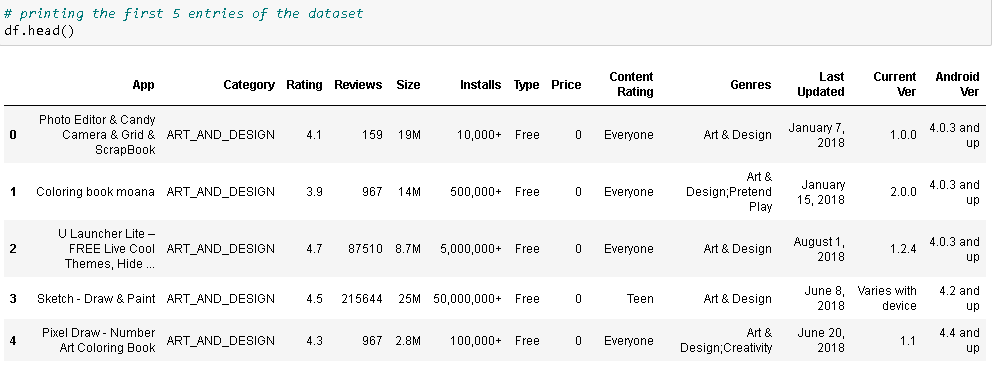
**1. Install the necessary libraries and read the provided dataset. (1 point)**

****



Approach Used: Imported the following necessary libraries:

* Pandas library contains functions to handle dataframes
* Numpy library contains mathematical functions
* Scipy.stats contains large number of functions for statistical methods
* Matplotlib and seaborn is used for plotting different graphs
* Test\_train\_split is used to split the independent and target variables into test and train datasets
* We have imported a number of classifier modules which we would be using in predicting the target variables

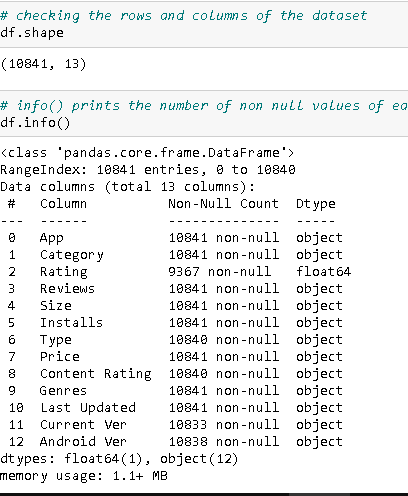
We have read the csv file using read\_csv() function from pandas library

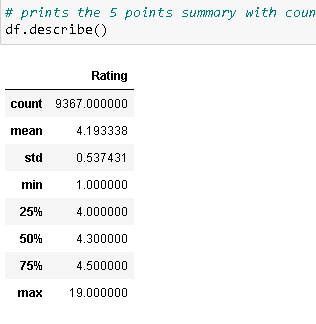
Insights and Inferences:

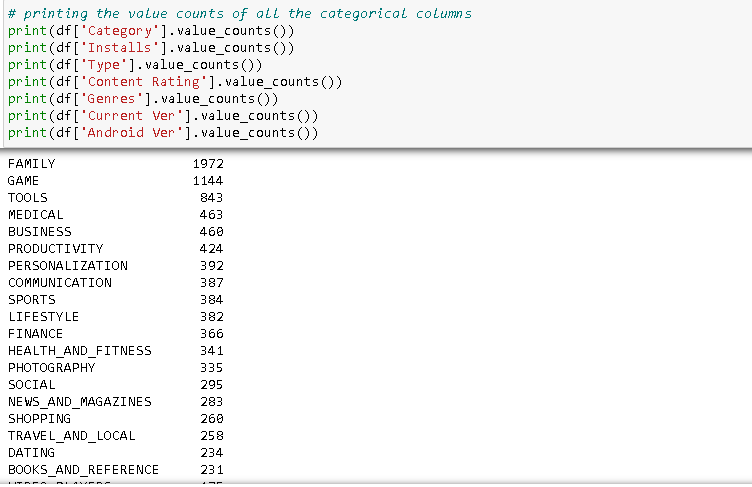
* Genres, Current Ver and Android Ver columns seem to be heavily cluttered
* Columns like size, installs, price can be used as integers but due to respective special characters, we have them as string.
* In the first view, Categories and genres seem to be heavily correlated (genres just contain subcategories in addition)

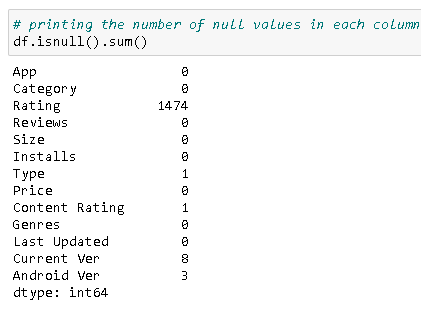
**2. EDA and Preprocessing (27 points)**

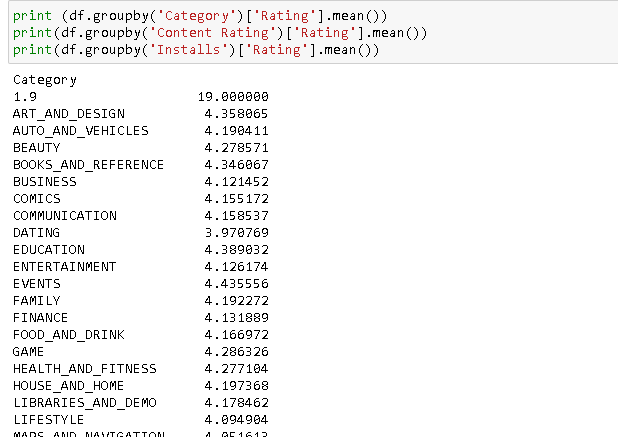
a. Check the info and summary statistics of the dataset. List out the columns that need to be worked upon for model building. (2 points)











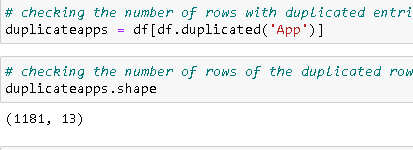
Approach Used: to check info and summary statistics of the dataset, we have used the following approaches

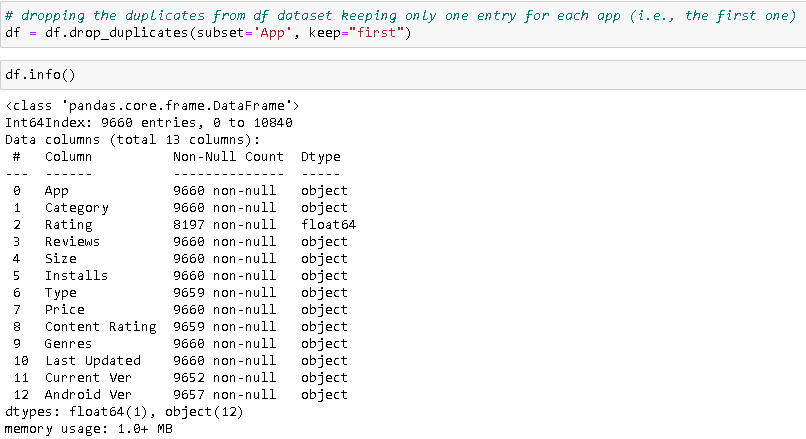
* Find out total number of rows and columns
* Check out info of the dataframe to know datatypes of each attribute, and the number of non-null values they contain
* Use describe() function to get 5 point summary for numerical attributes
* Take a look at the value\_counts at all the supposedly categorical attributes
* Check out the mean rating for each categories of some crucial categorical attributes
* Find out the total number of null entries for each column

Insights and Inferences:

* Columns like size, installs, price can be used as integers but due to respective special characters, we have them as string.
* Some columns contain null values. Ratings column contains a lot of null values
* Reviews can easily be converted to numerical value
* There is an outlier 19 in 'Rating' column
* Category contains 1.9 as an entry, which seems incorrect.
* Installs column has 'Free' as an entry which needs to be worked upon
* Type column contains one entry as 0, which has to be replaced with 'Free'
* Current ver and android ver columns seem too cluttered to be taken into the model.
* We can now confirm the previous insight that Genres column just contain subcategories in addition to the values of Category column
* No substantial pattern in rating for respective categories (of Content Rating, Installs and Category) is observed
* **Columns that need to be worked upon: Category; Rating; Reviews; Installs;Size;Price; Content Rating; Type**

b. Check if there are any duplicate entries for the apps (1 point)





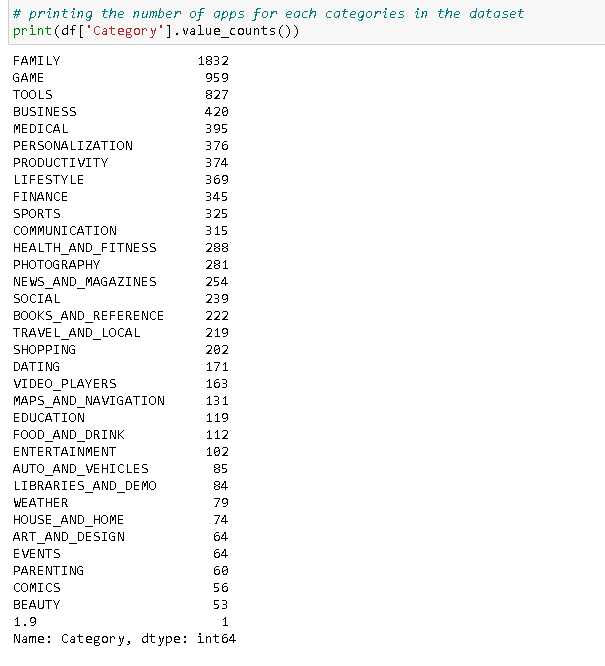
Approach Used:

* We first find out all the rows which have duplicate entries for Apps and store them to duplicateapps dataframe
* We can then find out the number of duplicate entries (for apps) by using duplicateapps.shape
* We drop the duplicated by using drop\_duplicates() function based on the column Apps, keeping the first henry for each App

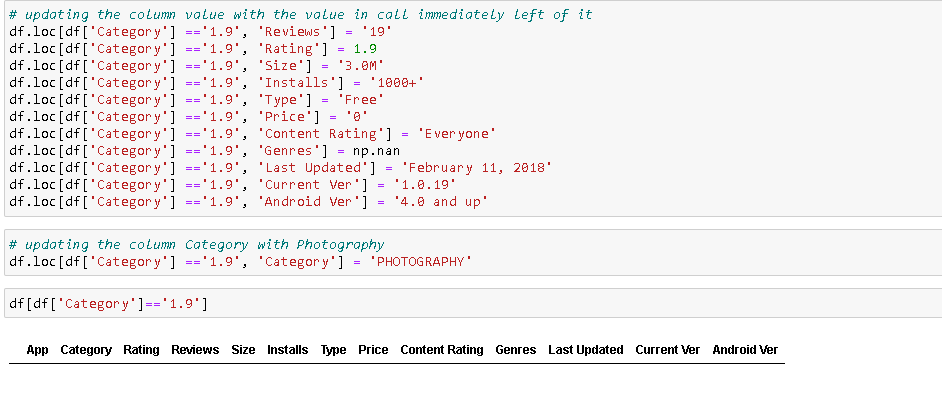
Insights and Inferences:

* There are 1181 duplicate entries
* After removing them, we have a total of 9660 rows left

c. Check if there are any wrong values in the ‘Category’ column and impute them with relevant values. (2 points)







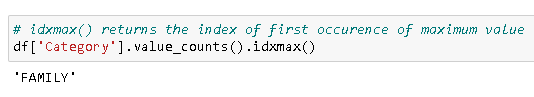
Approach Used:

* We print all the distinct values in the Category attribute and number of rows pertaining to them.
* 1.9 is an odd entry for a Category column, so, we take a look at the row containing that value
* All the entries in this row, starting from categories, have been shifted one column left.
* We conclude from the app name that the category for this app is PHOTOGRAPHY
* We then replace all the values of the row with their corrected values (in effect, we shift them one column right and replace the Category column with PHOTOGRAPHY)

Insights and Inferences:

* Category with 1.9 as an entry needs to be worked upon
* Seems that all the entries starting from the Category column are shifted one cell left.
* From the name of App, we can conclude that Category for this app is Photography
* We will update all the columns in this row with proper values

d. Which category has the highest number of apps? (2 points)



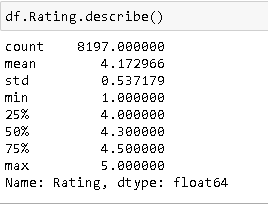
Approach Used:

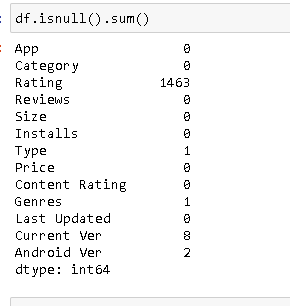
* We use the idxmax() function which returns the row with the highest value counts

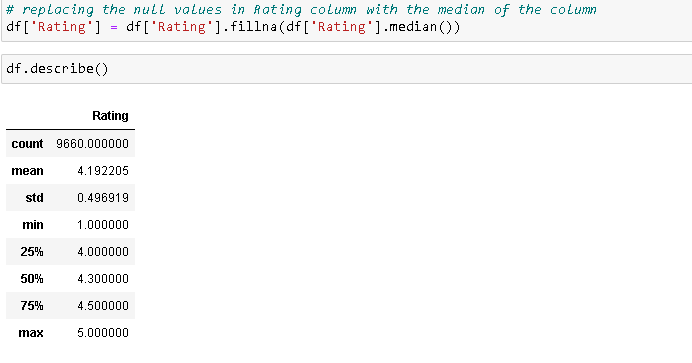
Insights and Inferences:

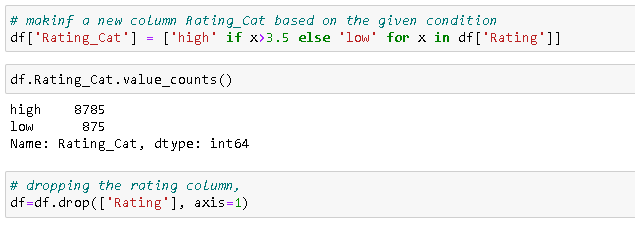
* Apps in Family category are highest in number

e. Check the distribution of rating column and convert ratings into two categories and save it in the data frame as ‘Rating\_cat’ ( high = +>3.5 and remaining as low) (2 points)









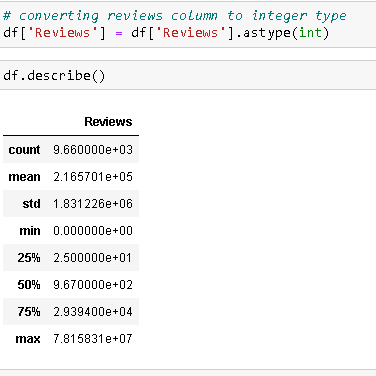
Approach Used:

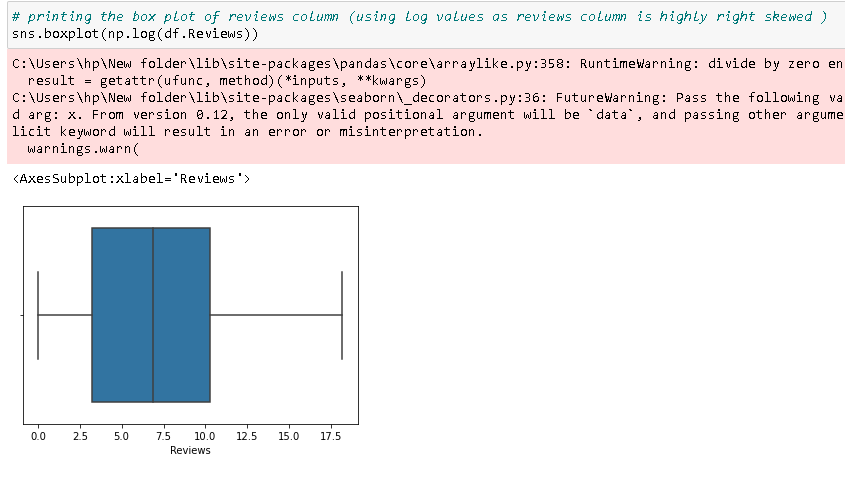
* Use describe function to get an idea about the distribution of Ratings
* We then count the number of null values
* We fill null values with median values
* Then we make a new column using the given condition by using if- else condition
* We drop the Ratings column

Insights and Inferences:

* Most of the Ratings are above 4 point
* Ratings column contains an abnormally high number of null values
* Number of Apps in the high rating category are 8785 and those in low category are 875.

f. Convert the ‘Review’ column to a numerical column and impute invalid values if there are any. (1 point)





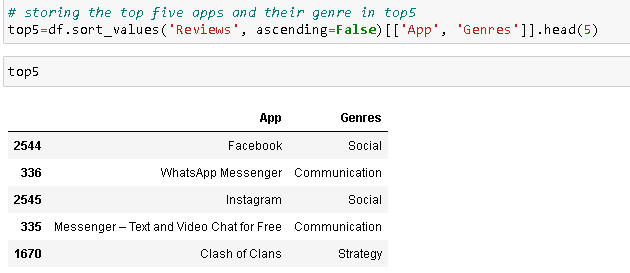
Approach Used:

* We convert Reviews to int variable type by using astype() function
* To check the distribution of the Reviews attribute, we use the describe function. Additionally we can visualise it using Boxplot
* We use the log values of reviews to get a rational view of the distribution of the attribute.

Insights and Inferences:

* Reviews column seem to have a heavy right skew so much so that even the log (review) cannot remove the right skew during visualisation.

g. Name the top 5 apps which have the highest number of reviews and their genre? (1 point)



Approach Used:

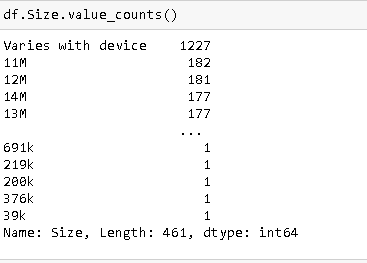
* We sort the dataframe by Reviews (descending) and extract the App and Genre with highest 5 reviews
* We save it to dataframe top5

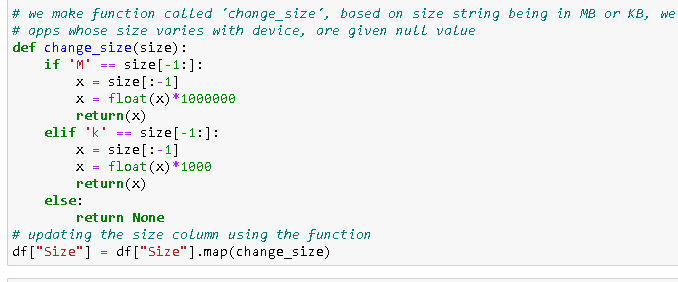
Insights and Inferences:

* As expected, Facebook and Whatsapp are among the top two reviewed apps.
* In Fact Instagram and Messenger also belong to Facebook. Top 4 highest reviewed apps belong to Facebook
* We can also infer that there is a strong correlation between number of installs and reviews (Assuming these apps are also among the most downloaded apps)

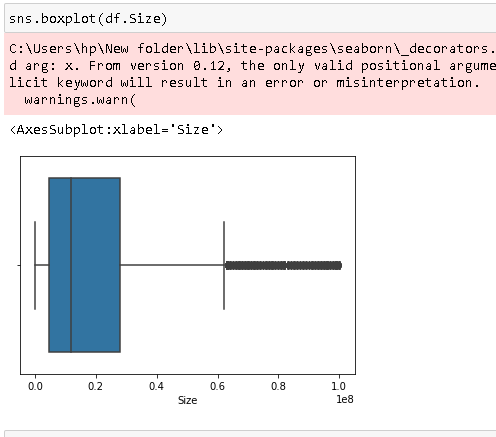
h. Make the values of ‘Size’ as integers by replacing M and K with correct values.

Convert all the values to numeric and make invalid values to NaN. ( 3 points)









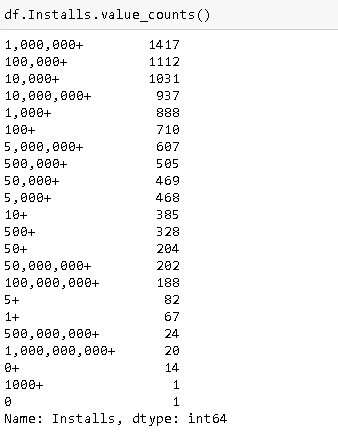
Approach Used:

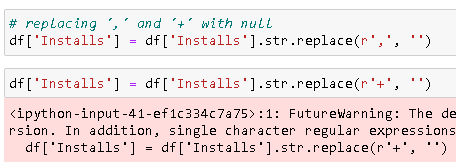
* We first see the value counts of size, to see if there are any peculiar values in this column
* We make a function called 'change\_size', based on size string being in MB or KB, we convert it to bytes.Apps whose size varies with device, are given null value
* We update the null values with ‘ffill’ method (the empty space gets updated with the value in the trailing row).We know that the given dataframe contains rows in such a way that apps with same category are placed together, Thus assuming that apps from similar category tend to have similar size dimension, we use ‘ffill’ method.

Insights and Inferences:

* 1227 apps’ Size vary with device
* We know that the given dataframe contains rows in such a way that apps with same category are placed together, Thus assuming that apps from similar category tend to have similar size dimension, we use ‘ffill’ method.

i. Remove “,” and “+” from the values of the “Installs” column and change the datatype. (3 points)







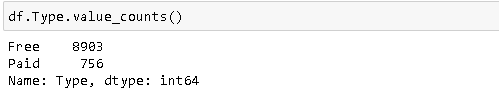
Approach Used:

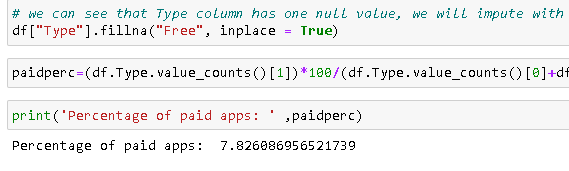
* We use str.replace() function to replace ‘,’ and ‘+’ with emptiness
* We convert values of Installs column to float

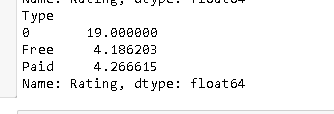
Insights and Inferences:

* Apps’ installs range from 0 installs to above 100000000 downloads

j. What is the percentage of paid apps in the data? (2 points)







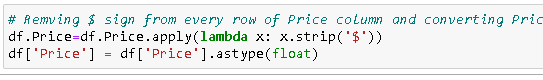
Approach Used:

* We first use value\_counts() to get an intuition about the paid apps and free apps
* We fill the null values with median (‘Free’) value.
* We define a variable paidperc which uses formula (number of paid apps/ (total number of apps))
* We extract the values of ‘Number of paid apps’ and ‘Number of free apps’ using the array :df.Type.value\_counts()[0/1]

Insights and Inferences:

* value\_counts function suggests that Type column contains one null value
* Only 7.82 % apps are paid, rest are free.
* We also see that paid apps have higher average rating compared to free apps.

k. Remove the “$” sign the “Price” column values and make it a numerical column. (2 points)

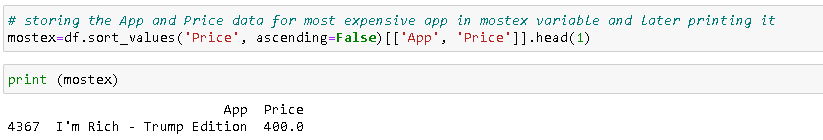


Approach Used:

* We use strip(‘$’) function which removes the given string from a string
* Following it, we convert Price to float datatype.

Insights and Inferences:

l. Which is the most expensive app and how much does it cost? (2 points)



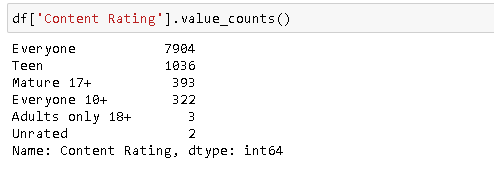
Approach Used:

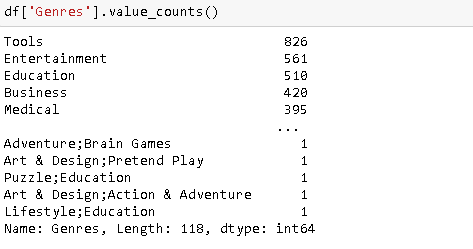
* We sort the dataframe in descending order. We save the first row(i.e., the one with highest price)
* We save this value to mostex

Insights and Inferences:

* I’m Rich-Trump Edition is the most expensive app which costs $400

m. Drop columns that you feel can not be used for model building. ExampleApp, Content Rating, Genre, Last updated, Current Ver, and Android Ver columns from the final data frame. (2 points)







Approach Used:

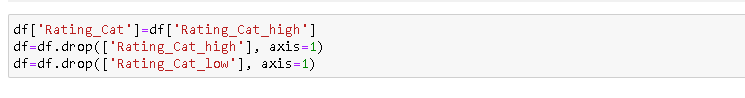
* We take a look at the Genres and Content Rating to see if we can use them for model building
* We then drop the unnecessary columns i.e., Genres, Last Updated, Current Ver, Android Ver, App. We use simple drop() function with inplace=True

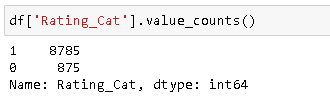
Insights and Inferences:

* We also already concluded that ‘genres’ are merely mention the subcategories for the apps, building upon the ‘category’ data.
* We had already concluded that ‘Current ver’ and ‘Android Ver’ are extremely cluttered and redundant so much so that they cannot be converted to any meaningful independent variable for model building, So we drop them.

n. Encode categorical column (Type, Rating\_categories, Category) [ Hint - use get\_dummies] (2 points)







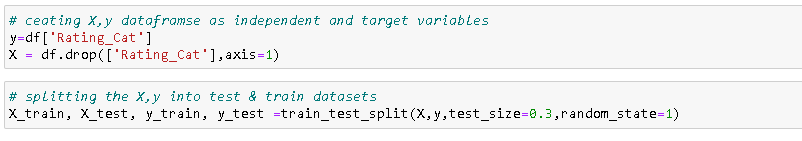
Approach Used:

* We use get\_dummies function for encoding
* After encoding, since Rating\_Cat is the target attribute, we need one column.
* We assign Rating\_Cat\_High to Rating\_Cat column and drop both the other columns Rating\_Cat\_High and Rating\_Cat\_Low.
* It means that apps with rating higher than 3.5 are valued 1 and those with rating lower than 3.5 are valued 0.

Insights and Inferences:

* 8785 apps have ratings higher than 3.5 anad 875 apps have lower rating than 3.5

**3. Prepare data for modeling. (2 points) a. Segregate dependent variable and independent features into two separate variables and split the data into train and test set [ Use 70:30 split ]**

****

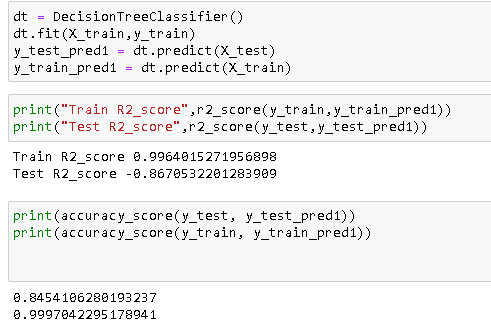
Approach Used:

* We make X dataframe that contains independent variables
* We make y that contains Rating\_Cat column
* We use train\_test\_split() to split the X and y into train and test datasets.

Insights and Inferences:

**4. Build a classifier model to predict the rating category (Rating\_cat - high or low) using the following algorithm and make predictions on the test data. Evaluate the model and report your results. (16 points - 4 points each)**

a. Decision Tree Classifier



Approach Used:

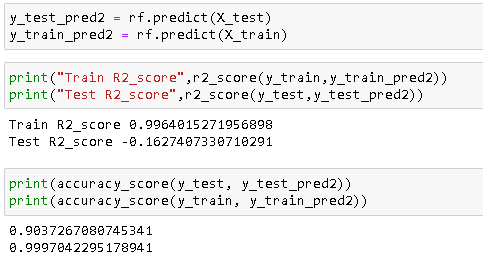
* We assign DecisionTreeClassifier() model to dt train it on X\_train and y\_train data
* We then use the model fit to predict target variable on both test as well as train set
* We calculate r2 score and accuracy score for predicted target variable vs actual target variable

Insights and Inferences:

* We can see that there is overfitting on the train data, so we can expect a bit worse accuracy score then the train set accuracy. (Accuracy for test data: 84.5%, accuracy for test data: 99.9%)

b. Random Forest model





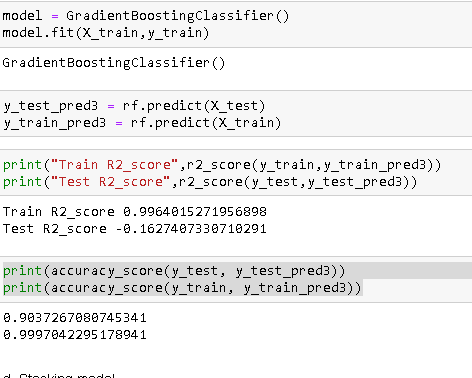
Approach Used:

* We assign RandomForestClassifier() model to rf and train it on X\_train and y\_train data
* We then use the model fit to predict target variable on both test as well as train set
* We calculate r2 score and accuracy score for predicted target variable vs actual target variable

Insights and Inferences:

* We can see again that there is overfitting on the train data, but this time the accuracy of test set has improved by 6%.. (Accuracy for test data: 90.3%, accuracy for test data: 99.9%)

c. Gradient Boosting model



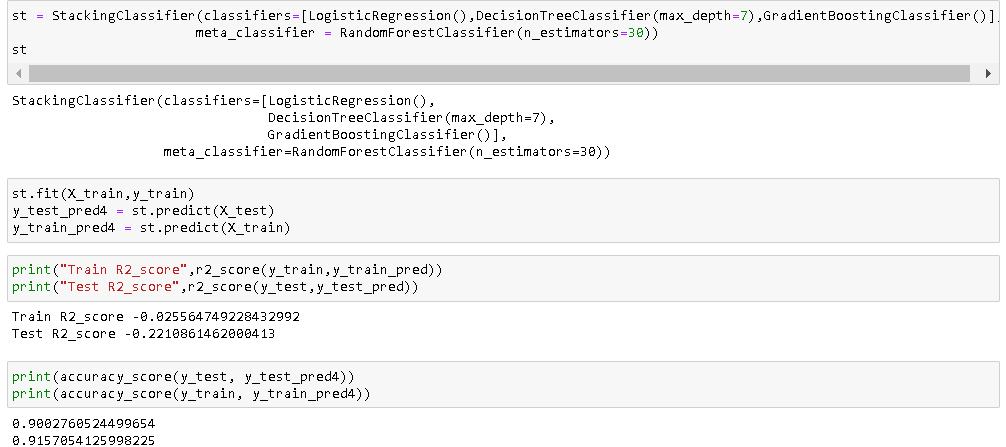
Approach Used:

* We assign GradientBoostingClassifier() model to ‘model’ and train it on X\_train and y\_train data
* We then use the model fit to predict target variable on both test as well as train set
* We calculate r2 score and accuracy score for predicted target variable vs actual target variable

Insights and Inferences:

* We can see again that there is overfitting on the train data..Accuracy of test data is exactly same as the Random forest model. (Accuracy for test data: 90.3%, accuracy for test data: 99.9%)

d. Stacking model



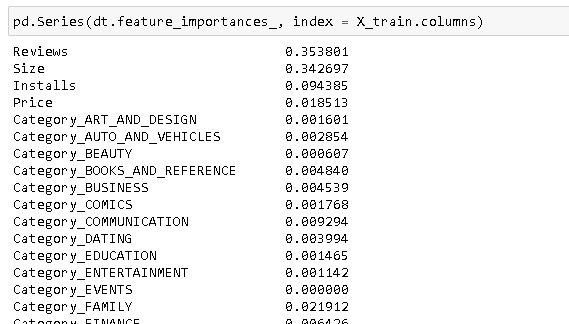
Approach Used:

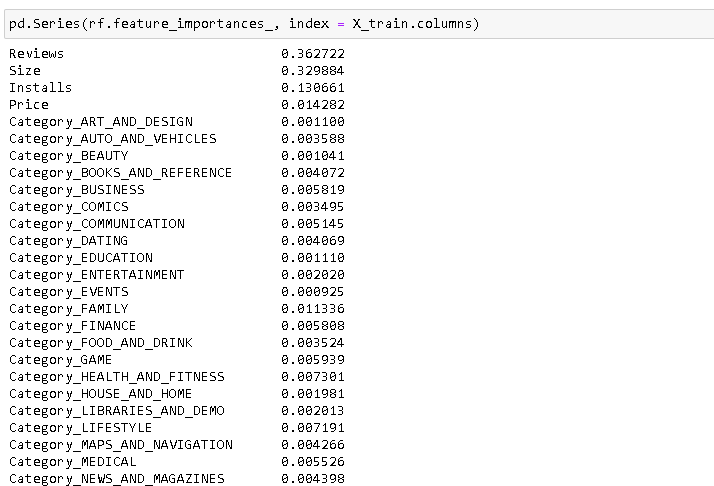
* We first define a model ‘st’ that contains LogisticRegression(), DecisionTreeClassifier() and GradientBoostingClassifier as ensembles and assign RandomForestClassifier() as meta\_classifier
* We train it on X\_train and y\_train data
* We then use the model fit to predict target variable on both test as well as train set
* We calculate r2 score and accuracy score for predicted target variable vs actual target variable

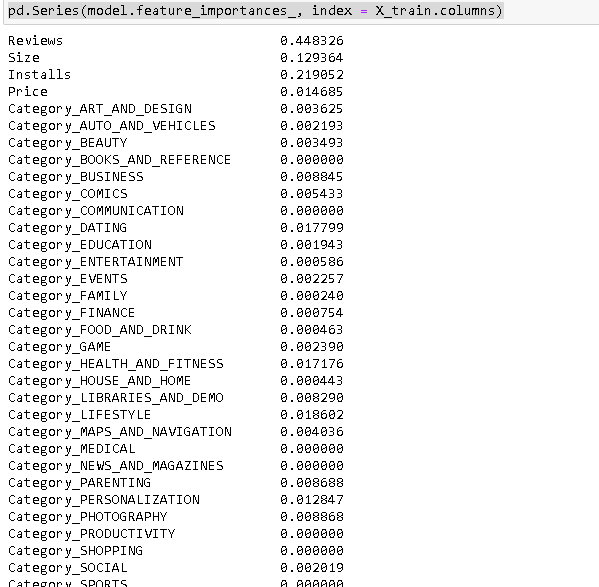
Insights and Inferences:

* We can see overfitting on the train data has reduced substantially, accuracy score of test set remains similar. So, this can be taken as reliable model compared to other 4.

**5. Check the importance of different features by using model.feature\_importances\_ function in Python ( 2 points)**

****

****

****

Approach Used:

* We have used model.feature\_importances\_ to print the importance of different features in predicting the rating
* We couldn’t print the feature importance of StackingClassifier as it comes from mlxtend library which doesn’t contain the feature\_importance\_ function.

Insights and Inferences:

* However, the feature importance tables for the three models suggest strongly that more than 80% of the explaination is done by the three columns that are: Reviews, Installs and Size.

**6. Comment on your results and findings from the above analysis. What can you infer about how to make a highly rated mobile App from this project? ( 2 points)**

Family category apps, paid apps with higher size tend to have better ratings. But they also will have lower number of installs and reviews.