Predict Google Play Store App Ratings

Authors

Randall Byrd, Riyazh Dholakia, Sujal Patel & Biprojoyti Paul

Introduction

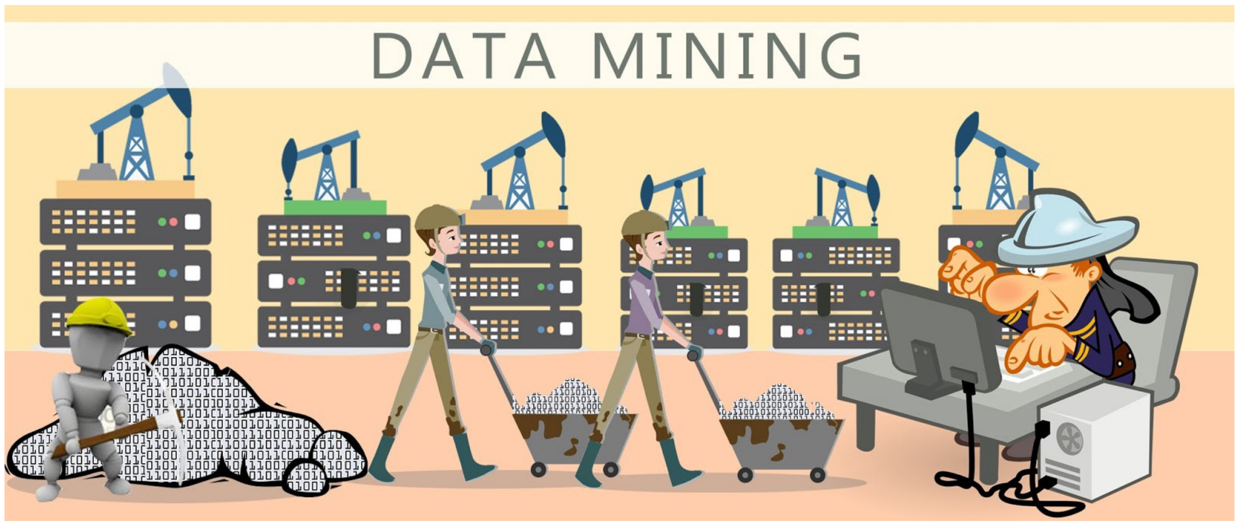
Welcome to our article! We are extremely excited to explain and share our thought process on how to **Predict Google Play Store App Ratings.** We still start off with basic Data Mining definitions and get into algorithms with our Google Play Store dataset.

Click here to access code on GitHub:
https://github.com/riyazhdholakia/PredictGooglePlayStoreAppRatings

Click here for this article on Medium:
https://medium.com/quickknowledge/predict-google-play-store-app-ratings-7a2634c65d87

Data Mining

Data mining is the process of looking through a data set and finding correlations, anomalies and or patterns that can be of usefulness. In other words, it is having a large dataset full of scattered information and trying to make sense of it by finding meaningfulness. For example, if a company like Macys wanted to predict the sales of a new item, they could look at similar items.



Python

Most of the data scientist use python because of the great built-in library functions and the decent community. Python now has 70,000 libraries. Python is simplest programming language to pick up compared to other language. That is the main reason data scientists use python more often, for machine learning and data mining data analyst want to use some language which is easy to use. That is one of the main reasons to use python. Specifically, for data scientist the most popular data built in open source library is called panda. As we have seen earlier in our previous assignment when we need to plot scatterplot, heat maps, graphs, 3-dimensional data python built-in library comes very helpful.

Software

We Use Anaconda Navigator to launch Jupyter Notebook. Then we choose Python 3 for coding.

Data

There are various ways of finding the data. Most ideal way to find data is to search it online. There is good amount of website that can provide large data, for example, Data.Gov, The Federal Reserve, Kaggle.

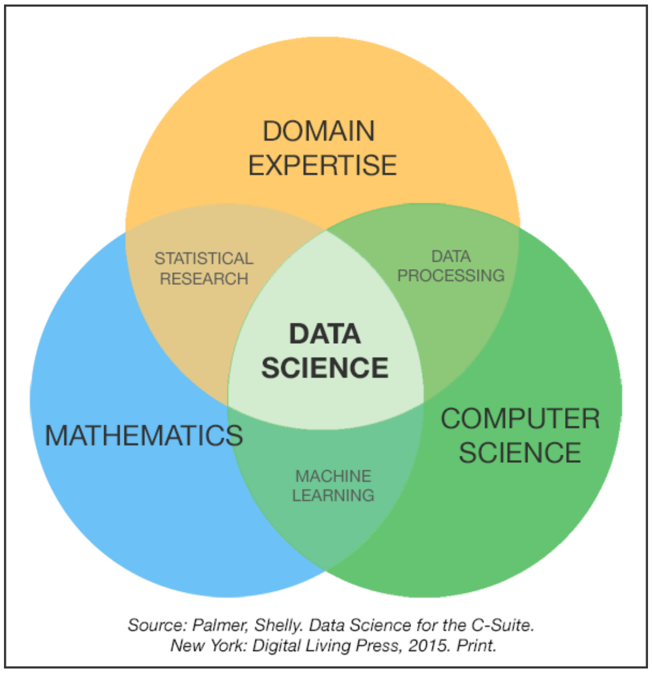
Our Dataset

Kaggle: https://www.kaggle.com/lava18/google-play-store-apps

Dataset was obtained, by scrapping off Google Play Store.

Domain Knowledge

It is hard to do projects in a domain you do not know or understand. You should understand the situation of the data and the importance of it. Suppose you want to build a model to predict how frequent oil changes are need Japanese cars. You should know the domain to understand the variables to gather, to use the data to predict the outcome. If you understand the domain you will understand false positives, outliers, and better measure errors. If you do not have a domain about car manufacturing, parts, vehicle models, etc. you will not be able to make a model. After you get the data and model, you should understand and analyze the data so you can communicate your findings. These are just a few important reasons why domain knowledge is important.



Body

Now you have introduction of Data Mining, let's get into it!

Hypothesis

Predict Google Play Store app ratings.

Independent Variables

Independent variables that do not have any dependence variables that can change the outcome of independent variables. Another name for independent variables is Predictor variables. In data, independent variables will also be known as regressors, controlled variables, manipulated variables, explanatory variable, exposure variable, and/or input variable. Most of the time the companies want to control independent variable to see the relationship of the dependent variables factors and the outcome.

Dependent Variables

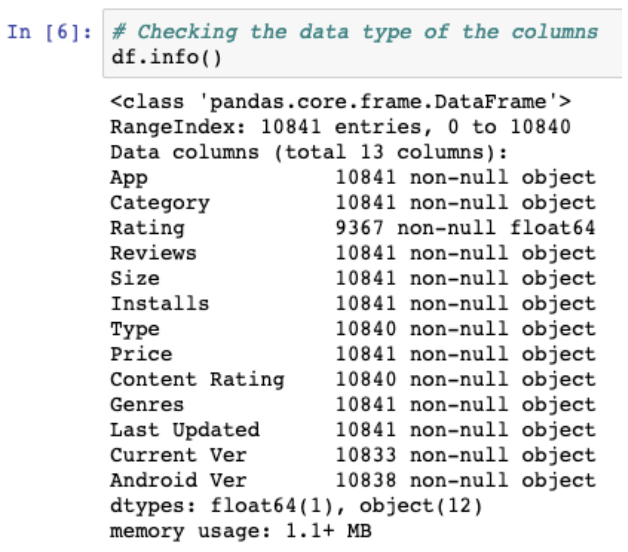
Dependent variables are the opposite. There outcome is dependent on another variable. Dependent variables can measure the effects of the independent variables. Dependent variables usually depend on the independent variables. Another name for dependent variables is Predicted variables. In data, dependent variables will also be known as response variable, regressand, measured variable, observed variable, responding variable, explained variable, outcome variable, experimental variable, and/or output variables.

Preprocessing

Preprocessing is important into transitioning raw data into a more desirable format. Undergoing the preprocessing process can help with completeness and compellability. For instance, you can see if certain values were recorded or not. Also, you can see how trustable the data is. It could also help with finding how consistent the values are. We need preprocessing because most real-world data are dirty. Data can be noisy i.e. the data can contain outliers or just errors in general. Data can also be incomplete i.e. there can be some missing values.

Data storage is getting easier and easier, most organizations are storing a bulk of data, for various purposes like auditing, information, prediction etc. Which turns out to be beneficial to the organization. But at the same time, bulk data said as big data increases the possibility of noisy data. Also, as this data would be coming from different sources, there is a higher chance of inconsistency. Even in smaller data set there are missing entries and inconsistent formats. All these anomalies hamper the cause of storing data, it can cause low quality mining, the prediction analysis might get drifted to the wrong side. While violets the cause of storing data, hence it is important to make the data as required which we say as preprocessing. This brings in data accuracy, completeness, consistency, timeliness, believability and interpretability.

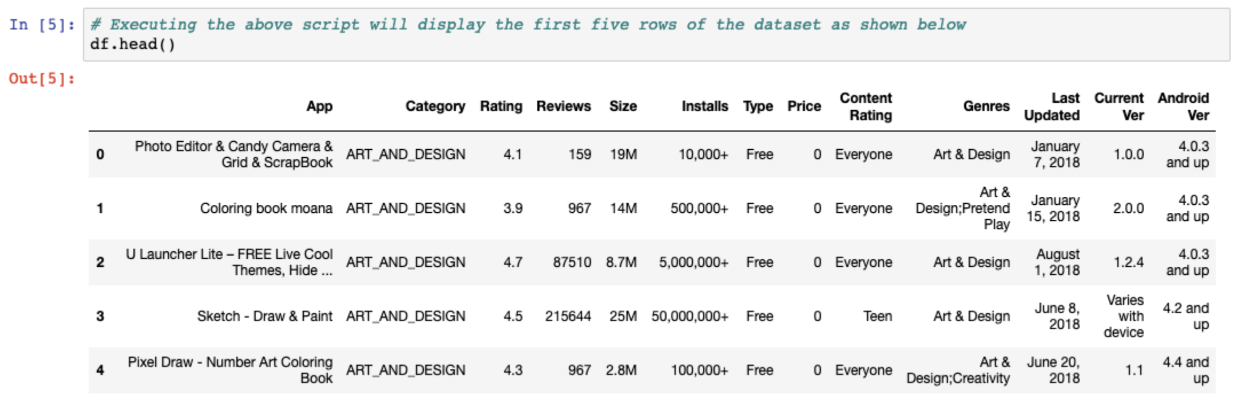
Data Types



```
In [6]: # Checking the data type of the columns
        df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 13 columns):
App               10841 non-null object
Category          10841 non-null object
Rating            9367 non-null float64
Reviews           10841 non-null object
Size              10841 non-null object
Installs          10841 non-null object
Type              10840 non-null object
Price             10841 non-null object
Content Rating    10840 non-null object
Genres            10841 non-null object
Last Updated      10841 non-null object
Current Ver       10833 non-null object
Android Ver       10838 non-null object
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

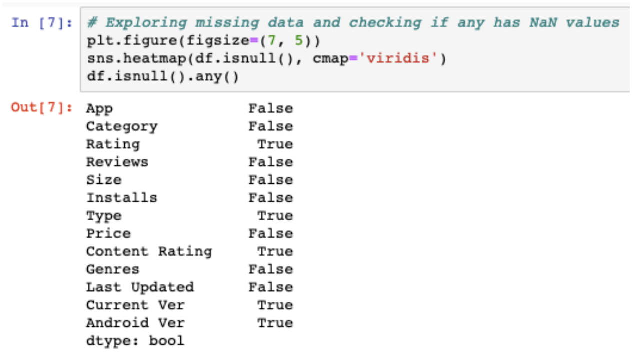
Type of Attributes

This is the First step of Data Data-preprocessing. We differentiate between different types of attributes and then preprocess the data. So here is a description of attribute types.
1. Qualitative (Nominal (N), Ordinal (O), Binary(B)).
2. Quantitative (Discrete, Continuous)

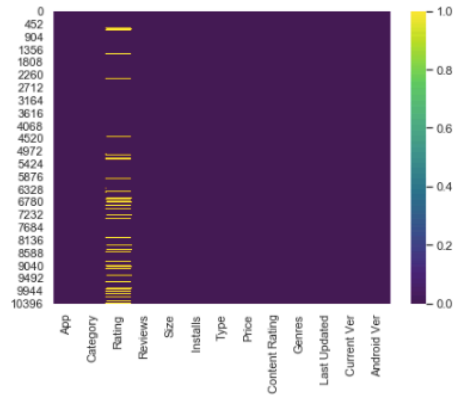


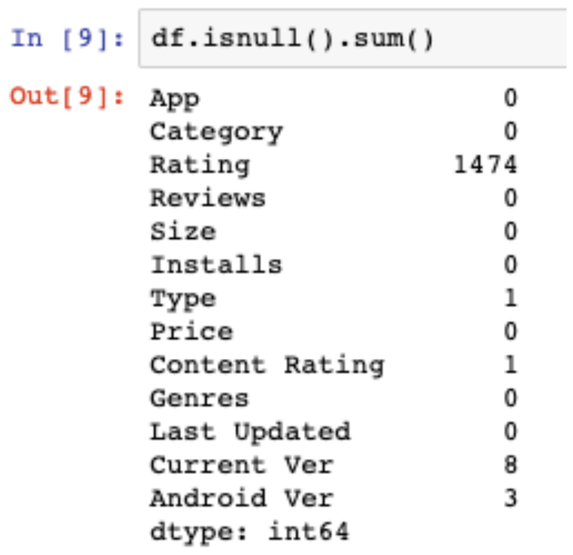
```
In [5]: # Executing the above script will display the first five rows of the dataset as shown below
        df.head()
Out[5]:
```

	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	January 7, 2018	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	January 15, 2018	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	August 1, 2018	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	June 8, 2018	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	June 20, 2018	1.1	4.4 and up



```
In [7]: # Exploring missing data and checking if any has NaN values
        plt.figure(figsize=(7, 5))
        sns.heatmap(df.isnull(), cmap='viridis')
        df.isnull().any()

Out[7]: App               False
        Category          False
        Rating             True
        Reviews           False
        Size              False
        Installs          False
        Type               True
        Price             False
        Content Rating     True
        Genres            False
        Last Updated      False
        Current Ver        True
        Android Ver        True
        dtype: bool
```





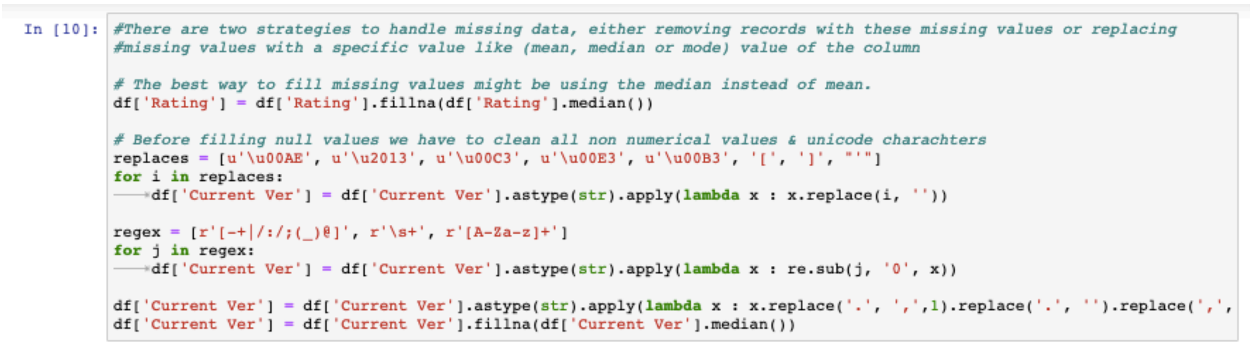
```
In [9]: df.isnull().sum()

Out[9]: 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
```

Looks like there are missing values in "Rating", "Type", "Content Rating" and " Android Ver". But most of these missing values in Rating column. First, we checked to see if there were any missing values/NaN values.

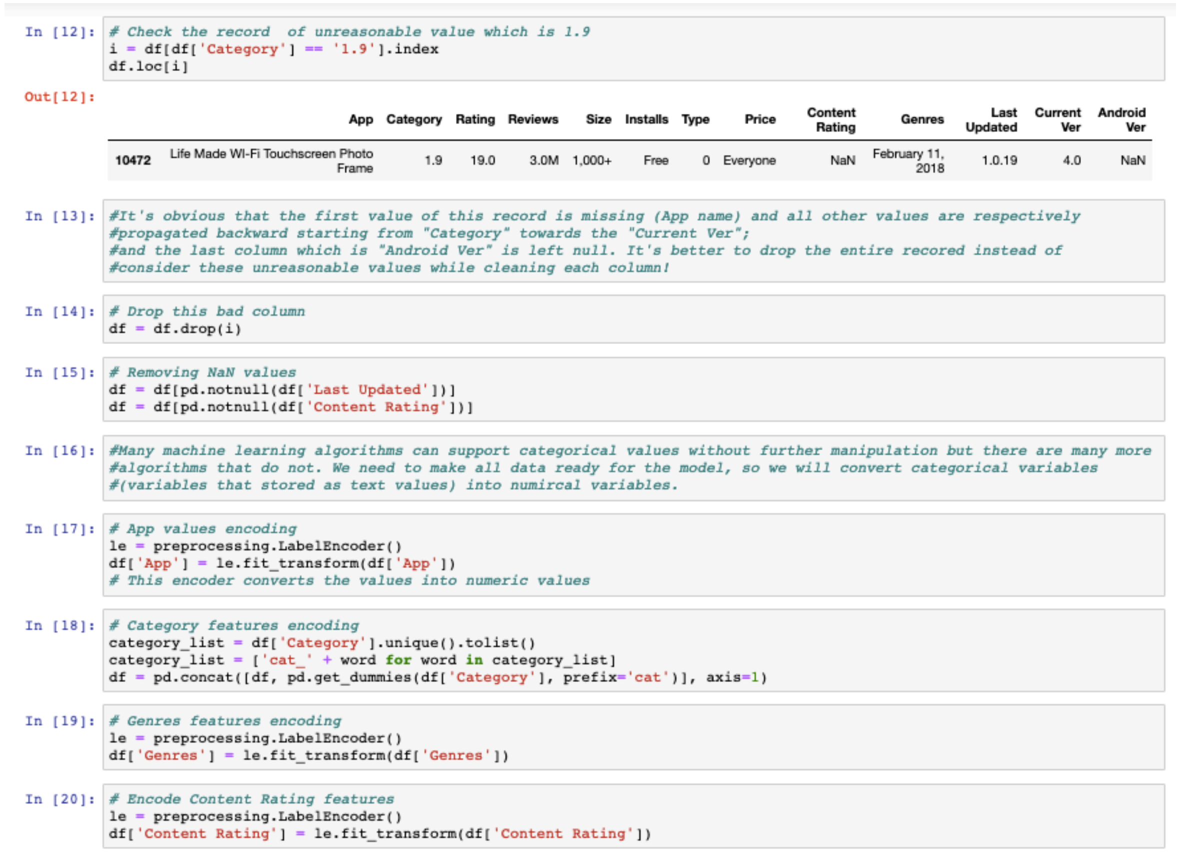
It's obvious that the first value of this record is missing (App name) and all other values are respectively propagated backward starting from "Category" towards the "Current Ver"; and the last column which is "Android Ver" is left null. It's better to drop the entire record instead of consider these unreasonable values while cleaning each column!

Many machine learning algorithms can support categorical values without further manipulation but there are many more algorithms that do not. We need to make all data ready for the model, so we will convert categorical variables (variables that stored as text values) into numerical variables.

There are two strategies to handle missing data, either removing records with these missing values or replacing missing values with a specific value like (mean, median or mode) value of the column. We used median to fill in the values. Using the median gave better results than the mean when replacing missing values.



```
In [10]: #There are two strategies to handle missing data, either removing records with these missing values or replacing
         #missing values with a specific value like (mean, median or mode) value of the column

         # The best way to fill missing values might be using the median instead of mean.
         df['Rating'] = df['Rating'].fillna(df['Rating'].median())

         # Before filling null values we have to clean all non numerical values & unicode charachters
         replaces = [u'\u00AE', u'\u2013', u'\u00C3', u'\u00E3', u'\u00B3', '[', ']', "'"]
         for i in replaces:
             df['Current Ver'] = df['Current Ver'].astype(str).apply(lambda x : x.replace(i, ''))

         regex = [r'[-+|/:/;(_)@]', r'\s+', r'[A-Za-z]+']
         for j in regex:
             df['Current Ver'] = df['Current Ver'].astype(str).apply(lambda x : re.sub(j, '0', x))

         df['Current Ver'] = df['Current Ver'].astype(str).apply(lambda x : x.replace('.', ',',1).replace('.', '').replace(',',
         df['Current Ver'] = df['Current Ver'].fillna(df['Current Ver'].median())
```



```
In [12]: # Check the record  of unreasonable value which is 1.9
         i = df[df['Category'] == '1.9'].index
         df.loc[i]

Out[12]:
                                   App  Category  Rating  Reviews  Size  Installs  Type     Price   Content     Genres       Last  Current  Android
                                                                                                     Rating                Updated      Ver      Ver
         10472  Life Made WI-Fi Touchscreen Photo     1.9    19.0     3.0M  1,000+  Free  0  Everyone      NaN  February 11,   1.0.19      4.0      NaN
                                         Frame                                                                       2018

In [13]: #It's obvious that the first value of this record is missing (App name) and all other values are respectively
         #propagated backward starting from "Category" towards the "Current Ver";
         #and the last column which is "Android Ver" is left null. It's better to drop the entire recored instead of
         #consider these unreasonable values while cleaning each column!

In [14]: # Drop this bad column
         df = df.drop(i)

In [15]: # Removing NaN values
         df = df[pd.notnull(df['Last Updated'])]
         df = df[pd.notnull(df['Content Rating'])]

In [16]: #Many machine learning algorithms can support categorical values without further manipulation but there are many more
         #algorithms that do not. We need to make all data ready for the model, so we will convert categorical variables
         #(variables that stored as text values) into numircal variables.

In [17]: # App values encoding
         le = preprocessing.LabelEncoder()
         df['App'] = le.fit_transform(df['App'])
         # This encoder converts the values into numeric values

In [18]: # Category features encoding
         category_list = df['Category'].unique().tolist()
         category_list = ['cat_' + word for word in category_list]
         df = pd.concat([df, pd.get_dummies(df['Category'], prefix='cat')], axis=1)

In [19]: # Genres features encoding
         le = preprocessing.LabelEncoder()
         df['Genres'] = le.fit_transform(df['Genres'])

In [20]: # Encode Content Rating features
         le = preprocessing.LabelEncoder()
         df['Content Rating'] = le.fit_transform(df['Content Rating'])
```

Normalization or Standardization

The terms normalization and standardization are sometimes used interchangeably, but they usually refer to different things. Normalization usually means to scale a variable to have a value between 0 and 1, while standardization transforms data to have a mean of zero and a standard deviation of 1. Normalization makes training less sensitive to the scale of features, so we can better solve for coefficients. Standardizing tends to make the training process well behaved because the numerical condition of the optimization problems is improved. We did not normalize or standardize this dataset, it was not necessary.

Dummy variables

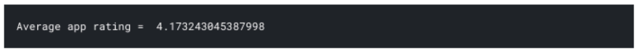
A dummy variable can only be in the quantitative value of 0 and 1. 1 represents the value exists while 0 represents it is missing value. It helps with categorical effect. We can implement this feature into our data. In this case, we did not need any dummy variables. Example for other datasets, suppose we had a feature that has male and female. We can add a dummy variable Xi that tells us Male 1, or Non-Male 0. In this case, it would help for prediction.

Descriptive Statics

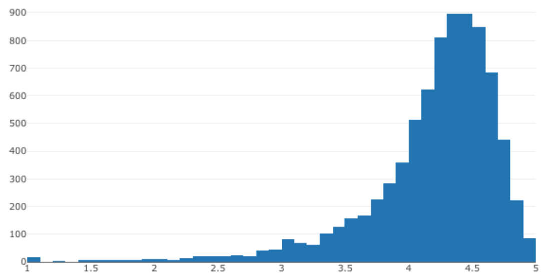
Simply, Descriptive static is representation of the specific large dataset sample. It gives us the coefficient that summarize the data. Descriptive statics can be measured by measurement. One is by central Tendency another one is by standard deviation. Central Tendency have broken into median, mean and mode.

Mean

Mean is the average. So, it is calculated by adding all the figures and later divide by the number of figures.



```
Average app rating =  4.173243045387998
```



Mode

Mode means in the dataset which value is occurring more often.

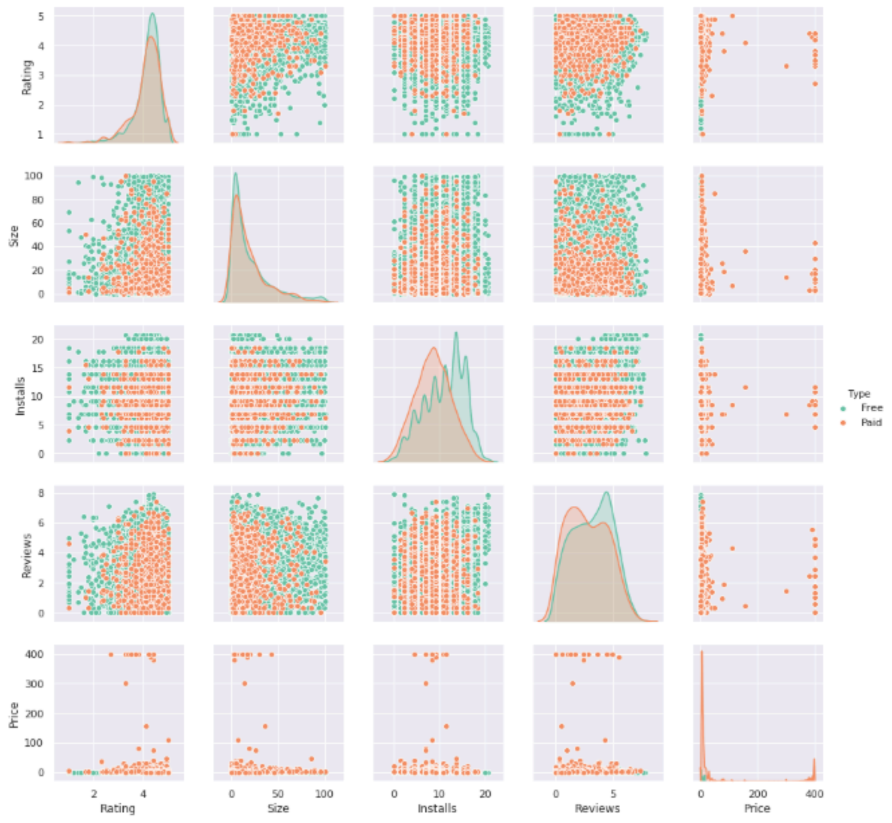
Median

Median is the figure which occurred in the middle of the dataset.

Standard Deviation

The standard deviation is square of variance. That is one of the ways to measure the data. To calculate the standard deviation is first find the mean. Then each number will be subtracted from mean. Then take the mean of those squared differences. After that take, the squared difference then we will be done.

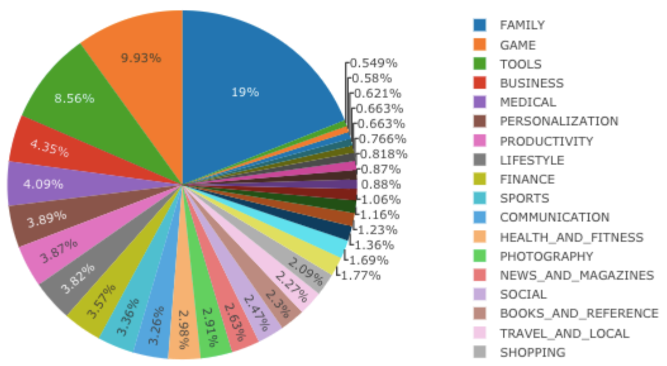
Below these plots can help us learn more about our data.

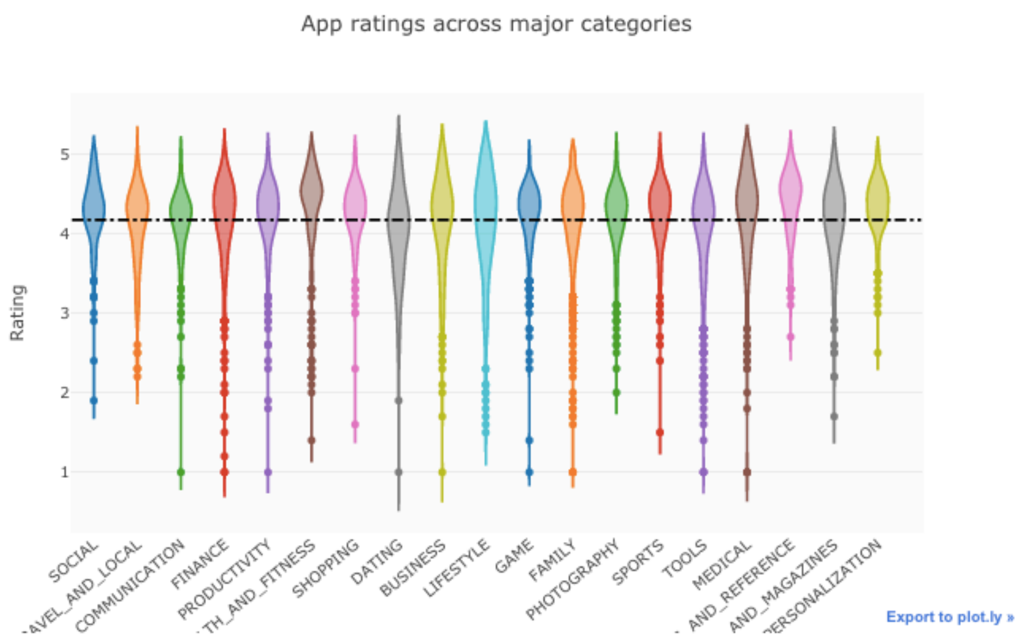


```
This is the basic exploratory analysis to look for any evident patterns or relationships bet
ween the features.
```

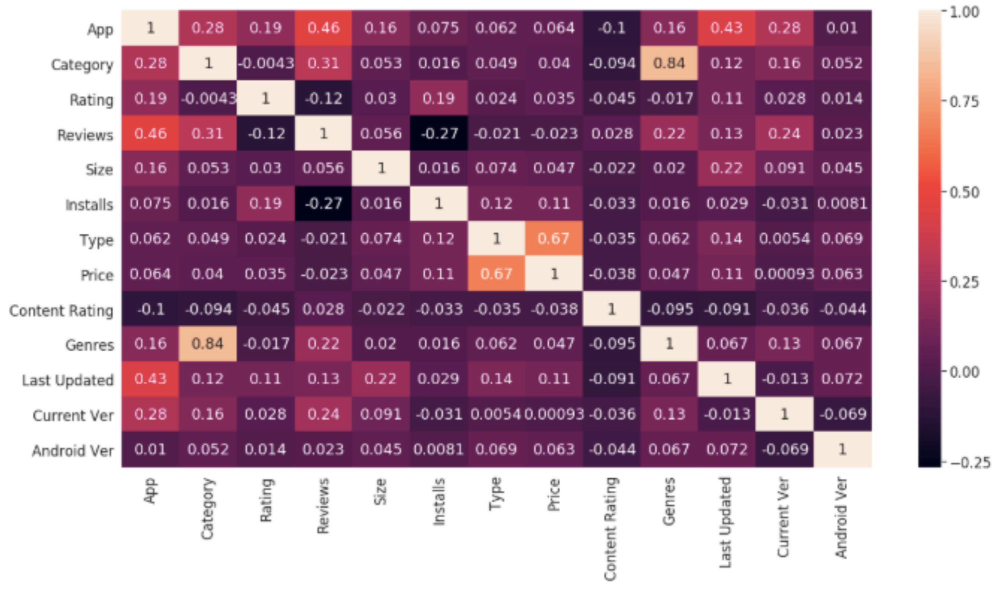
Android market breakdown

Which category has the highest share of (active) apps in the market?





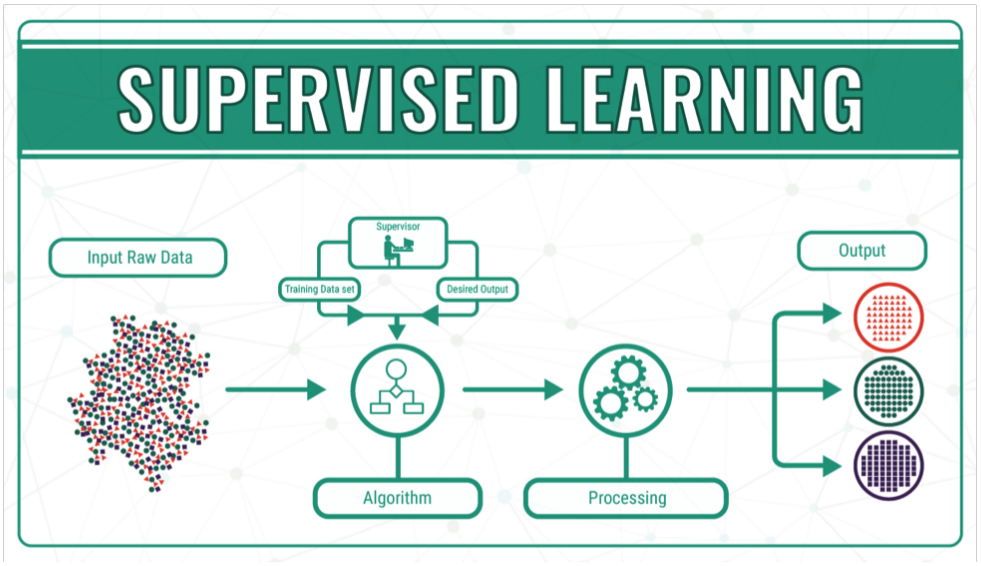
Correlation Heat Map



	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
App	1	0.28	0.19	0.46	0.16	0.075	0.062	0.064	-0.1	0.16	0.43	0.28	0.01
Category	0.28	1	-0.0043	0.31	0.053	0.016	0.049	0.04	-0.094	0.84	0.12	0.16	0.052
Rating	0.19	-0.0043	1	-0.12	0.03	0.19	0.024	0.035	-0.045	-0.017	0.11	0.028	0.014
Reviews	0.46	0.31	-0.12	1	0.056	-0.27	-0.021	-0.023	0.028	0.22	0.13	0.24	0.023
Size	0.16	0.053	0.03	0.056	1	0.016	0.074	0.047	-0.022	0.02	0.22	0.091	0.045
Installs	0.075	0.016	0.19	-0.27	0.016	1	0.12	0.11	-0.033	0.016	0.029	-0.031	0.0081
Type	0.062	0.049	0.024	-0.021	0.074	0.12	1	0.67	-0.035	0.062	0.14	0.0054	0.069
Price	0.064	0.04	0.035	-0.023	0.047	0.11	0.67	1	-0.038	0.047	0.11	0.00093	0.063
Content Rating	-0.1	-0.094	-0.045	0.028	-0.022	-0.033	-0.035	-0.038	1	-0.095	-0.091	-0.036	-0.044
Genres	0.16	0.84	-0.017	0.22	0.02	0.016	0.062	0.047	-0.095	1	0.067	0.13	0.067
Last Updated	0.43	0.12	0.11	0.13	0.22	0.029	0.14	0.11	-0.091	0.067	1	-0.013	0.072
Current Ver	0.28	0.16	0.028	0.24	0.091	-0.031	0.0054	0.00093	-0.036	0.13	-0.013	1	-0.069
Android Ver	0.01	0.052	0.014	0.023	0.045	0.0081	0.069	0.063	-0.044	0.067	0.072	-0.069	1

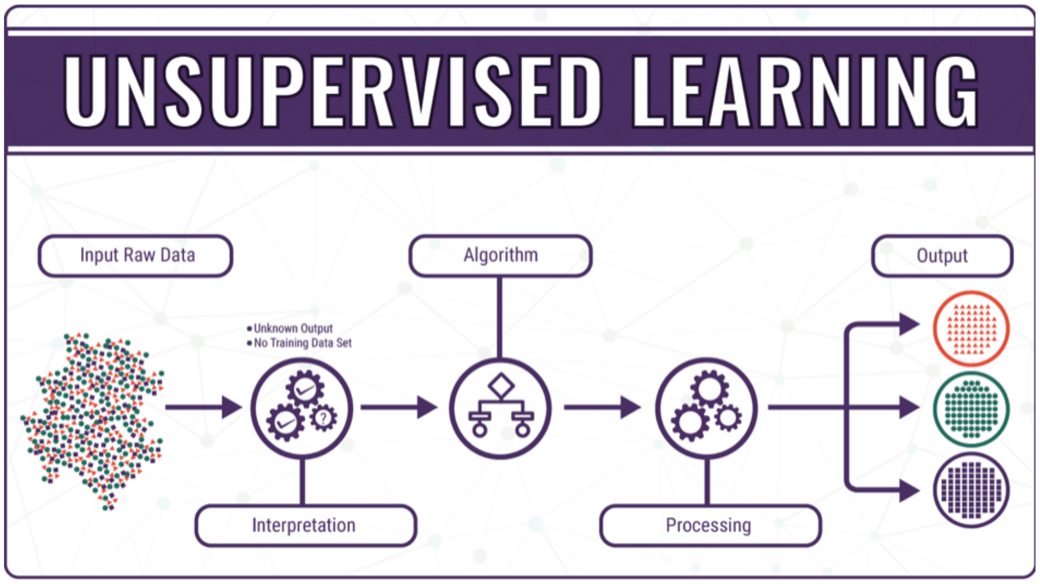
Supervised Learning

Supervised learning takes a known input set in a dataset and tries to learn the regression or classification model. In other words, with supervised learning humans teach machines how to learn from data through known inputs/outputs. An example of this can be a human teaching a machine to recognize a single card (in a deck of cards). First, he/she can show the machine how to recognize the symbol of the card i.e. diamond, club, spade, heart. After that the number or face can be recognized. Moving forward i.e. after the teaching process is done, the machine will then know how to read the next card and give an accurate answer for its face value.



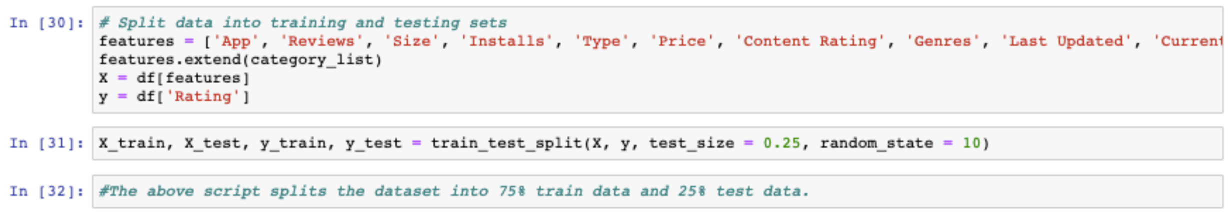
Unsupervised Learning

Unsupervised learning is a type of machine learning that does not require labelled data to train a model. In other words, the computer teaches itself and does not need any human supervision. Unsupervised Learning also is a way to do machine learning that helps with clustering analysis. Clustering analysis helps find hidden patterns in data using similarity. Basically, helps make references of input data without responses that are labeled. Common algorithms are hierarchical clustering, k-means, and mixture models (like Gaussian mixture models). As an example, think of a pond with various types of fish in it, and then imagine a device that can classify and group the different types of fish without having any supervised learning of how to go about the classification. The art of the computer using an algorithm without a human's help is unsupervised learning. One popular unsupervised learning algorithm that could separate the fish could be k means clustering. With k means clustering the goal of the algorithm is to assign different data points a specific group i.e. separate the fish by type. Where the data points are clustered using feature similarity. If no information is provided Unsupervised Learning would work well.



Training and Test Split

Separating training and test sets, most of the data is used for training sets and only some of the data is used for test sets. Separating data into training and testing sets is a significant piece of evaluating data mining models. By using similar data for training and testing, you can minimize the effects of data discrepancies and better understand the characteristics of the model. We split our data into 75 training and 25 test. Mostly, people use 80% train set and 20% test set.



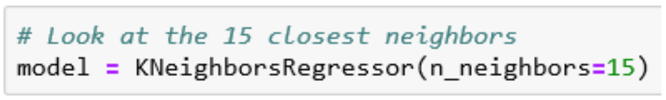
```
In [30]: # Split data into training and testing sets
         features = ['App', 'Reviews', 'Size', 'Installs', 'Type', 'Price', 'Content Rating', 'Genres', 'Last Updated', 'Curren
         features.extend(category_list)
         X = df[features]
         y = df['Rating']

In [31]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 10)

In [32]: #The above script splits the dataset into 75% train data and 25% test data.
```

Algorithms

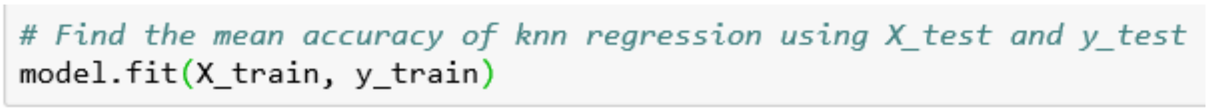
There are many algorithms we can use on our dataset. Some algorithms will work better for you over others. You will just have to try few different algorithms for your dataset, and see which one gives you the best results. Here are few popular algorithms we used that worked best for our dataset.

K-Nearest Neighbor's

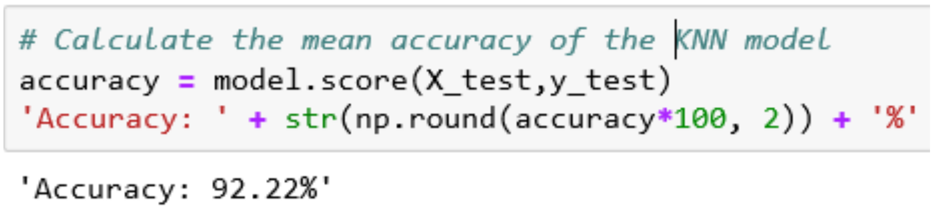
KNN is easiest supervised machine learning algorithm. It is the most basic machine learning algorithm you can find on scikit-learn. We can use KNN solve complicated problems. With the help of KNN we can do pattern recognition and data mining. KNN defines the similarity. From the given dataset KNN finds common groups between attributes. We split the data into training and test set. Then we can see how much similarity it becomes on the result.



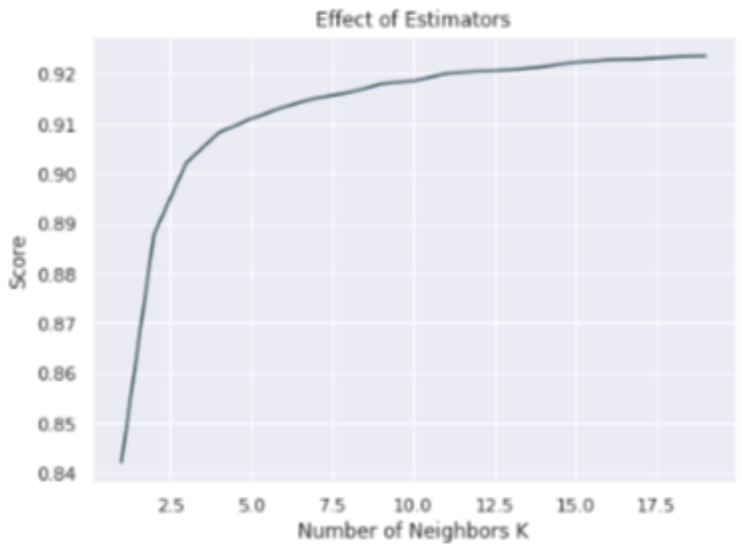
```
# Look at the 15 closest neighbors
model = KNeighborsRegressor(n_neighbors=15)
```



```
# Find the mean accuracy of knn regression using X_test and y_test
model.fit(X_train, y_train)
```



```
# Calculate the mean accuracy of the KNN model
accuracy = model.score(X_test,y_test)
'Accuracy: ' + str(np.round(accuracy*100, 2)) + '%'

'Accuracy: 92.22%'
```



Random Foresting

Random forest is basically creating a forest of decision trees and make it random. We will see a relationship between our trees and the result we are going to get. For example- If I am deciding what movie to watch this Sunday, and I want the suggestion from my friends. My friends will ask me random question like what genre first and then what specific type of movies from that genre. So, over here friend's suggestions are treated as decision tree. Finally, I am going to choose a movie who got the most votes. That is the result. These a random foresting algorithm example.

Some of the advantages and disadvantages of random forest classifiers are as follows:

Advantages

One advantage of random forest model is it never over-fits the model. It can handle missing values. We can use Random forest for both classification and regression. The random forest can help us identify the most important part from our training dataset.

Disadvantages

Random forests help for real-time prediction but that is slow in nature. It is a complex algorithm so sometimes it is hard to implement.

```
model = RandomForestRegressor(n_jobs=-1)
# Try different numbers of n_estimators - this will take a minute or so
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. estimator")
plt.ylabel("score")
plt.plot(estimators, scores)
results = list(zip(estimators,scores))
results
```

```
predictions = model.predict(X_test)
'Mean Absolute Error:', metrics.mean_absolute_error(y_test, predictions)
```

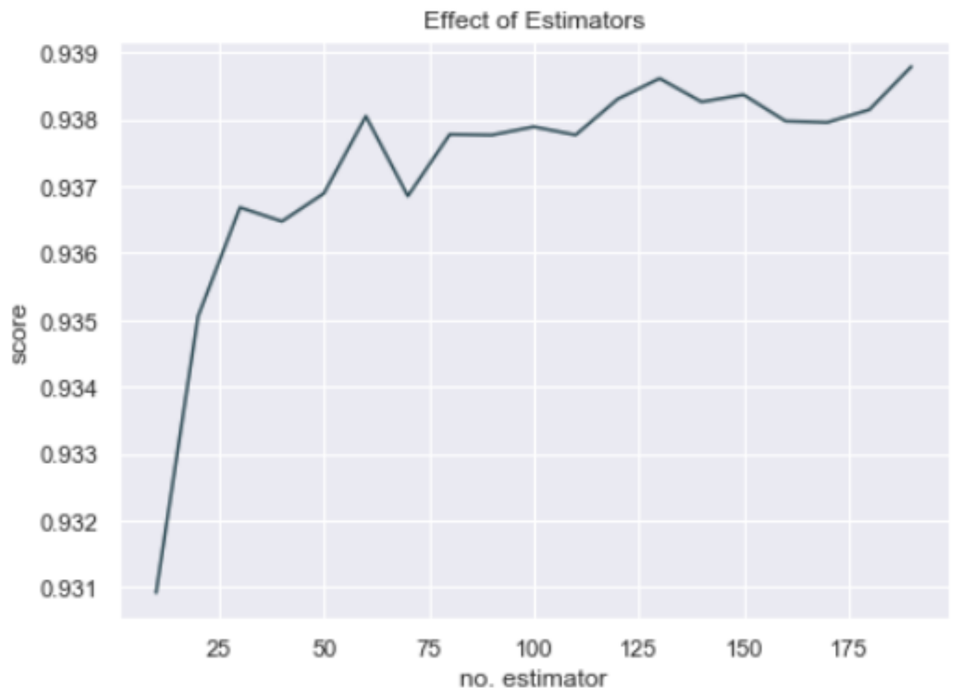
```
('Mean Absolute Error:', 0.24074655272868536)
```

```
'Mean Squared Error:', metrics.mean_squared_error(y_test, predictions)
```

```
('Mean Squared Error:', 0.15970111477956886)
```

```
'Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, predictions))
```

```
('Root Mean Squared Error:', 0.399626218834011)
```



Clustering

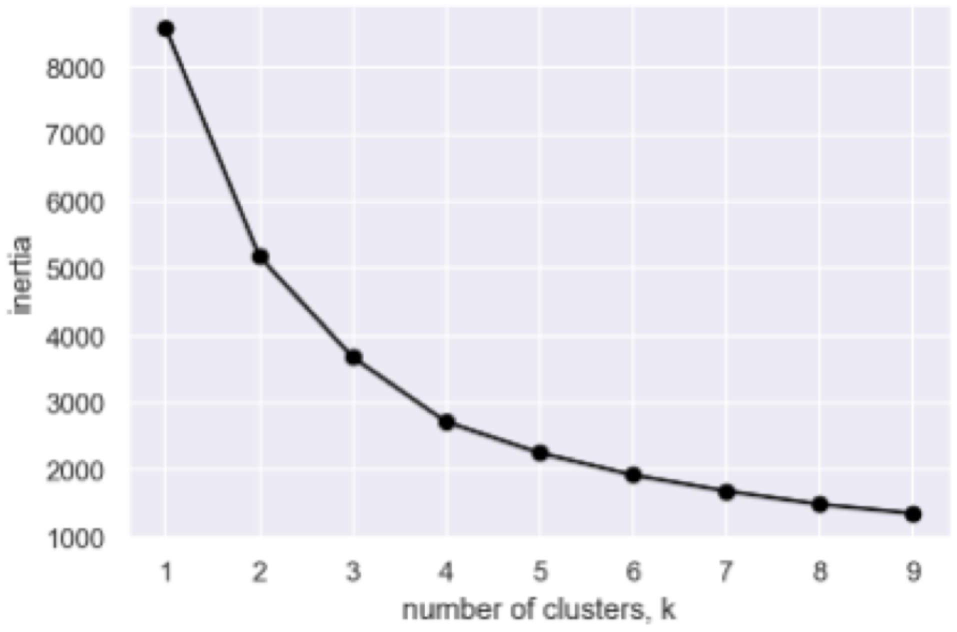
Clustering is basically a group of items or elements. Think about a pack of pencils, this is a cluster of pencils vs a single pencil. Clustering term is used in many ways. Clustering in data mining can help us see trends and compare data. This really helps on plots especially when each value is colored.

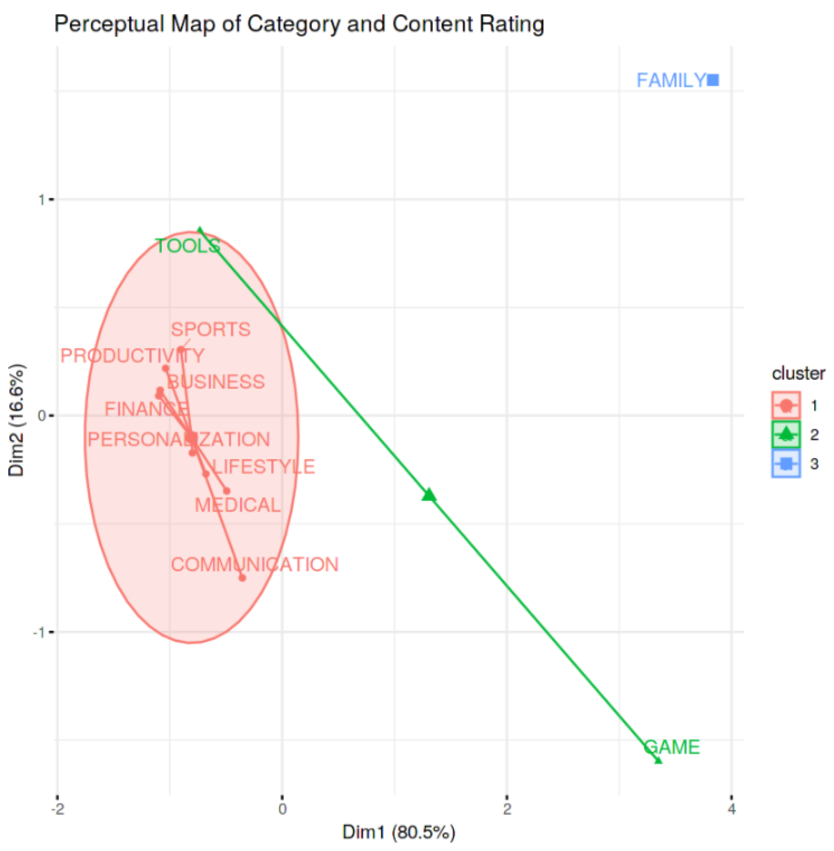
K-Means Clustering

K-Means Clustering Algorithm works by comparing or using similarity for each data point. This is commonly used with Unsupervised Learning. The K term means the number of groups clustered. Then we get the results of the centers of the clusters. Then we can plot this as well as our original clustered data. For example, K-Means you should be an expert in your data. This mean clustering numbers should be known before you run your algorithm. K-Means works when the data is in more of a flipped cone shape.

Our K is 4, look for the elbow point, in the plot below.

```
set.seed(123)
km.res <- kmeans(contingency_table, 3, 25)
fviz_cluster(km.res, data = contingency_table, palete=c("#2E9FDF","#00AFBB"),
             ellipse.type="euclid", star.plot=TRUE, repel = TRUE, ggtheme = theme_minimal(),
             main="Perceptual Map of Category and Content Rating")
```





Hierarchical Clustering

Similarly, hierarchical clustering also clusters your data. This is also commonly used with Unsupervised Learning. This will also return centers of the clustered data. The process in which the results will vary. It will merge small clusters together to form bigger clusters until there is only one cluster left. You then decide in dendrogram which cluster values you want to pull. This will all depend on the data that you have. Hierarchical Clustering works differently because he will do the clustering for you. It will combine the clusters till there is only one left. You will get to see it on a plot which you decide the number of clusters. You usually want to follow a rule of thumb on this. This good especially if you do not have much knowledge of your data set. Also, Hierarchical Clustering works better with smaller data sets. We will not be using this, since K-Means Clustering worked better for us.

K-Fold Cross

K-fold cross it is a technique that is used to divide the data set into K number of folds for testing. Where each fold is used as a testing set at some point. It is a technique that is used to include limited data. K-fold is also less biased then other methods. Another word, it a is a strategy used to evaluate the aptitude of the model on new information. There are regular strategies that you can use to choose the estimation of k for your dataset. There are normally utilized minor departure from cross-validation, for example, stratified and rehashed that are accessible in scikit-learn. Cross-validation is a measurable technique used to gauge the expertise of AI models. It is regularly utilized in applied AI to look at and select a model for a given prescient demonstrating issue since it is straightforward, simple to actualize, and brings about aptitude evaluates that for the most part have a lower inclination than different strategies.

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 classification and 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 about 91% accuracy if an app will have more than 100,000 installs and be a hit on the Google Play Store.

Fun Facts

Auto and Vehicles and Entertainment do not have highly rated apps in these categories, this would be a great place to innovate a new app.

For advertisement revenue, focus on the top 40 most install apps to maximum views.

The Size, Type, Price, Content Rating, and Genre features should all be used to most accurately determine if an app will gain maximum installs.

In conclusion, we hope you enjoyed learning about Data Mining process on a dataset, and we enjoyed rattling your brain. We also had an amazing time writing this article and will continue sharing about Data Mining. If you have any questions feel free to reach out to us via email, Medium, LinkedIn or GitHub.

Resources

Medium:

- https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f
- https://towardsdatascience.com/hypothesis-testing-in-machine-learning-using-python-a0dc89e169ce

Kaggle:

- https://www.kaggle.com/lava18/google-play-store-apps

YouTube:

- https://www.youtube.com/watch?v=6kZ-OPLNcgE
- https://www.youtube.com/watch?v=s-9Qqpv2hTY

GitHub our code:

- https://github.com/riyazhdholakia/PredictGooglePlayStoreAppRatings

Medium our article:

- https://medium.com/quickknowledge/predict-google-play-store-app-ratings-7a2634c65d87