed and then converted to the numeric type.

In [38]:
```
data['Price'] = data.Price.apply(lambda x: x.strip('$'))
```

In [39]:
```
data['Price'] = pd.to_numeric(data['Price'])
```

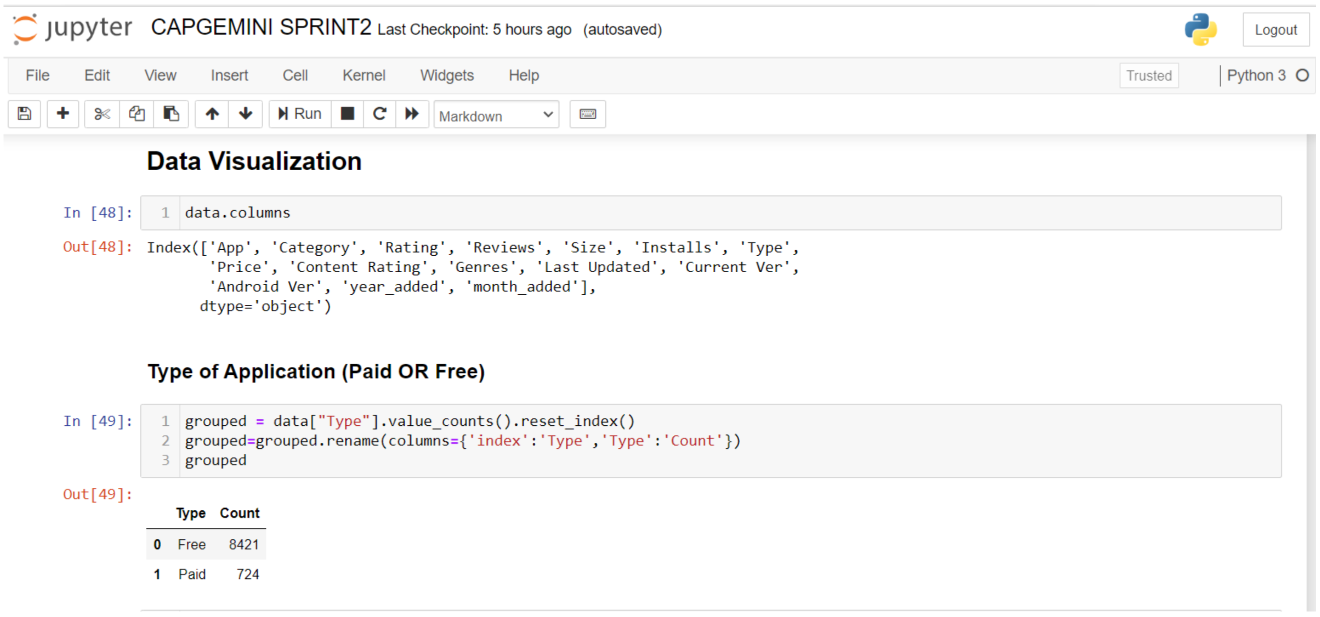
In [40]:
```
printinfo()
```

Out[40]:

	data_type	null_count	unique_count
App	object	0	8434
Category	object	0	33
Rating	float64	0	39
Reviews	int32	0	4680

6.2 EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is an approach to analyze the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.



Data Visualization

```
In [48]: data.columns
Out[48]: Index(['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type',
                'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver',
                'Android Ver', 'year_added', 'month_added'],
               dtype='object')
```

Type of Application (Paid OR Free)

```
In [49]: grouped = data["Type"].value_counts().reset_index()
         grouped=grouped.rename(columns={'index':'Type','Type':'Count'})
         grouped
Out[49]:
```

	Type	Count
0	Free	8421
1	Paid	724

6.2.1 Free VS Paid

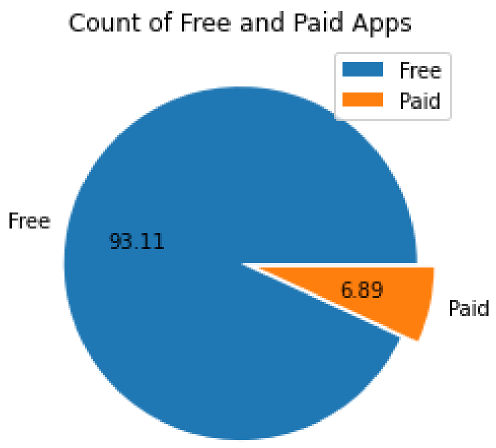
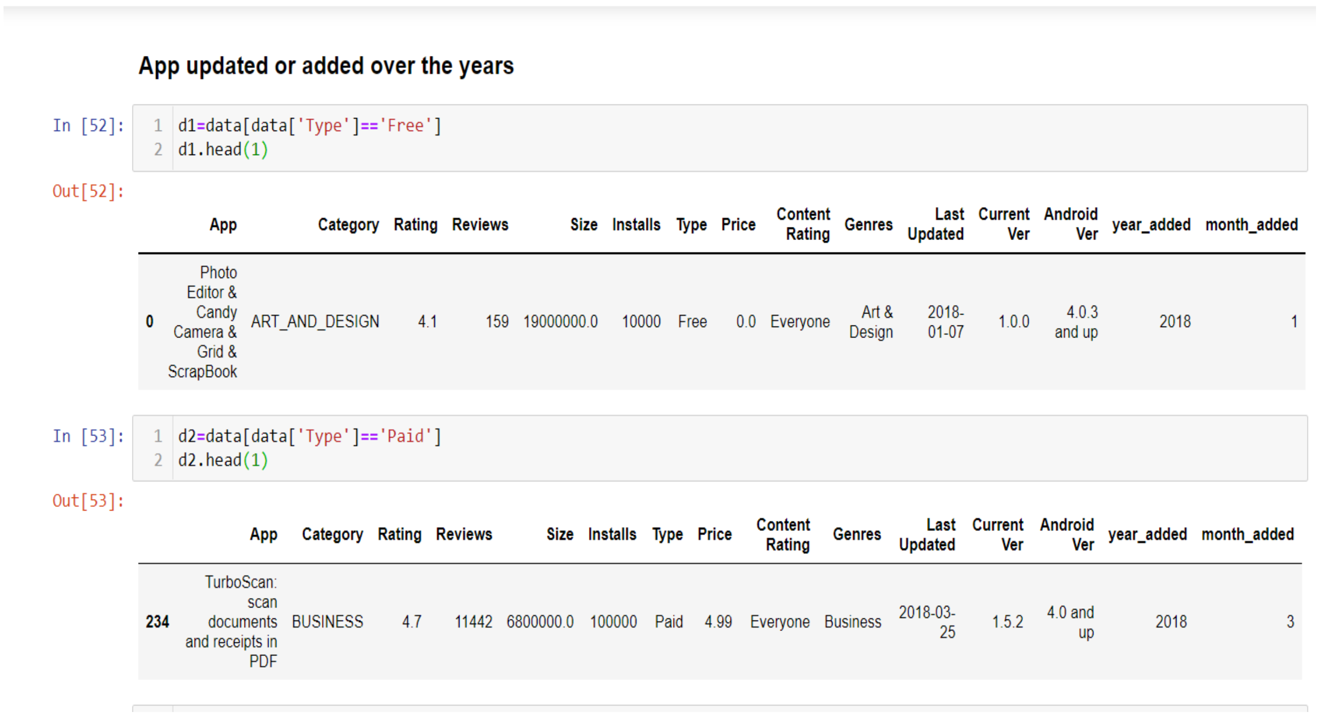


Figure 6.3

Here we can see that 93.11% apps are free and 6.89% apps are paid on Google Play Store, so we can say that Most of the apps are free on Google Play Store.

6.2.2 Updated Apps



App updated or added over the years

In [52]:
```
d1=data[data['Type']=='Free']
d1.head(1)
```

Out[52]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver	year_added	month_added
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000000.0	10000	Free	0.0	Everyone	Art & Design	2018-01-07	1.0.0	4.0.3 and up	2018	1

In [53]:
```
d2=data[data['Type']=='Paid']
d2.head(1)
```

Out[53]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver	year_added	month_added
234	TurboScan: scan documents and receipts in PDF	BUSINESS	4.7	11442	6800000.0	100000	Paid	4.99	Everyone	Business	2018-03-25	1.5.2	4.0 and up	2018	3

In the below plot we plot the app updated or added over the year Free vs Paid. By observing this plot we conclude that before 2011 there were no paid apps. But with the year free apps are added in huge amount in comparison to paid apps. We can conclude that people like free service more than paid service.

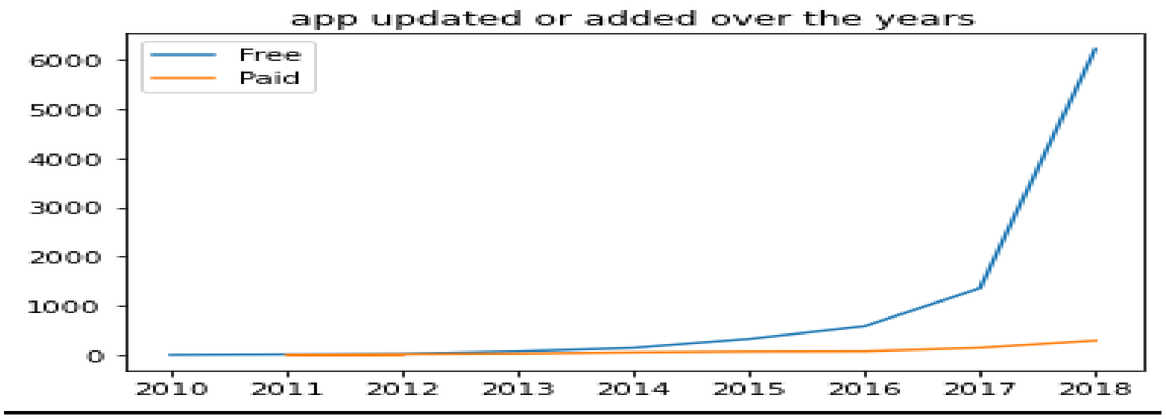
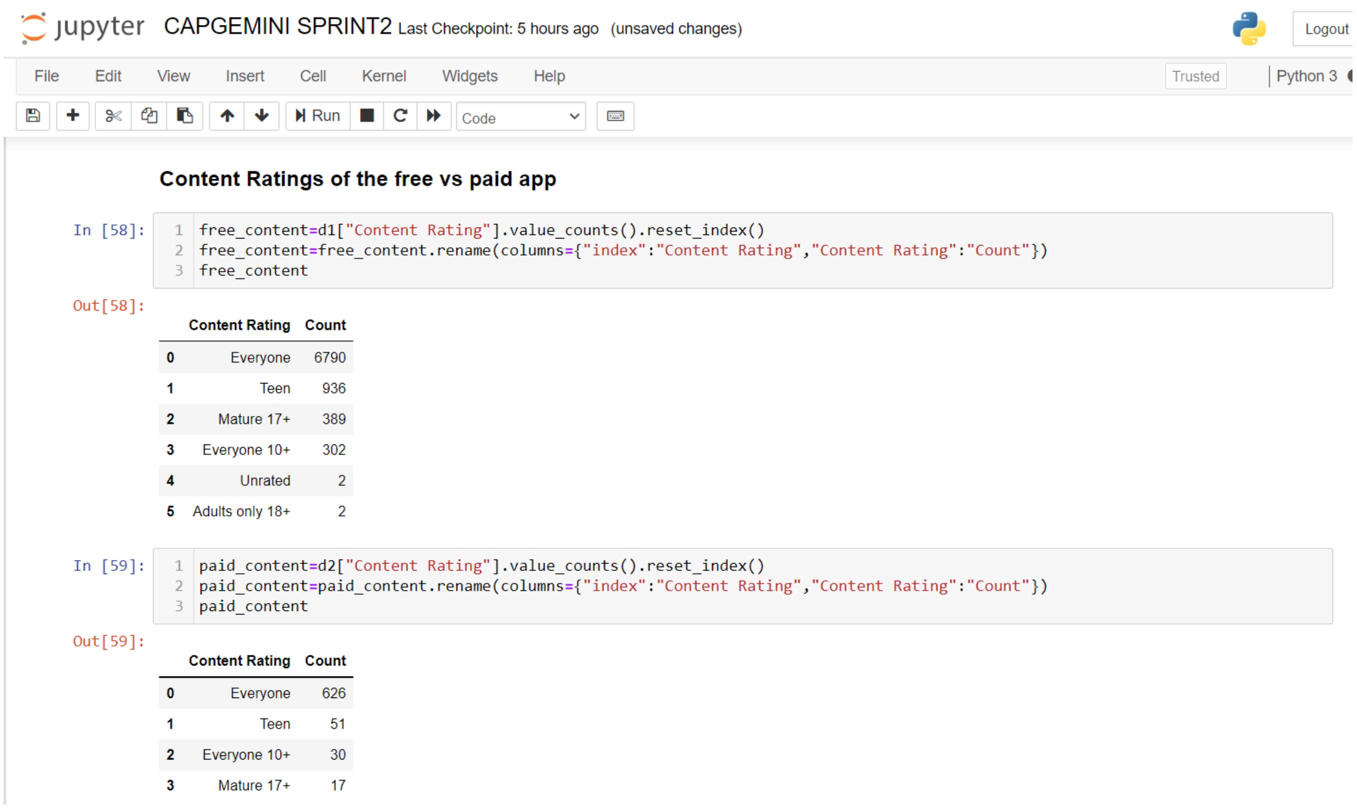


Figure 6.4

6.2.3 Content Ratings of the free vs paid app


jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

Content Ratings of the free vs paid app

In [58]:
```
free_content=d1["Content Rating"].value_counts().reset_index()
free_content=free_content.rename(columns={"index":"Content Rating","Content Rating":"Count"})
free_content
```

Out[58]:

	Content Rating	Count
0	Everyone	6790
1	Teen	936
2	Mature 17+	389
3	Everyone 10+	302
4	Unrated	2
5	Adults only 18+	2

In [59]:
```
paid_content=d2["Content Rating"].value_counts().reset_index()
paid_content=paid_content.rename(columns={"index":"Content Rating","Content Rating":"Count"})
paid_content
```

Out[59]:

	Content Rating	Count
0	Everyone	626
1	Teen	51
2	Everyone 10+	30
3	Mature 17+	17

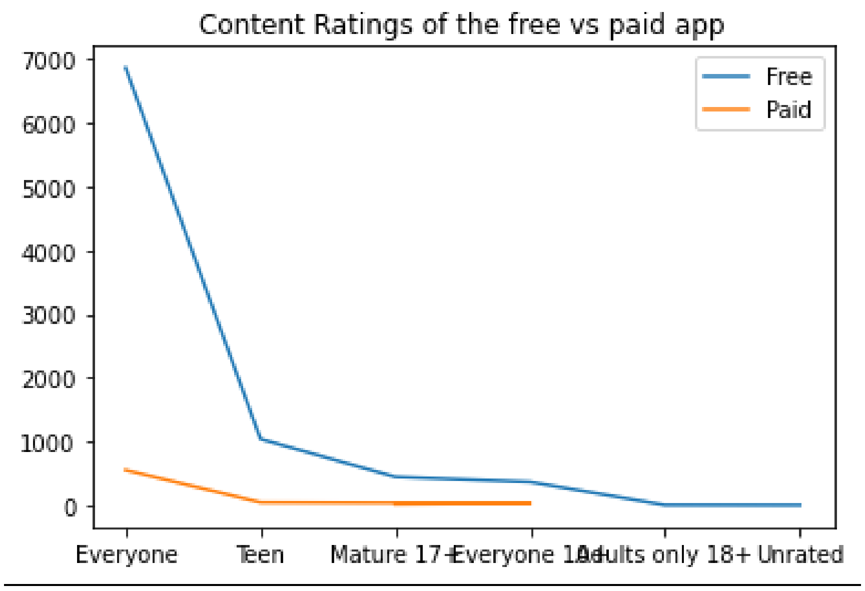
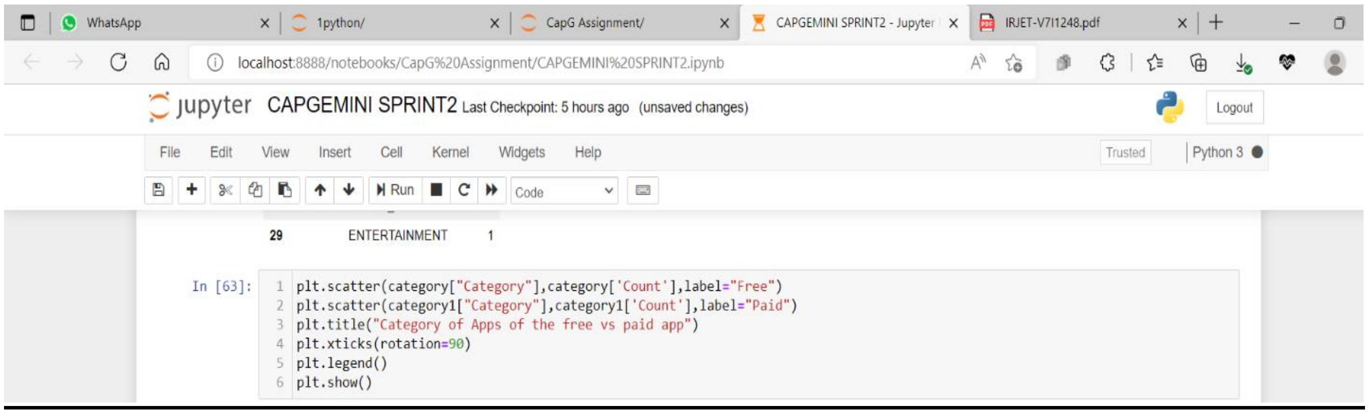


Figure 6.5

The above plot shows that most of the app content rating are for everyone and most of them are Free.

6.2.4 Category of Apps


WhatsApp
1python/
CapG Assignment/
CAPGEMINI SPRINT2 - Jupyter
IRJET-V7I1248.pdf
localhost:8888/notebooks/CapG%20Assignment/CAPGEMINI%20SPRINT2.ipynb
jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

```
29        ENTERTAINMENT     1
```

```
In [63]:
plt.scatter(category["Category"],category['Count'],label="Free")
plt.scatter(category1["Category"],category1['Count'],label="Paid")
plt.title("Category of Apps of the free vs paid app")
plt.xticks(rotation=90)
plt.legend()
plt.show()
```

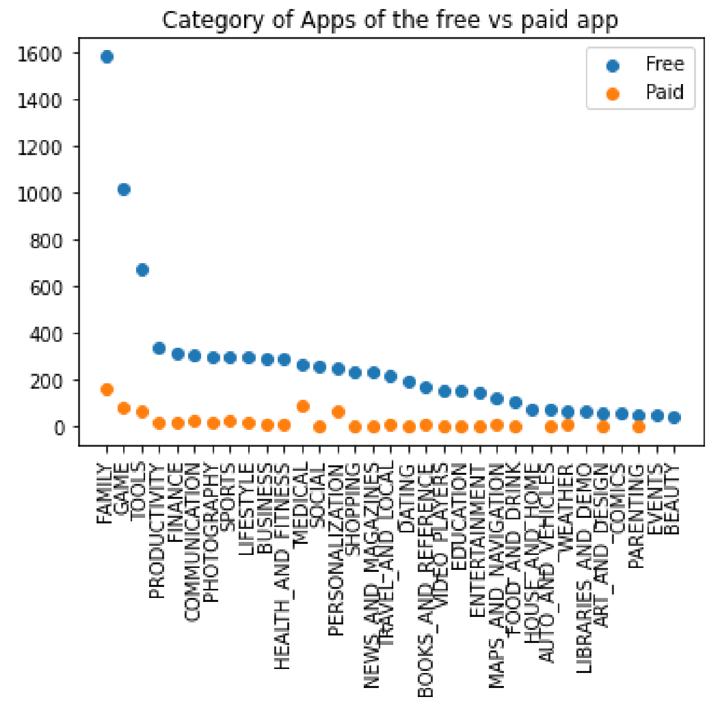
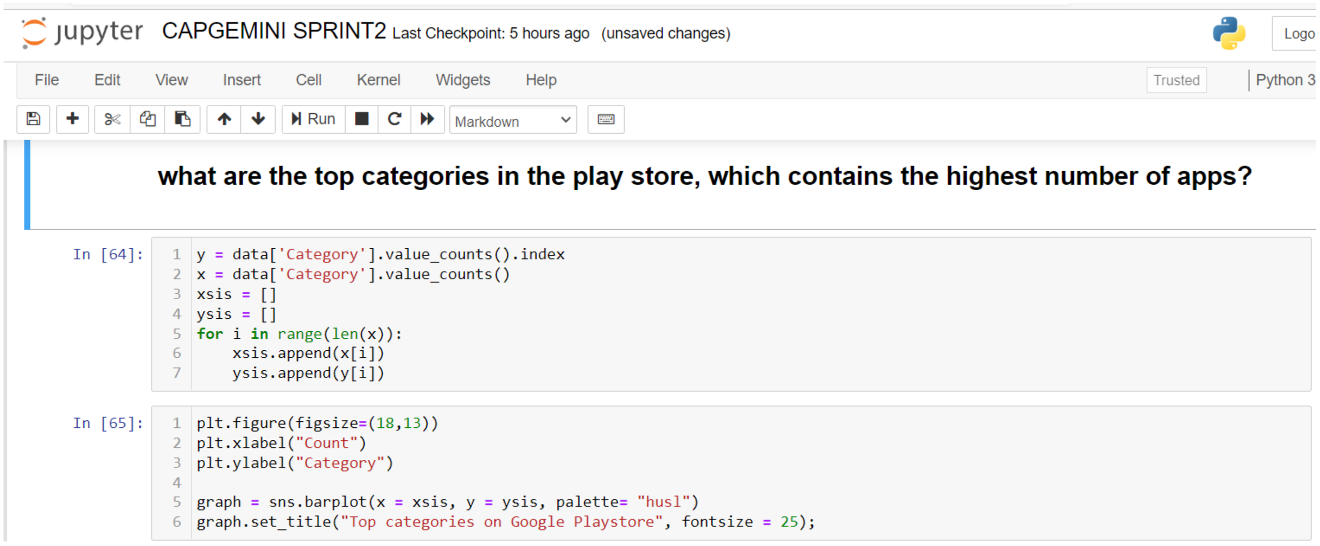


Figure 6.6

From the above chart we can find that most of the apps which are on Google Play Store belong to Family, Gamming and Tools.

6.2.5 Top Categories in google play store



what are the top categories in the play store, which contains the highest number of apps?

In [64]:
```python
y = data['Category'].value_counts().index
x = data['Category'].value_counts()
xsis = []
ysis = []
for i in range(len(x)):
    xsis.append(x[i])
    ysis.append(y[i])
```

In [65]:
```python
plt.figure(figsize=(18,13))
plt.xlabel("Count")
plt.ylabel("Category")

graph = sns.barplot(x = xsis, y = ysis, palette= "husl")
graph.set_title("Top categories on Google Playstore", fontsize = 25);
```

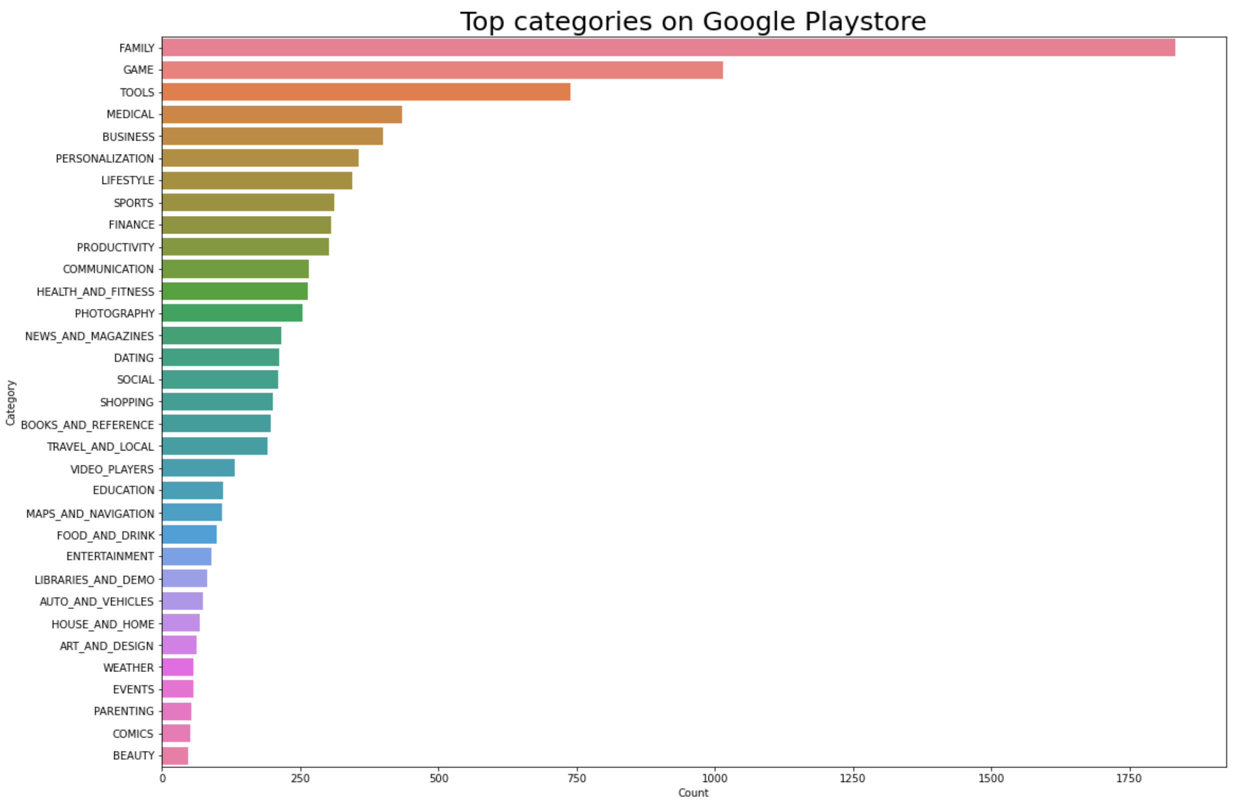


Figure 6.7 Top categories on Play-store

So there are all total of 33 categories in the dataset from the above output we can come to the conclusion that in the play store most of the apps are under Family & Game category and least are of Beauty & Comics Category.

6.2.6 Content Rating

Which category of Apps from the 'Content Rating' column is found more on the play store?

```
In [66]: x2 = data['Content Rating'].value_counts().index
         y2 = data['Content Rating'].value_counts()

         x2sis = []
         y2sis = []
         for i in range(len(x2)):
             x2sis.append(x2[i])
             y2sis.append(y2[i])
```

```
In [67]: plt.figure(figsize=(12,10))
         plt.bar(x2sis,y2sis,width=0.8,color=['#15244C','#FFFF48','#292734','#EF2920','#CD202D','#ECC5F2'], alpha=0.8);
         plt.title('Content Rating',size = 20);
         plt.ylabel('Apps(Count)');
         plt.xlabel('Content Rating');
```

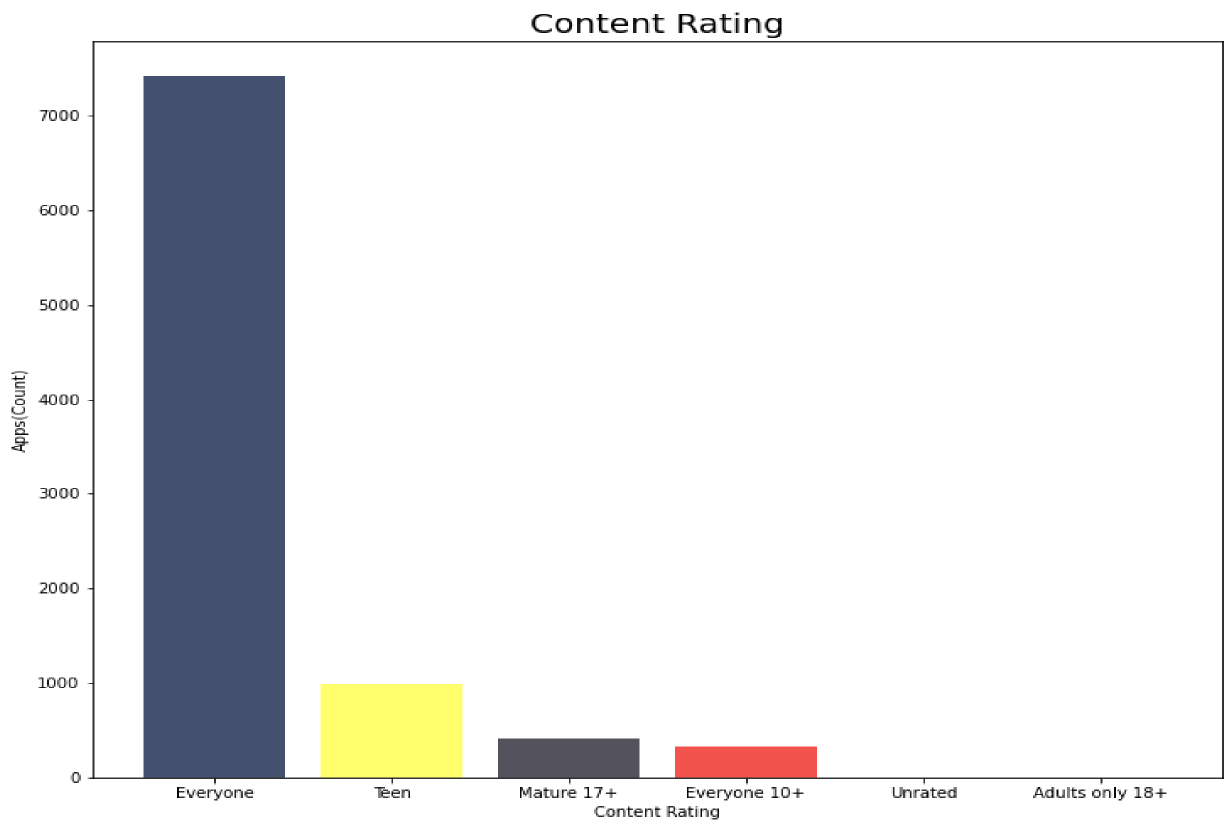


Figure 6.8 Content Rating

From the above plot, we can see that the Everyone category has the highest number of apps.

6.2.7 Distribution of the ratings of the data frame

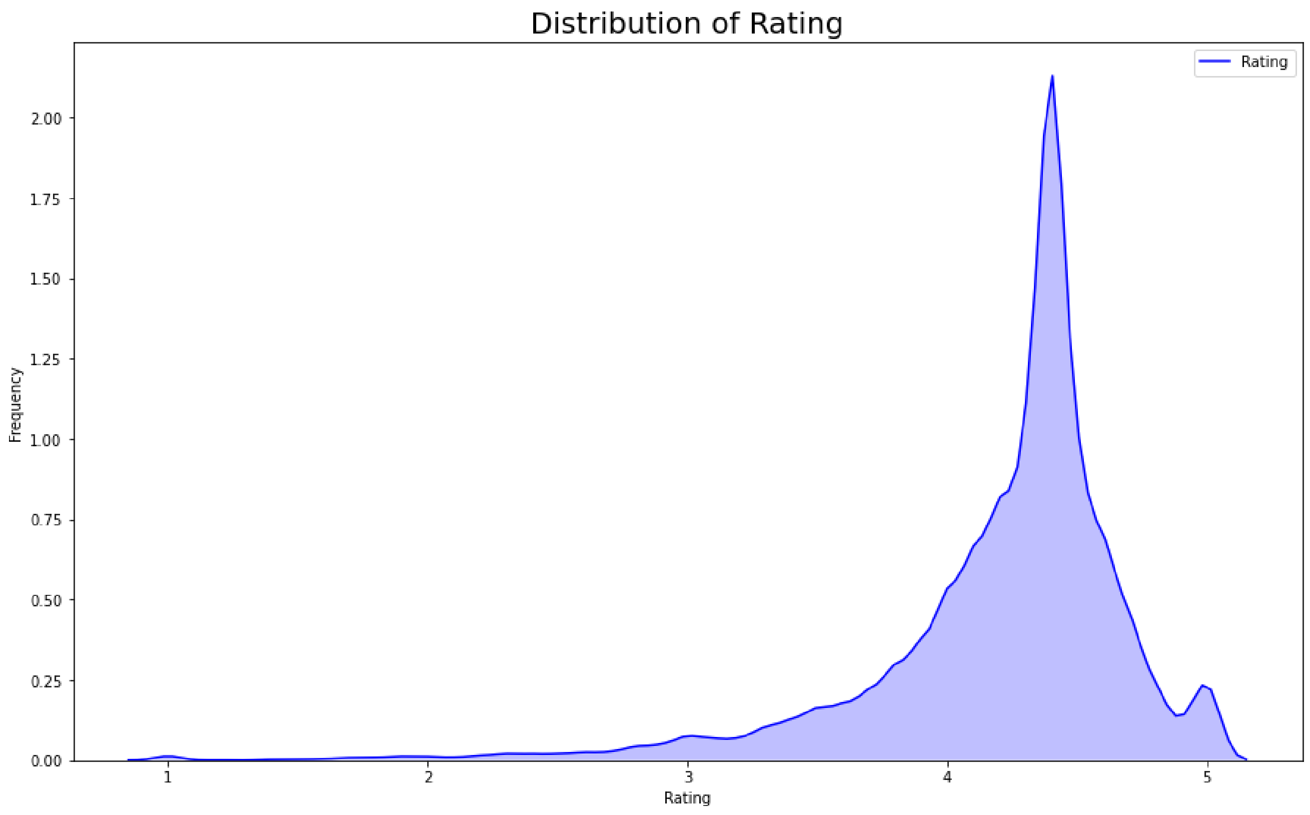
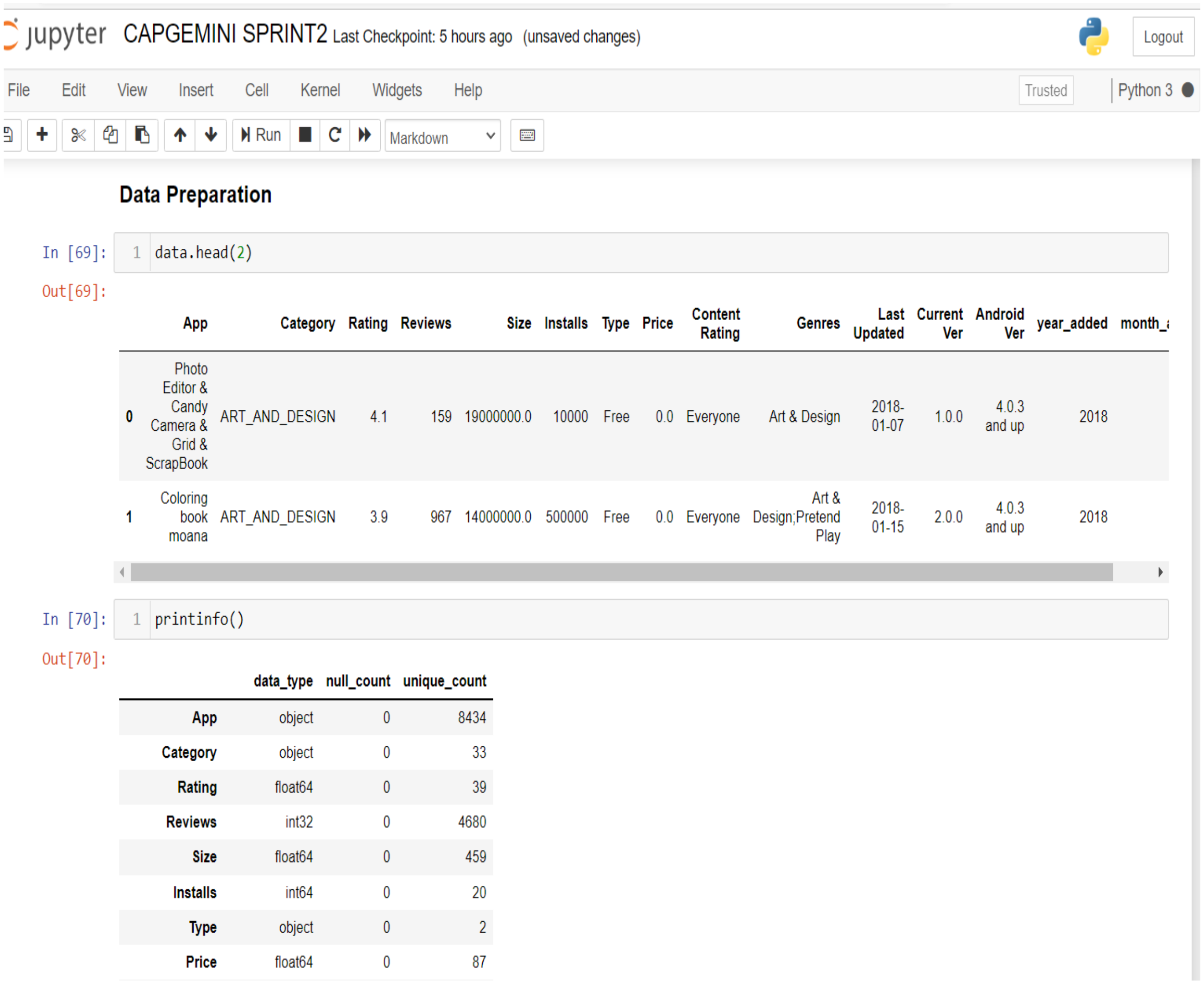


Figure 6.9 Rating Distribution

From the above graph, we can come to the conclusion that most of the apps in the google play store are rated between 3.5 to 4.8.

6.3 DATA PREPARATION

Data Preprocessing is a crucial and very first step before building and deploying your Machine Learning Model. And while building a model it's not the case that every time you will get clean and formatted data to work on. It is mandatory to clean and check the data before use. So, we use data preprocessing for these.


jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

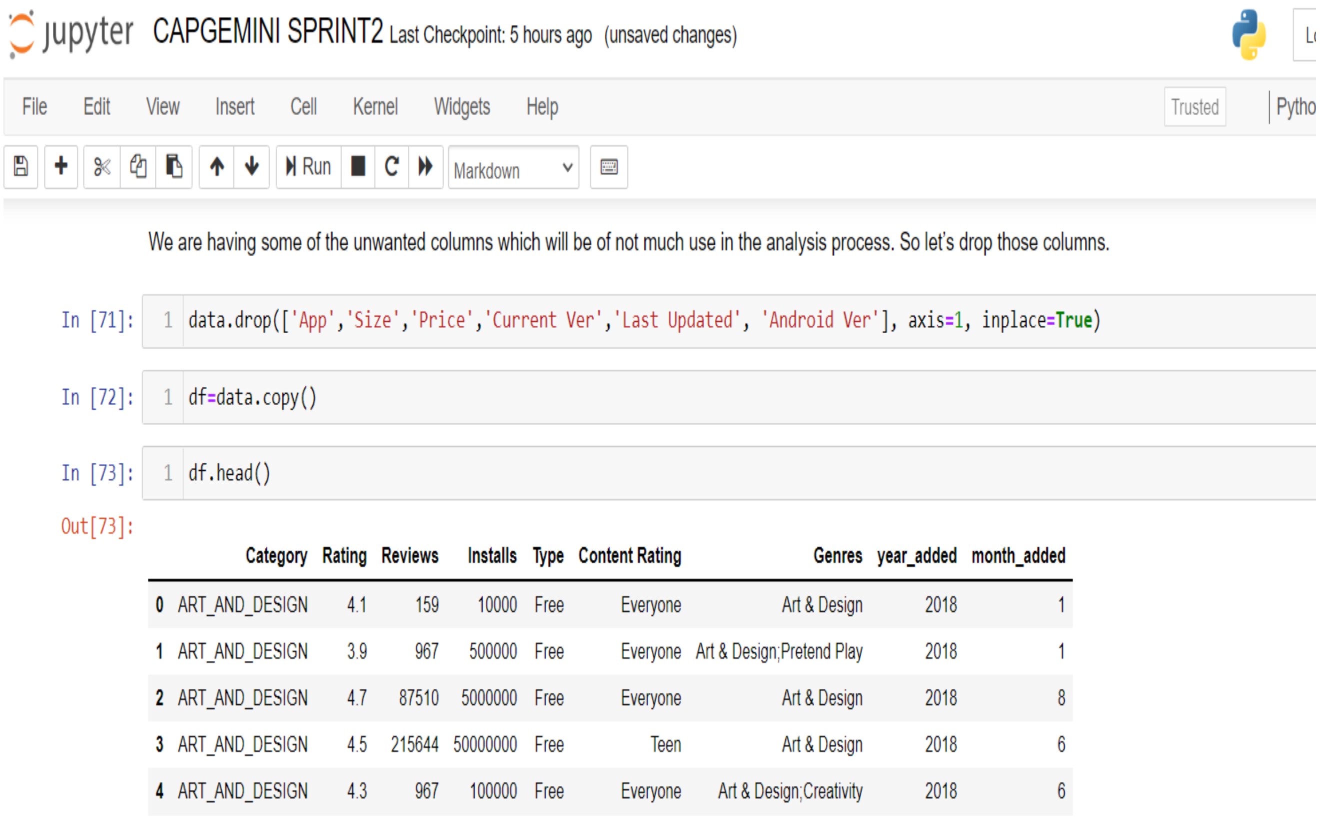
Data Preparation

In [69]:
```
data.head(2)
```
Out[69]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver	year_added	month_a
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000000.0	10000	Free	0.0	Everyone	Art & Design	2018-01-07	1.0.0	4.0.3 and up	2018	
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000000.0	500000	Free	0.0	Everyone	Art & Design;Pretend Play	2018-01-15	2.0.0	4.0.3 and up	2018	

In [70]:
```
printinfo()
```
Out[70]:

	data_type	null_count	unique_count
App	object	0	8434
Category	object	0	33
Rating	float64	0	39
Reviews	int32	0	4680
Size	float64	0	459
Installs	int64	0	20
Type	object	0	2
Price	float64	0	87


jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Pytho
Run
Markdown

We are having some of the unwanted columns which will be of not much use in the analysis process. So let's drop those columns.

```
In [71]: data.drop(['App','Size','Price','Current Ver','Last Updated', 'Android Ver'], axis=1, inplace=True)
```

```
In [72]: df=data.copy()
```

```
In [73]: df.head()
```

```
Out[73]:
```

	Category	Rating	Reviews	Installs	Type	Content Rating	Genres	year_added	month_added
0	ART_AND_DESIGN	4.1	159	10000	Free	Everyone	Art & Design	2018	1
1	ART_AND_DESIGN	3.9	967	500000	Free	Everyone	Art & Design;Pretend Play	2018	1
2	ART_AND_DESIGN	4.7	87510	5000000	Free	Everyone	Art & Design	2018	8
3	ART_AND_DESIGN	4.5	215644	50000000	Free	Teen	Art & Design	2018	6
4	ART_AND_DESIGN	4.3	967	100000	Free	Everyone	Art & Design;Creativity	2018	6

6.4 FEATURE ENCODING

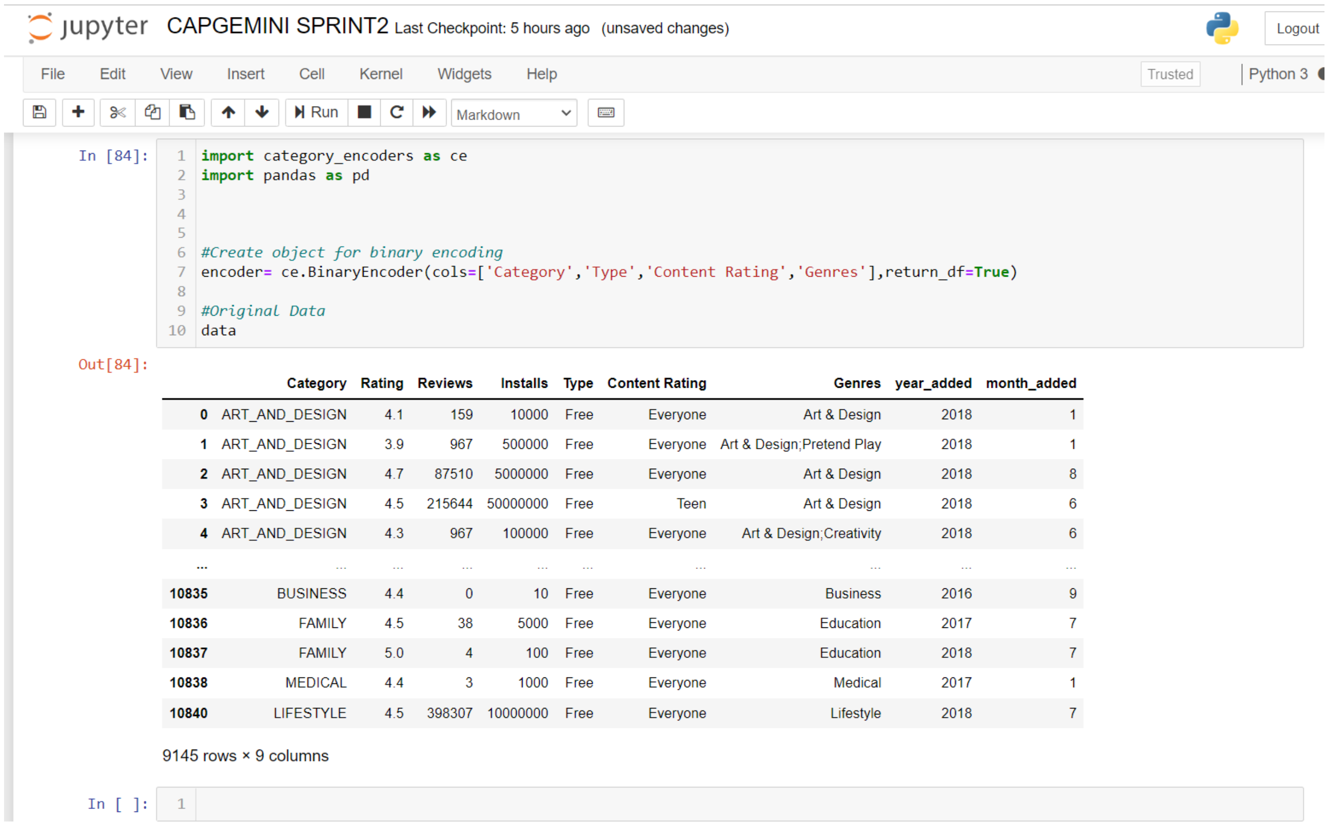
Machine learning models can only work with numerical values. For this reason, it is necessary to transform the categorical values of the relevant features into numerical ones. This process is called feature encoding.

6.4.1 Binary Encoding

Initially categories are encoded as Integer and then converted into binary code, then the digits from that binary string are placed into separate columns.

for e g: for 7: 1 1 1

This method is quite preferable when there are more number of categories. Imagine if you have 100 different categories. One hot encoding will create 100 different columns, but binary encoding only need 7 columns.


jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

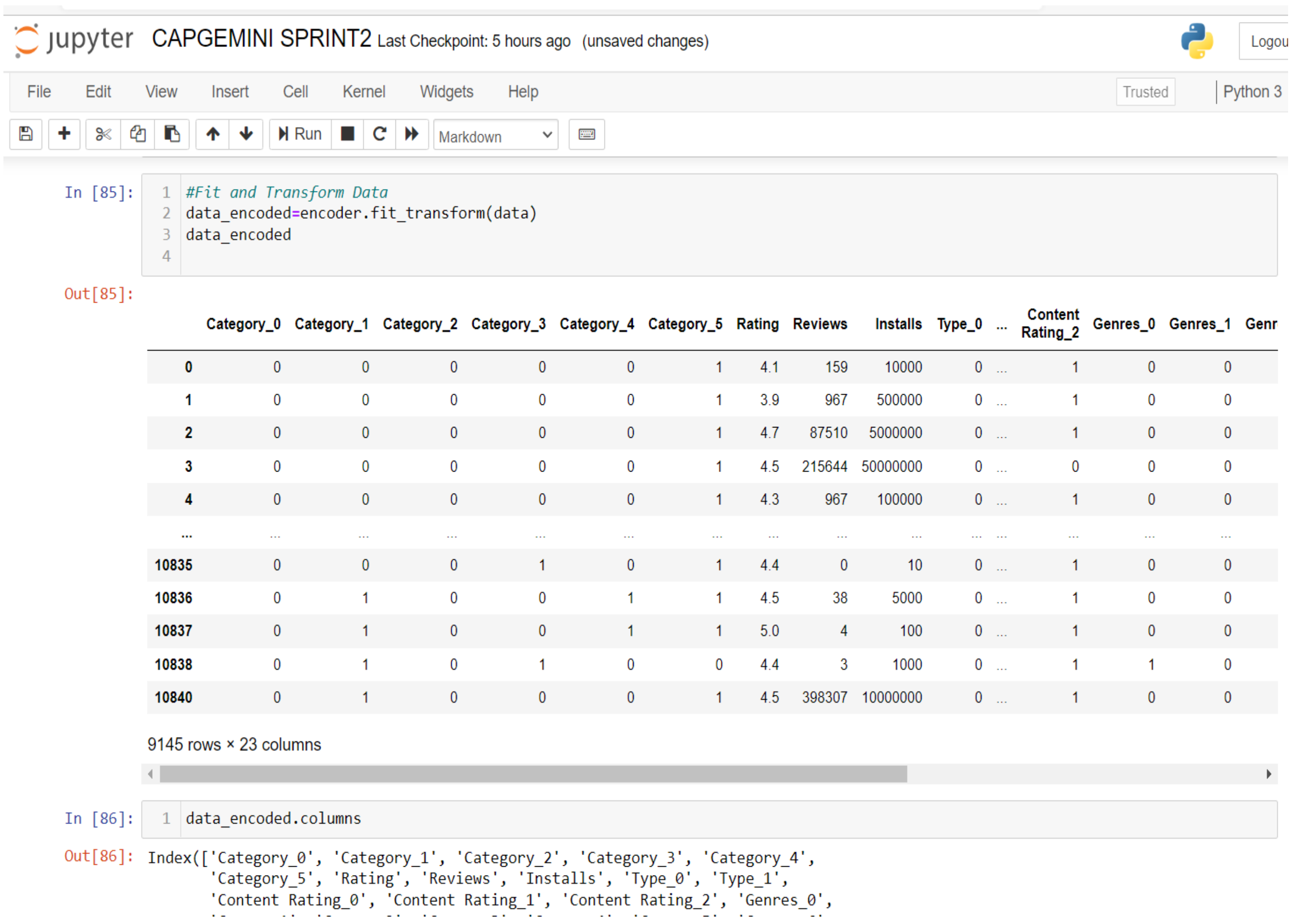
In [84]:
```
import category_encoders as ce
import pandas as pd



#Create object for binary encoding
encoder= ce.BinaryEncoder(cols=['Category','Type','Content Rating','Genres'],return_df=True)

#Original Data
data
```

Out[84]:

	Category	Rating	Reviews	Installs	Type	Content Rating	Genres	year_added	month_added
0	ART_AND_DESIGN	4.1	159	10000	Free	Everyone	Art & Design	2018	1
1	ART_AND_DESIGN	3.9	967	500000	Free	Everyone	Art & Design;Pretend Play	2018	1
2	ART_AND_DESIGN	4.7	87510	5000000	Free	Everyone	Art & Design	2018	8
3	ART_AND_DESIGN	4.5	215644	50000000	Free	Teen	Art & Design	2018	6
4	ART_AND_DESIGN	4.3	967	100000	Free	Everyone	Art & Design;Creativity	2018	6
...	...	...	...	...	...	...	...	...	...
10835	BUSINESS	4.4	0	10	Free	Everyone	Business	2016	9
10836	FAMILY	4.5	38	5000	Free	Everyone	Education	2017	7
10837	FAMILY	5.0	4	100	Free	Everyone	Education	2018	7
10838	MEDICAL	4.4	3	1000	Free	Everyone	Medical	2017	1
10840	LIFESTYLE	4.5	398307	10000000	Free	Everyone	Lifestyle	2018	7

9145 rows × 9 columns

In [ ]:



jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (unsaved changes)

In [85]:
```
#Fit and Transform Data
data_encoded=encoder.fit_transform(data)
data_encoded
```

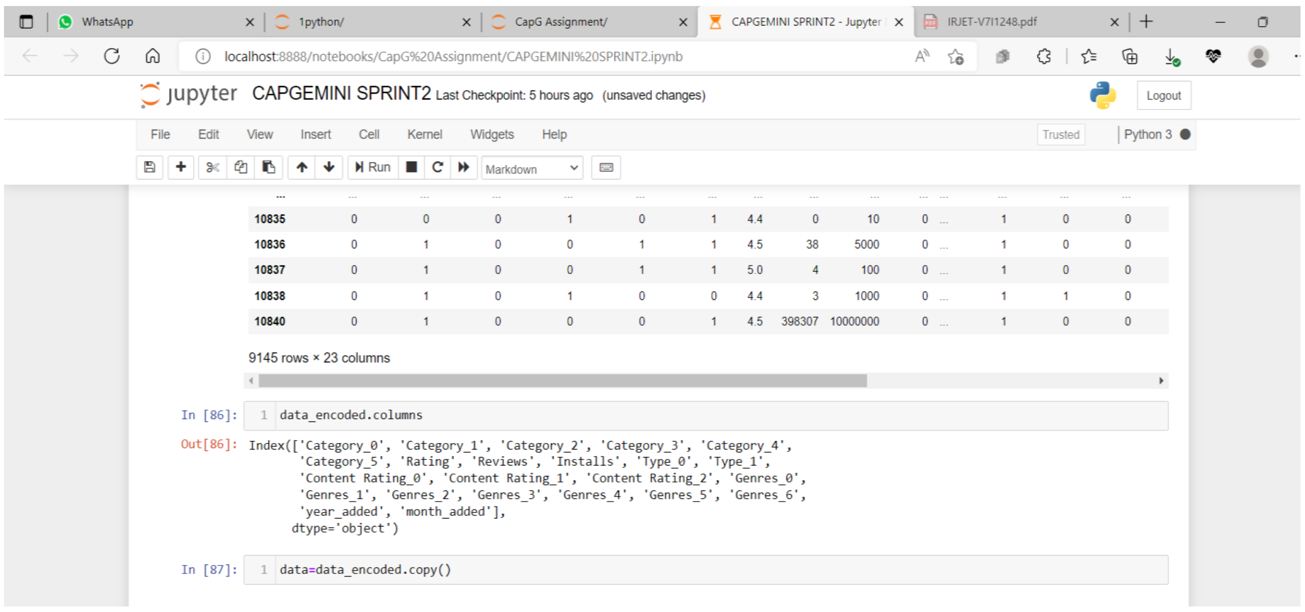
Out[85]:

	Category_0	Category_1	Category_2	Category_3	Category_4	Category_5	Rating	Reviews	Installs	Type_0	...	Content Rating_2	Genres_0	Genres_1	Genr
0	0	0	0	0	0	1	4.1	159	10000	0	...	1	0	0	
1	0	0	0	0	0	1	3.9	967	500000	0	...	1	0	0	
2	0	0	0	0	0	1	4.7	87510	5000000	0	...	1	0	0	
3	0	0	0	0	0	1	4.5	215644	50000000	0	...	0	0	0	
4	0	0	0	0	0	1	4.3	967	100000	0	...	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
10835	0	0	0	1	0	1	4.4	0	10	0	...	1	0	0	
10836	0	1	0	0	1	1	4.5	38	5000	0	...	1	0	0	
10837	0	1	0	0	1	1	5.0	4	100	0	...	1	0	0	
10838	0	1	0	1	0	0	4.4	3	1000	0	...	1	1	0	
10840	0	1	0	0	0	1	4.5	398307	10000000	0	...	1	0	0	

9145 rows × 23 columns

In [86]:
```
data_encoded.columns
```

Out[86]:
```
Index(['Category_0', 'Category_1', 'Category_2', 'Category_3', 'Category_4',
       'Category_5', 'Rating', 'Reviews', 'Installs', 'Type_0', 'Type_1',
       'Content Rating_0', 'Content Rating_1', 'Content Rating_2', 'Genres_0',
```



localhost:8888/notebooks/CapG%20Assignment/CAPGEMINI%20SPRINT2.ipynb

jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (unsaved changes)

...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10835	0	0	0	1	0	1	4.4	0	10	0	...	1	0	0
10836	0	1	0	0	1	1	4.5	38	5000	0	...	1	0	0
10837	0	1	0	0	1	1	5.0	4	100	0	...	1	0	0
10838	0	1	0	1	0	0	4.4	3	1000	0	...	1	1	0
10840	0	1	0	0	0	1	4.5	398307	10000000	0	...	1	0	0

9145 rows × 23 columns

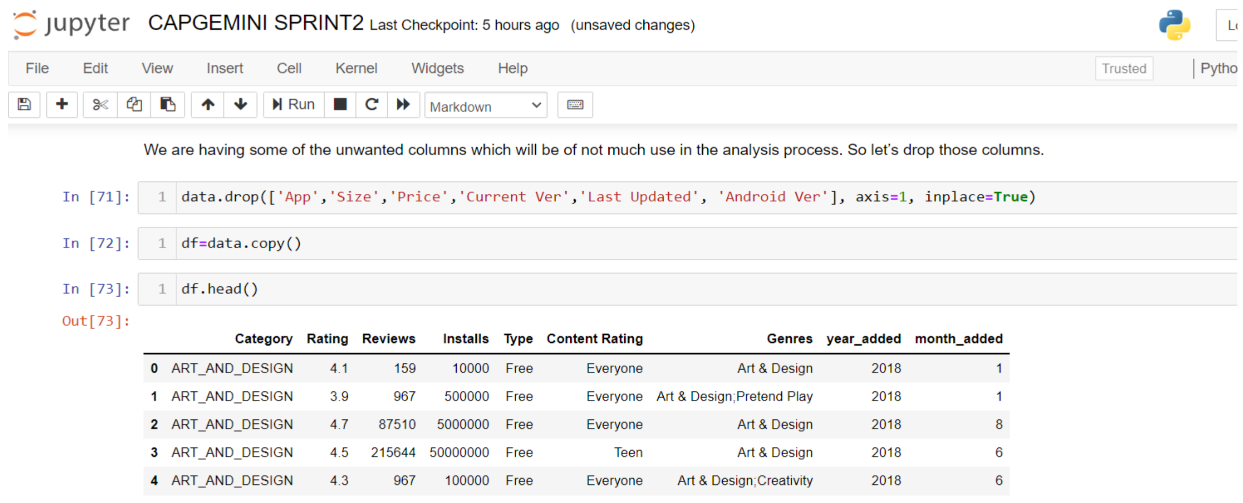
In [86]:
```
data_encoded.columns
```

Out[86]:
```
Index(['Category_0', 'Category_1', 'Category_2', 'Category_3', 'Category_4',
       'Category_5', 'Rating', 'Reviews', 'Installs', 'Type_0', 'Type_1',
       'Content Rating_0', 'Content Rating_1', 'Content Rating_2', 'Genres_0',
       'Genres_1', 'Genres_2', 'Genres_3', 'Genres_4', 'Genres_5', 'Genres_6',
       'year_added', 'month_added'],
      dtype='object')
```

In [87]:
```
data=data_encoded.copy()
```

6.5 FEATURE SELECTION

While building a machine learning model for real-life dataset, we come across a lot of features in the dataset and not all these features are important every time. Adding unnecessary features while training the model leads us to reduce the overall accuracy of the model, increase the complexity of the model and decrease the generalization capability of the model and makes the model biased. Even the saying “Sometimes less is better” goes as well for the machine learning model. Hence, **feature selection** is one of the important steps while building a machine learning model. Its goal is to find the best possible set of features for building a machine learning model.



jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python
Run
Markdown

We are having some of the unwanted columns which will be of not much use in the analysis process. So let's drop those columns.

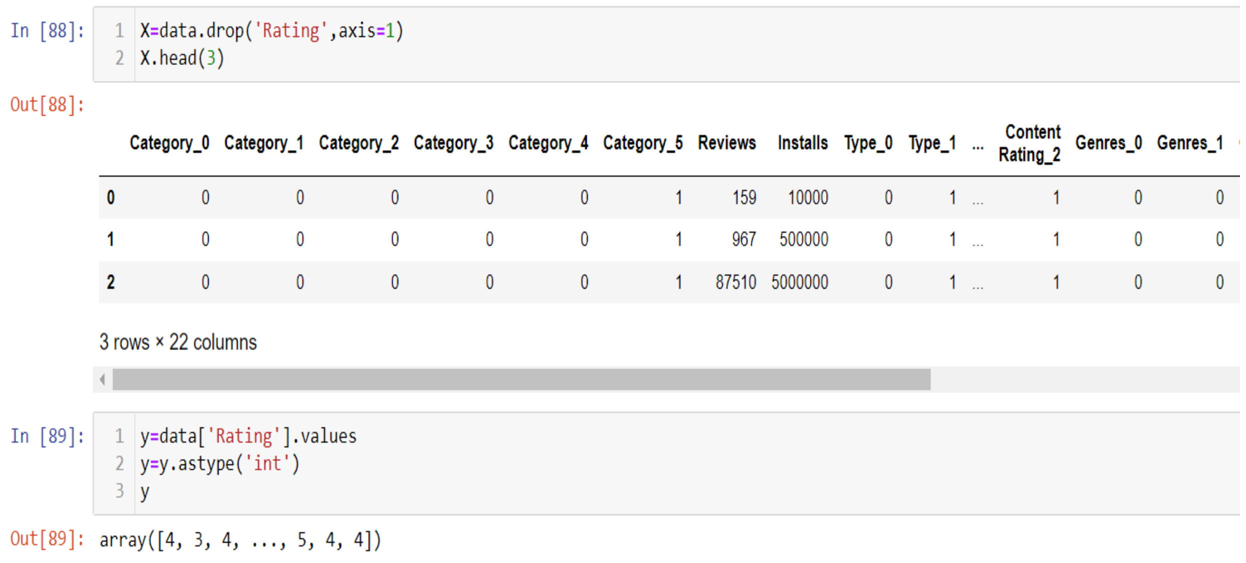
```
In [71]: data.drop(['App','Size','Price','Current Ver','Last Updated', 'Android Ver'], axis=1, inplace=True)
```

```
In [72]: df=data.copy()
```

```
In [73]: df.head()
```

Out[73]:

	Category	Rating	Reviews	Installs	Type	Content Rating	Genres	year_added	month_added
0	ART_AND_DESIGN	4.1	159	10000	Free	Everyone	Art & Design	2018	1
1	ART_AND_DESIGN	3.9	967	500000	Free	Everyone	Art & Design;Pretend Play	2018	1
2	ART_AND_DESIGN	4.7	87510	5000000	Free	Everyone	Art & Design	2018	8
3	ART_AND_DESIGN	4.5	215644	50000000	Free	Teen	Art & Design	2018	6
4	ART_AND_DESIGN	4.3	967	100000	Free	Everyone	Art & Design;Creativity	2018	6



```
In [88]: X=data.drop('Rating',axis=1)
         X.head(3)
```

Out[88]:

	Category_0	Category_1	Category_2	Category_3	Category_4	Category_5	Reviews	Installs	Type_0	Type_1	...	Content Rating_2	Genres_0	Genres_1
0	0	0	0	0	0	1	159	10000	0	1	...	1	0	0
1	0	0	0	0	0	1	967	500000	0	1	...	1	0	0
2	0	0	0	0	0	1	87510	5000000	0	1	...	1	0	0

3 rows × 22 columns

```
In [89]: y=data['Rating'].values
         y=y.astype('int')
         y
```

```
Out[89]: array([4, 3, 4, ..., 5, 4, 4])
```

6.6 TRAIN AND TEST SPLIT

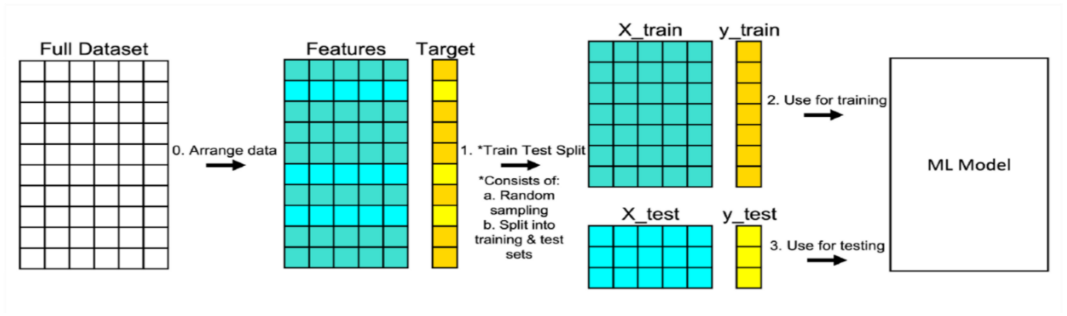
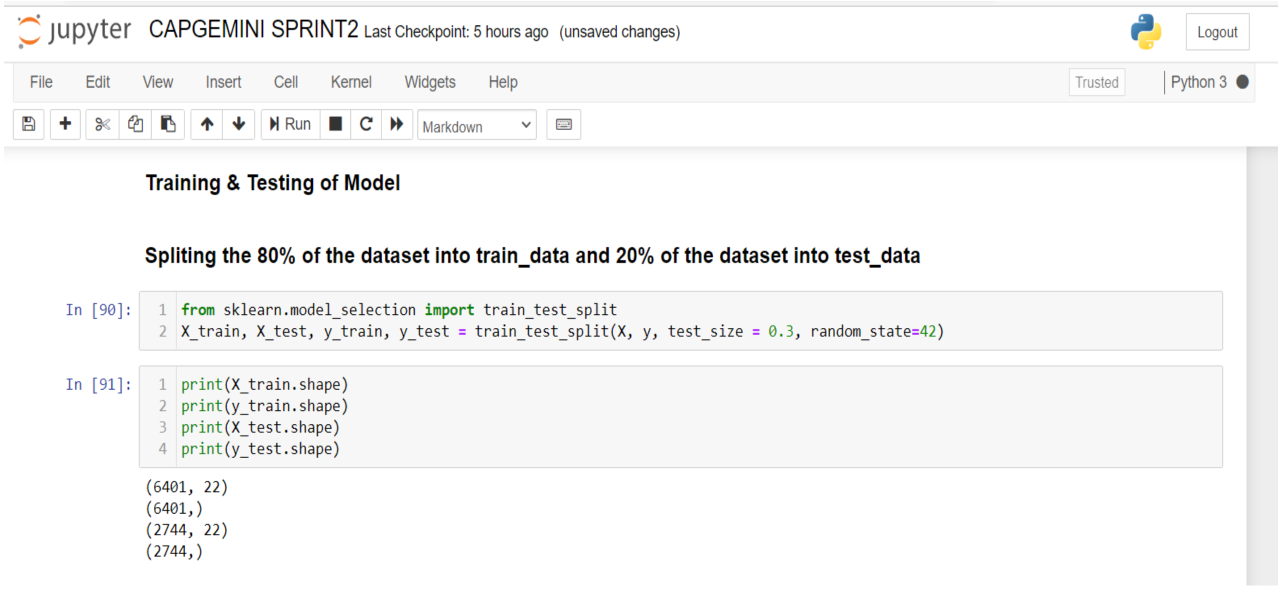


Figure 6.10 Train and Test Split

The train-test split is used to estimate the performance of machine learning algorithms that are applicable for prediction-based Algorithms/Applications. This method is a fast and easy procedure to perform such that we can compare our own machine learning model results to machine results. By default, the Test set is split into 30 % of actual data and the training set is split into 70% of the actual data.

We need to split a dataset into train and test sets to evaluate how well our machine learning model performs. The train set is used to fit the model, and the statistics of the train set are known. The second set is called the test data set, this set is solely used for predictions.



Training & Testing of Model

Spliting the 80% of the dataset into train_data and 20% of the dataset into test_data

In [90]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=42)
```

In [91]:
```python
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(6401, 22)
(6401,)
(2744, 22)
(2744,)
```

6.7 ALGORITHMS

6.7.1 Linear Regression:

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc. Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

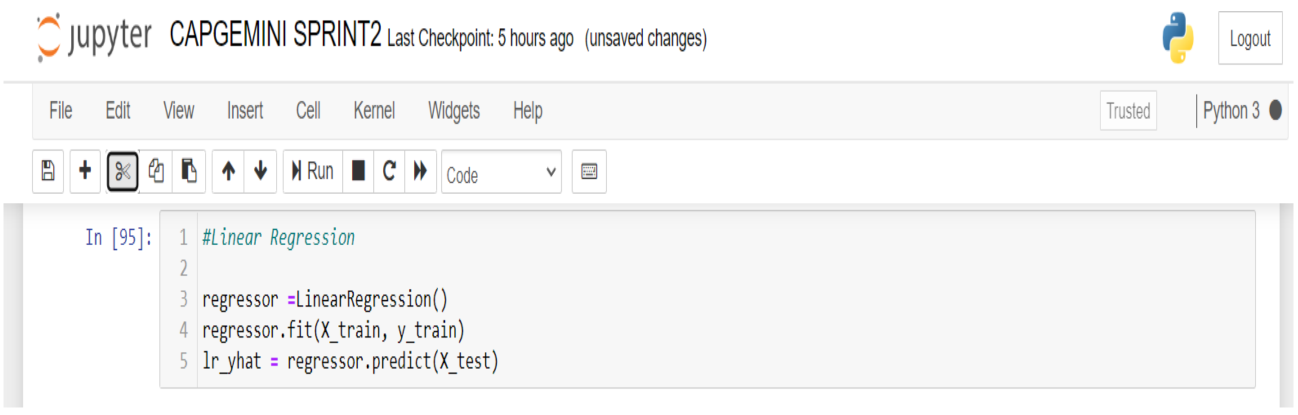
Mathematically, we can represent a linear regression as:

$$y= a_0+a_1x+ \varepsilon$$

Here

Y= Dependent Variable (Target Variable)
X= Independent Variable (predictor Variable)
a0= intercept of the line (Gives an additional degree of freedom)
a1 = Linear regression coefficient (scale factor to each input value).
ε = random error

The values for x and y variables are training datasets for Linear Regression model representation.


jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

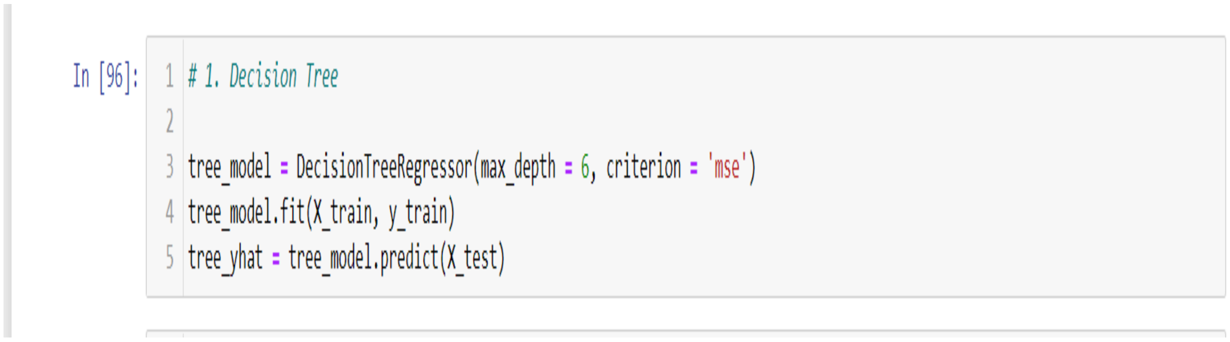
```
In [95]: #Linear Regression

         regressor =LinearRegression()
         regressor.fit(X_train, y_train)
         lr_yhat = regressor.predict(X_test)
```

6.7.2 Decision Tree

Decision Tree is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs, and utility. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

Decision Tree Regression**:**

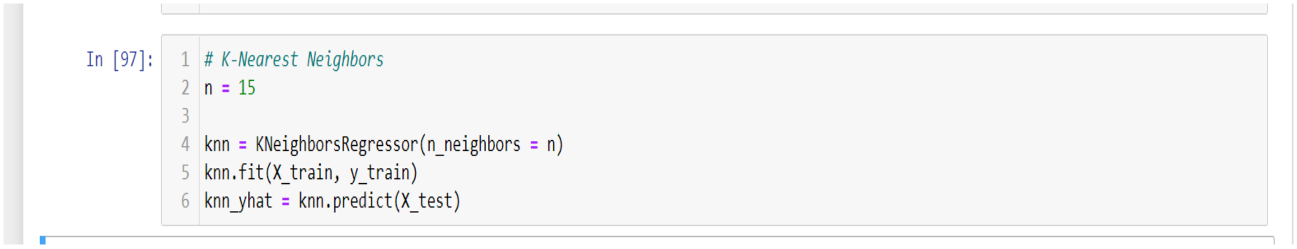
Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.



```
In [96]:  # 1. Decision Tree

          tree_model = DecisionTreeRegressor(max_depth = 6, criterion = 'mse')
          tree_model.fit(X_train, y_train)
          tree_yhat = tree_model.predict(X_test)
```

6.7.3 Knn

K Nearest Neighbors Regression first stores the training examples. During prediction, when it encounters a new instance (or test example) to predict, it finds the K number of training instances nearest to this new instance. Then predicts the target value for this instance by calculating the mean of the target values of these nearest neighbors.

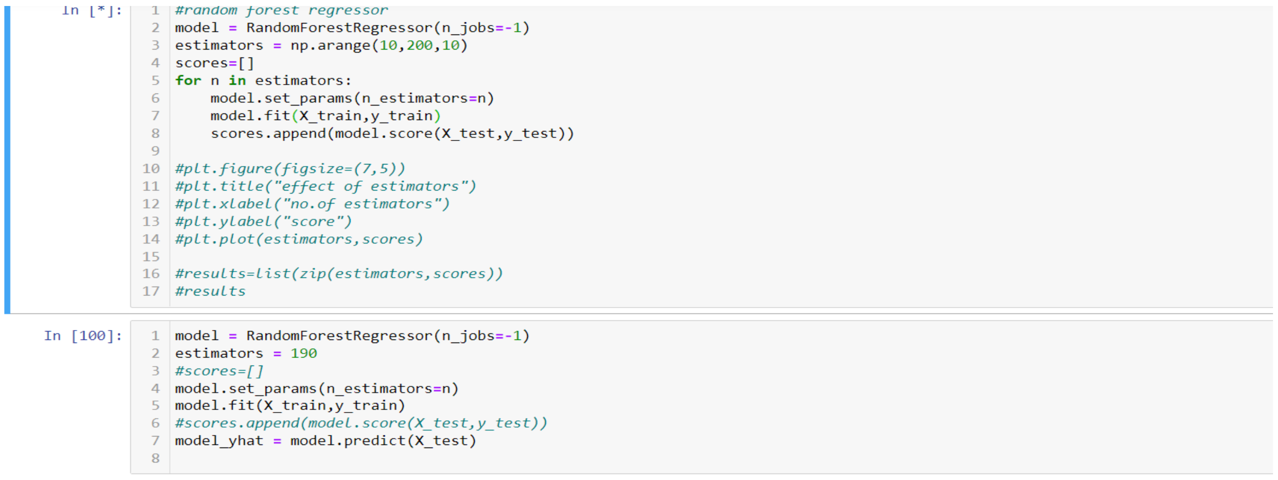


```
In [97]:
# K-Nearest Neighbors
n = 15

knn = KNeighborsRegressor(n_neighbors = n)
knn.fit(X_train, y_train)
knn_yhat = knn.predict(X_test)
```

6.7.4 Random Forest Regressor

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data, and hence the output doesn't depend on one decision tree but on multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs. This part is called Aggregation.

Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees. Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.



```
In [*]:
#random forest regressor
model = RandomForestRegressor(n_jobs=-1)
estimators = np.arange(10,200,10)
scores=[]
for n in estimators:
    model.set_params(n_estimators=n)
    model.fit(X_train,y_train)
    scores.append(model.score(X_test,y_test))

#plt.figure(figsize=(7,5))
#plt.title("effect of estimators")
#plt.xlabel("no.of estimators")
#plt.ylabel("score")
#plt.plot(estimators,scores)

#results=list(zip(estimators,scores))
#results
```

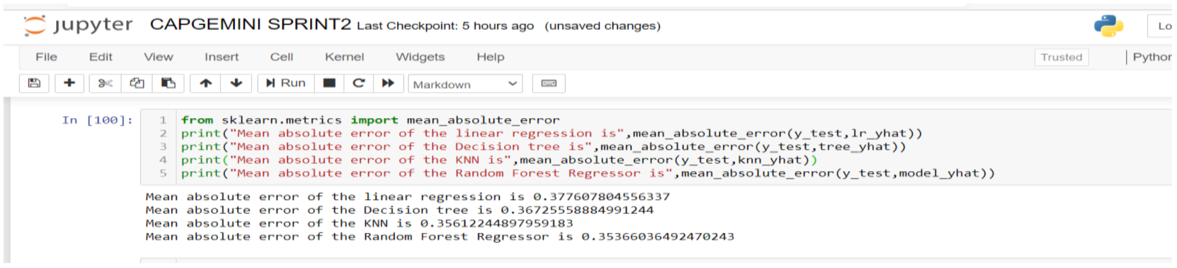
```
In [100]:
model = RandomForestRegressor(n_jobs=-1)
estimators = 190
#scores=[]
model.set_params(n_estimators=n)
model.fit(X_train,y_train)
#scores.append(model.score(X_test,y_test))
model_yhat = model.predict(X_test)

```

6.8 PERFORMANCE METRICS

6.8.1 Mean Absolute Error

As the name suggest, the metric is mostly focused on the errors. This means the difference between the actual observation and the predicted observation. MAE is mostly used to evaluate regression models such as linear models. Basically, all the observations are in continuous form. To implement it in any language, it follows the logic below in the order of the steps.

- Getting the Error, **Error** = Actual observation – predicted observation
  - When you get all the errors, you will realize that some errors are positive, and others are negative
- Getting the **Absolute Error** = |Error|
  - This step ignores the sign before the error. Treating the positive and negative errors observed as absolute
- Getting the **Average (Mean) of the absolute errors**
  - This involves adding all the errors and dividing with the total number of observations.


jupyter CAPGEMINI SPRINT2 Last Checkpoint: 5 hours ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Run
Markdown
In [100]:

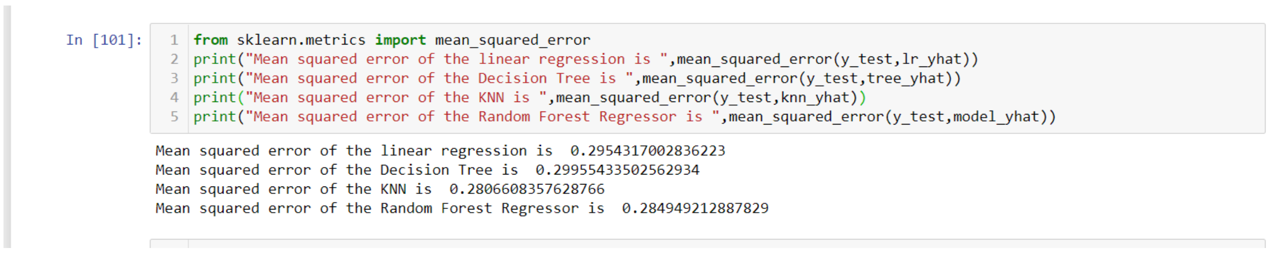
```
from sklearn.metrics import mean_absolute_error
print("Mean absolute error of the linear regression is",mean_absolute_error(y_test,lr_yhat))
print("Mean absolute error of the Decision tree is",mean_absolute_error(y_test,tree_yhat))
print("Mean absolute error of the KNN is",mean_absolute_error(y_test,knn_yhat))
print("Mean absolute error of the Random Forest Regressor is",mean_absolute_error(y_test,model_yhat))
```

```
Mean absolute error of the linear regression is 0.377607804556337
Mean absolute error of the Decision tree is 0.36725558884991244
Mean absolute error of the KNN is 0.35612244897959183
Mean absolute error of the Random Forest Regressor is 0.35366036492470243
```

6.8.2 Mean Square Error

It is simply the average of the square of the difference between the original values and the predicted values.

$$MSE = \frac{1}{N}\sum_{j=1}^{N}(predicted - input)^2$$

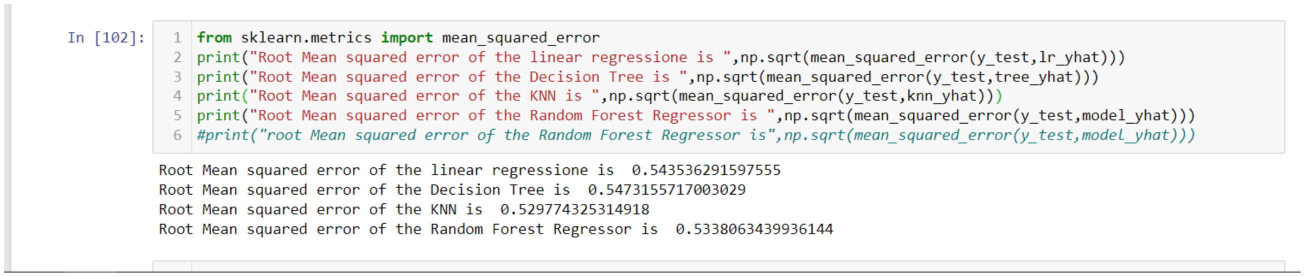


```
In [101]:
from sklearn.metrics import mean_squared_error
print("Mean squared error of the linear regression is ",mean_squared_error(y_test,lr_yhat))
print("Mean squared error of the Decision Tree is ",mean_squared_error(y_test,tree_yhat))
print("Mean squared error of the KNN is ",mean_squared_error(y_test,knn_yhat))
print("Mean squared error of the Random Forest Regressor is ",mean_squared_error(y_test,model_yhat))
```

```
Mean squared error of the linear regression is  0.2954317002836223
Mean squared error of the Decision Tree is  0.29955433502562934
Mean squared error of the KNN is  0.2806608357628766
Mean squared error of the Random Forest Regressor is  0.284949212887829
```

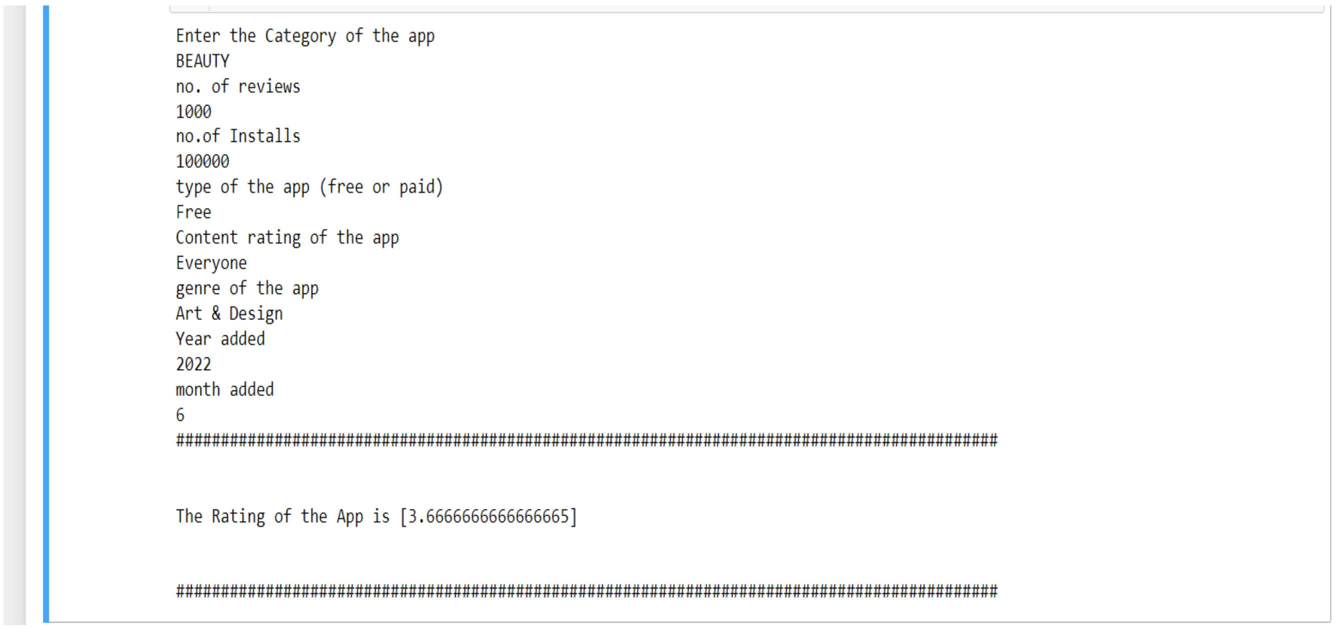
6.8.3 Root Mean Squared Error or RMSE

RMSE is the standard deviation of the errors which occur when a prediction is made on a dataset. This is the same as MSE (Mean Squared Error) but the root of the value is considered while determining the accuracy of the model.



```
In [102]:
from sklearn.metrics import mean_squared_error
print("Root Mean squared error of the linear regressione is ",np.sqrt(mean_squared_error(y_test,lr_yhat)))
print("Root Mean squared error of the Decision Tree is ",np.sqrt(mean_squared_error(y_test,tree_yhat)))
print("Root Mean squared error of the KNN is ",np.sqrt(mean_squared_error(y_test,knn_yhat)))
print("Root Mean squared error of the Random Forest Regressor is ",np.sqrt(mean_squared_error(y_test,model_yhat)))
#print("root Mean squared error of the Random Forest Regressor is",np.sqrt(mean_squared_error(y_test,model_yhat)))
```

```
Root Mean squared error of the linear regressione is  0.543536291597555
Root Mean squared error of the Decision Tree is  0.5473155717003029
Root Mean squared error of the KNN is  0.529774325314918
Root Mean squared error of the Random Forest Regressor is  0.5338063439936144
```



```
Enter the Category of the app
BEAUTY
no. of reviews
1000
no.of Installs
100000
type of the app (free or paid)
Free
Content rating of the app
Everyone
genre of the app
Art & Design
Year added
2022
month added
6
############################################################################################


The Rating of the App is [3.6666666666666665]


############################################################################################
```

7.CONCLUSION

After undergoing these algorithms and process, we concluded that our hypothesis is true. Meaning you can predict the app ratings, however significant preprocessing must be done before you start the regression processes. The Play Store apps data has enormous potential to drive app-making businesses to success. Actionable insights can be drawn for developers to work on and capture the Android market! This shows that given the Size, Type, Price, Content Rating, and Genre of an app, we can predict rating of the app. And also we can say if an app will have more than 100,000 installs and be a hit on the Google Play Store.

User reviews are limited to identifying polarity and subjectivity. However, the massive increase in review-based data implies a requirement to focus also on performing predictions. This process is challenging yet fruitful, as user reviews are qualitative while ratings are essentially quantitative. The numeric scoring of apps within the Google App store could also be biased and overrated because higher ratings given by users potentially attract several new users disproportionately. This study therefore investigated the utilization of ensemble classifiers to predict numeric ratings for Google Play store apps supported the user reviews for those apps. Several ensemble classifiers were investigated to guage their performance on the reviews scraped from the Google App store. Future work includes the implementation of the deep learning technique to predict numeric rating.

8.REFRENCES

[1]. Statista, Number of available application in the Google Play store from December 2009 to March 2019, https://www.statista.com/ statistics/266210/number of-available-applications-in-the-goggle- store/, Online: accessed 22 May 2019.

[2]. Irjet.net,https://www.irjet.net/archives/V7/i12/IRJET-V7I1248.pdf

[3]. J. Horrigan, Online shopping, pew internet and American life project, Washington, DC, 2018, http://www.pewinternet.org/Repor ts/2008/Online Shopping/01-Summary-of-Findings.aspx Online: accessed 8 Aug. 2014.

[4]. D. Pagano and W. Maalej, User feedback in the app store: an empirical study, in Proc. IEEE Int. Requirements Eng. Conf. (Rio de Janeiro, Brazil), July 2013, pp. 125–134.

[5]. T. Chumwatana, Using sentiment analysis technique for analyzing Thai customer satisfaction from social media, 2015.

[6]. T. Thiviya et al., Mobile apps' feature extraction based on user reviews using machine learning, 2019.