INTRODUCTION TO DATA SCIENCE

IDS PROJECT REPORT

Team members:

* ROHIT VISHWAKARMA ,SRN: PES1201800152
* DURJOY GHOSH ,SRN: PES1201802124
* SAURAV R , SRN : PES12018001511

PROJECT DATASET: Google playstore Appdata



# **Aim of the project**

The sole aim of our project is to find out what kind of apps are ruling the current market and what trends are they exhibiting.We will try to do the necessary cleaning on the data and try to get some useful insights from the data.

# **Data Cleaning**

Our data set was taken from the Kaggle datasets . This dataset originally had the around

Number of rows = 10841 rows

Number of columns = 13 columns

data.columns we got

App object

Category object

Rating float64

Reviews object

Size object

Installs object

Type object

Price object

Content Rating object

Genres object

Last Updated object

Current Ver object

Android Ver object

dtype: object

Here we see that there is a lack of cleaning in the data set .It needs some cleaning.

The Review , size and installs had to be in the int 64 but they are shown as object .

Lets explore each of them

print(data['Type'].value\_counts())

output :

Free 10039

Paid 800

0 1

Name: Type, dtype: int64

print(data['Content Rating'].value\_counts())

Everyone 8714

Teen 1208

Mature 17+ 499

Everyone 10+ 414

Adults only 18+ 3

Unrated 2

In statistics, **missing data**, or **missing values**, occur when no **data value** is stored for the variable in an observation. **Missing data** are a common occurrence and can have a significant effect on the conclusions that can be drawn from the **data**.

total = data.isnull().sum().sort\_values(ascending=False)

percent = (data.isnull().sum()/data.isnull().count()).sort\_values(ascending=False)

missing\_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])

missing\_data.head(6)

This made us know what percent of data is missing in every column.

|  | **Total** | **Percent** |
| --- | --- | --- |
| **Rating** | 1474 | 0.135965 |
| **Current Ver** | 8 | 0.000738 |
| **Android Ver** | 3 | 0.000277 |
| **Content Rating** | 1 | 0.000092 |
| **Type** | 1 | 0.000092 |
| **Last Updated** | 0 | 0.00000 |

The simplest way to deal with missing values in the data set is to drop the rows

which contains the missing values .The above table shows the number of the

missing values for each Rating , Current Ver , Android Version ,Content Rating

etc .

Why did we do this method of treating the missing values ?

Answer:

Pros: Complete removal of data with missing values results in robust and highly accurate model. Deleting a particular row or a column with no specific information is better, since it does not have a high weightage.

data.dropna(how ="any", inplace = True)

After droping the missing values. I again Plot the Percentage missing value in each of them.

total = data.isnull().sum().sort\_values(ascending=False)

percent = (data.isnull().sum()/data.isnull().count()).sort\_values(ascending=False)

missing\_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])

missing\_data.head(6)

|  | **Total** | **Percent** |
| --- | --- | --- |
| **Android Ver** | 0 | 0.0 |
| **Current Ver** | 0 | 0.0 |
| **Last Updated** | 0 | 0.0 |
| **Genres** | 0 | 0.0 |
| **Content Rating** | 0 | 0.0 |
| **Price** | 0 | 0.0 |

After remove missing data,

our data contain 9,360 records with 13 fields

print(data.shape)

Number of Rows and colums : (9360, 13)

After this missing value treatment we convert the object types into

int so we did the following

data['Reviews'] = data['Reviews'].apply(lambda x: int(x))

data.Installs = data.Installs.apply(lambda x: x.replace(',',''))

data.Installs = data.Installs.apply(lambda x: x.replace('+',''))

data.Installs = data.Installs.apply(lambda x: int(x))

result

App object

Category object

Rating float64

Reviews int64

Size object

Installs int64

Type object

Price object

Content Rating object

Genres object

Last Updated object

Current Ver object

Android Ver object

dtype: object

Now you can see the changes .

# C:\Users\ROHIT V\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\B495A691.tmpData Visualization using various Python Libraries .

Firstly we begin by finding the count plot of the available categories.

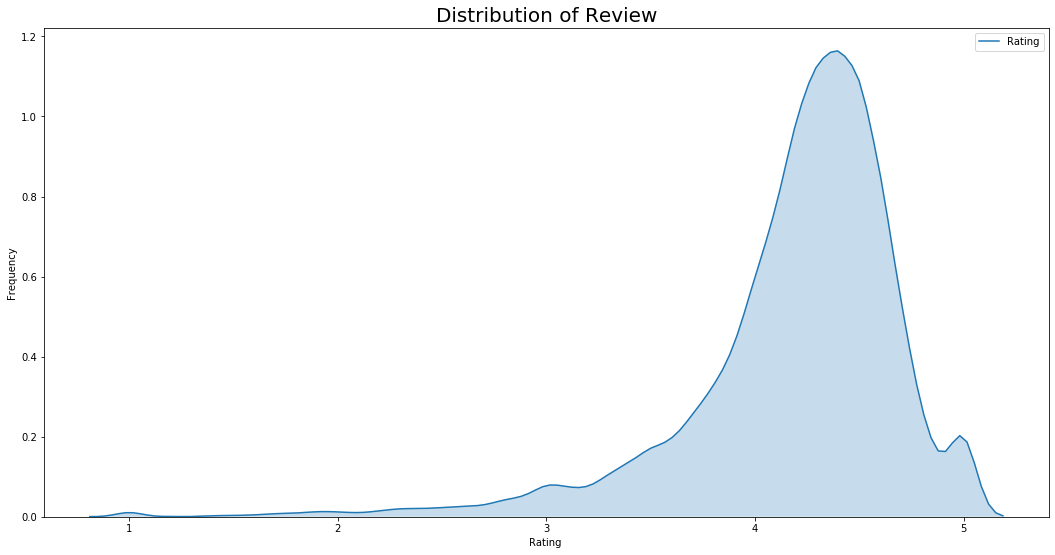
We can clearly see from the count plot that the mostly available app on the Google play stores are of the Category Family and then followed by Games. This gives an insight about the users of the Google play stores that use mobile for playing games and using Family category apps .

# The distribution of Rating.

Use a KdePlot for doing the frequency Distribution .

plt.figure(figsize=(18,9))

correct\_data = data.loc[data["Rating"]<=5]

sns.kdeplot(correct\_data.Rating ,shade = True)

From the kdeplot we can treat the rating given to an app as the Random variable. This basically represented the frequency distribution of the rating for the given dataSet .The Average rating of the playstore is around 4 which is high and thus we can say that on an average the Google Playstores keeps pretty good apps .

# C:\Users\ROHIT V\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\6DEA5076.tmpThe Distribution of Reviews.

The X axis here in the frequency distribution of the Frequency(Number of the installs) and Reviews shows a clear insight that the most of the apps in the playstore get the Review of less than 1 million . But we can say that the Good and popular apps obviously get more Reviews as seen in the plot .

# C:\Users\ROHIT V\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\BECF088E.tmpBox Plot for the data set

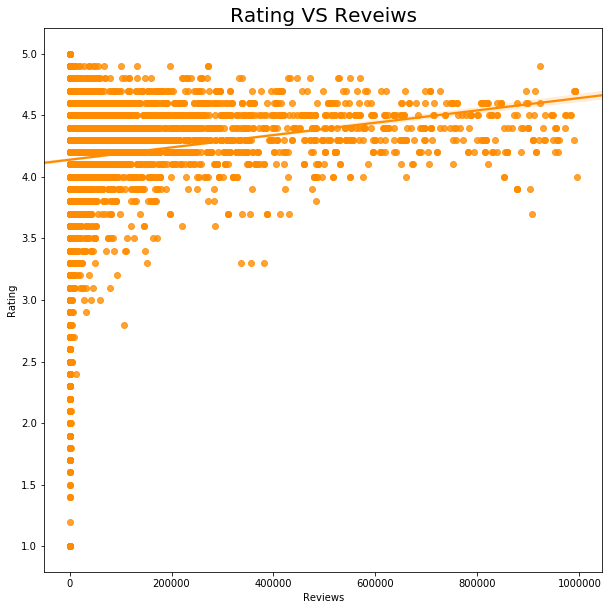
The above box plot just tries to represent the rating of application in each category is not different too much. This can be seen in the plot as mid 50 % of the apps of each category of lie nearly the same rating .

# C:\Users\ROHIT V\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\BEA5495D.tmpCount Plot for the Content rating of Rating>3

Here also we can see that the number of the apps given by each of the Content Rating in the range of 3.0 to 5.0 are mostly given Everyone followed by Teens .This plot

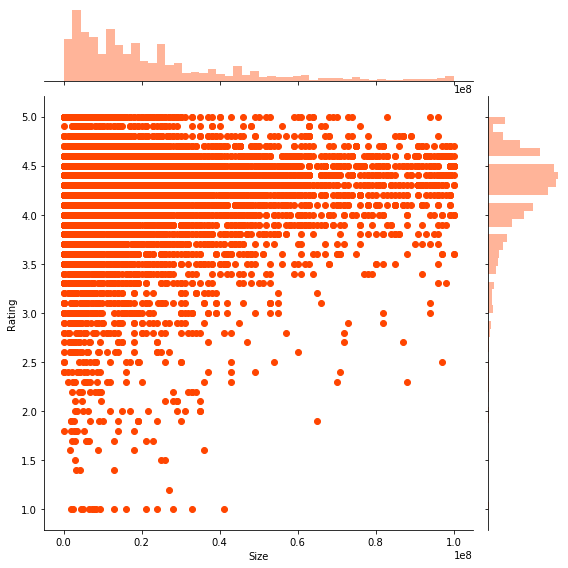
Also looks fairly normal .

# Rating Vs Reviews (Finding a relation between them)



We had seen in the last frequency distribution of reviews only some apps got the reviews of 1 Million .The insight that we can draw from plot is that if an application is very well known then we can say that the application will get high rating . The apps who have more reviews in generally have good rating .

# Joint plot for finding relation between Rating and Size



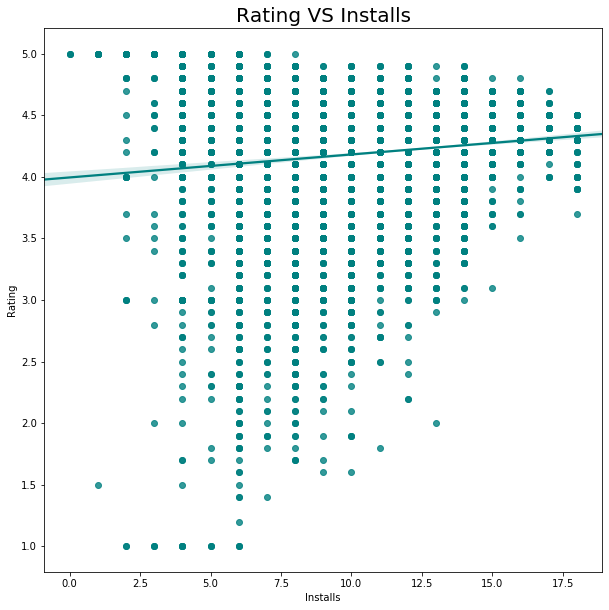
Above we have used Jointplot to find a correlation between different columns in the dataset. We have chosen the Rating and Size .

The insight that we found is that as size of the app increases then the rating of the Apps increases but the data points become less if the App is really very good in spite of having a huge size then it also gets good rating.

# C:\Users\ROHIT V\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\CCFC6953.tmpFree Apps and Paid Apps

The finding we found is most of the Apps in the playstore are free in the nature.

# Using Regplot for Rating and Installs

 From the above plot we understand that there is a linear relationship between the Ratings and Installs .

As the number of Installs increase the Rating of that Apps

also increases .

# Hypothesis Testing

Let’s put up some hypothesis test into this dataset of Google PlayStore

alpha = 0.05

Population : All apps in playstore. Parameter of Interest : µ .

**Hypothesis : Is the average rating for all apps is more than 4 ?.**

Thus we have

**H0 : µ ≤ 4 (NULL HYPOTHESIS)**

**H1 : µ > 4 (ALTERNATE HPOTHESIS)**

clearly we have the sample standard deviation = 0.515263 (s) sample size of n = 9360

We will use a Z score to calculate the p value

Before that we mention that

x\_bar ~ N(µ,σ^2/n) Normal distribution . but in our case we don’t have the σ

so using s ~ σ we get x\_bar ~ N(4,0.00002836)

z\_score for this data set comes out to around **35.6807511**

**The value of z is 35.67495. The value of p is < .00001. The result is significant at p < .05.**

**So p value < 0.00001**

**Therefore p value < 0.05 we reject the H0 hypothesis.**

# Thank You

# Tha



