Data Cleaning and Transformation

Introduction

Data Cleaning is defined as correcting or removing inaccurate values from the dataset. It is one of the crucial steps while performing Data Mining. A Data Scientist spends roughly about 60% of the time cleaning the dataset. Still, the process of Data Cleaning is not perfect. Even after cleaning the data, a dataset can never be assumed to be completely free of inaccurate values. The main aim of Data Cleaning is to normalize all the attributes so that a better and accurate model can be created. Some of the most used Data Cleaning methods are:

- You can completely ignore the tuple if it is missing some values. This method is not effective at all.
- You can input the missing data manually. This method only works if the dataset is small.
- You can calculate and use attribute mean to fill in the missing values in the dataset.
- You can delete the tuple from the dataset.

The best practice is to fill in the missing attribute values either manually or by taking the mean of the entire attribute column. Hence, we have used this data cleaning method in our assignment. We have two datasets. The game_detailed_info.csv and the bgg-15m-reviews.csv. Hence, we will perform all the methods on both of them one by one.

Reviews Dataset

The reviews dataset contains attributes like user, game ID, rating, comment, game name. Most of the attributes in this dataset contains text data. As Natural language Processing is out of scope for this project, we will be focusing on only the rating attribute from the first dataset. Here is a snapshot of the dataset for reference.

	user	rating	comment	ID	name
0	Torsten	10.0	NaN	30549	Pandemic
1	mitnachtKAUBO-I	10.0	Hands down my favorite new game of BGG CON 200...	30549	Pandemic
2	avlawn	10.0	I tend to either love or easily tire of co-op ...	30549	Pandemic
3	Mike Mayer	10.0	NaN	30549	Pandemic

As we are focused on the ratings only, we tried finding the null values present in ratings column.

```
#Finding column frequency to delete columns having >60% null values
#Column Frequency
df.isnull().sum(axis = 0)
```

```
user              66
rating             0
comment     12832316
ID                 0
name               0
dtype: int64
```

We used isnull() to find the null values and sum() to add up all the null values. The output was that there were no null values in the rating attribute. Then, we tried to converted the 0 values into null values but then also we did not find any null values for the rating attribute.

```
#Replacing 0 with Null values in the rating colunm
df['rating'] = df['rating'].map(lambda x:x if x !=0 else None)
```

```
df.isnull().sum(axis = 0)
```

```
user              66
rating             0
comment     12832316
ID                 0
name               0
dtype: int64
```

Next, to calculate the outliers in the dataset, we used IQR method. The IQR came out to be 2. So using the formula, we tried to find the max and min values. All values beyond these are considered as outliers. We got the min as 3 and max was beyond scope of the rating dataset hence it got discarded.

```
#Finding Outliers in the Rating using IQR
Q1 = df['rating'].quantile(0.25)
Q3 = df['rating'].quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
2.0
```

```
#Q1 = 6, Q3 = 8 and IQR = 2
#((df < (Q1-1.5*IQR)) | (df > (Q3+1.5*IQR))) will give a value which is less than 3
#As per the Outlier guidelines, all the ratings below 3 can be dropped.
#but we are keeping them as it is as they are important for our model.
```

We can remove the ratings below 3 as per the guidelines. But as we are trying to create a recommendation system, we will need ratings of 3 and below as well so that we can classify a bad game successfully. Hence, we have decided to keep all the values even after finding the outliers.

For doing the normalization of the rating attribute, we used the z-score. To simply define, a z-score tells us the distance between mean and the datapoint value. But more technically it's a measure of how many standard deviations below or above the population mean a raw score is.
We normalized the dataset using the pandas inbuilt function scaler.fit_transform. The rating attribute column was highly unnormalized. We compared the describe functions and histogram plots before and after we did normalization. The results proved that the dataset was not normalized. Please find the screenshots of the histograms and the describe function below. This concludes the data cleaning and data transformation part for the first dataset. Next, we will move to the game_detailed_info dataset and explore it further.

```
# Describe before Data Normalization
df['rating'].describe()
```

```
count      1.582327e+07
mean       7.054843e+00
std        1.599649e+00
min        1.401300e-45
25%        6.000000e+00
50%        7.000000e+00
75%        8.000000e+00
max        1.000000e+01
Name: rating, dtype: float64
```

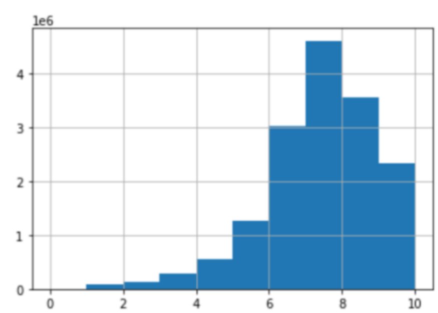
```
#Describe after Data Normalization
z_score_rating_df.describe()
```

	rating
count	1.582327e+07
mean	-8.320200e-12
std	1.000000e+00
min	-4.410246e+00
25%	-6.594221e-01
50%	-3.428471e-02
75%	5.908526e-01
max	1.841127e+00

Snapshot of the describe function before and after we applied Data Normalization.

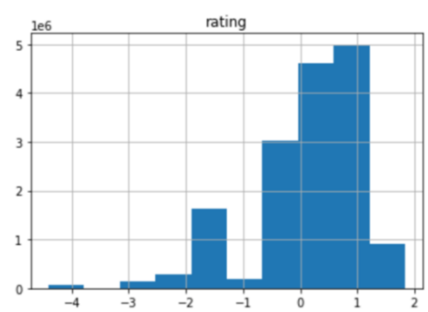
```
df['rating'].hist()
```

```
<AxesSubplot:>
```




```
z_score_rating_df.hist()
```

```
array([[<AxesSubplot:title={'center':'rating'}>]], dtype=object)
```



Snapshot of the histogram plot showing rating values before and after we applied Data Normalization.

Game Info Dataset

Game info dataset contains a lot of numeric attributes such as yearpublished, minplayers, maxplayers, playingtime, minplaytime, maxplaytime, etc. It also contains categorial data attributes such as descriptions, publisher, game genre, etc. For filling in the null values and for the scope of this particular assignment, we have focused more on the numeric attributes. The text data can be processed using NLP but that is out of scope right now. To find the null values in the dataset, we again used the isnull() and sum() built-in functions for finding the count of the null values in each attribute. The screenshot can be seen below with the initial null values:

```
#Sum of null values
df.isnull().sum()
```

```
id                      0
primary                 0
description             1
yearpublished           0
minplayers              0
maxplayers              0
playingtime             0
minplaytime             0
maxplaytime             0
minage                  0
boardgamecategory     221
boardgamemechanic    1549
boardgamefamily      4487
boardgamedesigner     430
boardgameartist      5397
boardgamepublisher      0
usersrated              0
average                 0
bayesaverage            0
Board Game Rank         0
dtype: int64
```

There can be cases where the value can be inputted 0 instead of null in the dataset. To verify, we used the lambda function to map all the numeric attributes and if it finds a 0 value then replace it by null value. We used the mapping and updated all the numeric data attributes to display null values when we use isnull() and sum() function again.

```
#Replacing 0 values by None Values
df['yearpublished'] = df['yearpublished'].map(lambda x:x if x !=0 else None)
df['minplayers'] = df['minplayers'].map(lambda x:x if x !=0 else None)
df['maxplayers'] = df['maxplayers'].map(lambda x:x if x !=0 else None)
df['playingtime'] = df['playingtime'].map(lambda x:x if x !=0 else None)
df['minplaytime'] = df['minplaytime'].map(lambda x:x if x !=0 else None)
df['maxplaytime'] = df['maxplaytime'].map(lambda x:x if x !=0 else None)
df['minage'] = df['minage'].map(lambda x:x if x !=0 else None)
```

After using the lambda function, we again counted the null values. We can see now a lot of null values as compared to the previous case. Now, we will work on calculating and inputting values in all these null places.

```
#Actual number of Null Values in the Dataset
df.isnull().sum()
```

```
id                      0
primary                 0
description             1
yearpublished         170
minplayers             51
maxplayers            167
playingtime           625
minplaytime           531
maxplaytime           625
minage               1173
boardgamecategory     221
boardgamemechanic    1549
boardgamefamily      4487
boardgamedesigner     430
boardgameartist      5397
boardgamepublisher      0
usersrated              0
average                 0
bayesaverage            0
Board Game Rank         0
dtype: int64
```

In order to input some meaningful data in place of the null values, we decided to use the mean. So, we calculated the individual mean values for all the attributes and inputted them wherever there was a null value. We used fillna() to automatically fill the null values with mean values from the attributes and saved the dataset.

```
#Imputing the mean values in place of null values
df['yearpublished'].fillna(df['yearpublished'].mean(), inplace=True)
df['minplayers'].fillna(df['minplayers'].mean(), inplace=True)
df['maxplayers'].fillna(df['maxplayers'].mean(), inplace=True)
df['playingtime'].fillna(df['playingtime'].mean(), inplace=True)
df['minplaytime'].fillna(df['minplaytime'].mean(), inplace=True)
df['maxplaytime'].fillna(df['maxplaytime'].mean(), inplace=True)
df['minage'].fillna(df['minage'].mean(), inplace=True)
```

Now, if we count for null values in the dataset using isnull() and sum() function, the number of null values should be 0 again.

```
df.isnull().sum()
```

```
id                     0
primary                0
description            1
yearpublished          0
minplayers             0
maxplayers             0
playingtime            0
minplaytime            0
maxplaytime            0
minage                 0
boardgamecategory    221
boardgamemechanic   1549
boardgamefamily     4487
boardgamedesigner    430
boardgameartist     5397
boardgamepublisher     0
usersrated             0
average                0
bayesaverage           0
Board Game Rank        0
dtype: int64
```

Next, for calculating the outliers, we used IQR again.

```
#Finding Outliers in the Rating using IQR
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
id                172062.250000
yearpublished         15.000000
minplayers             0.000000
maxplayers             2.000000
playingtime           60.000000
minplaytime           40.000000
maxplaytime           60.000000
minage                 4.000000
usersrated           325.750000
average                1.203658
bayesaverage           0.177217
Board Game Rank     9677.500000
dtype: float64
```

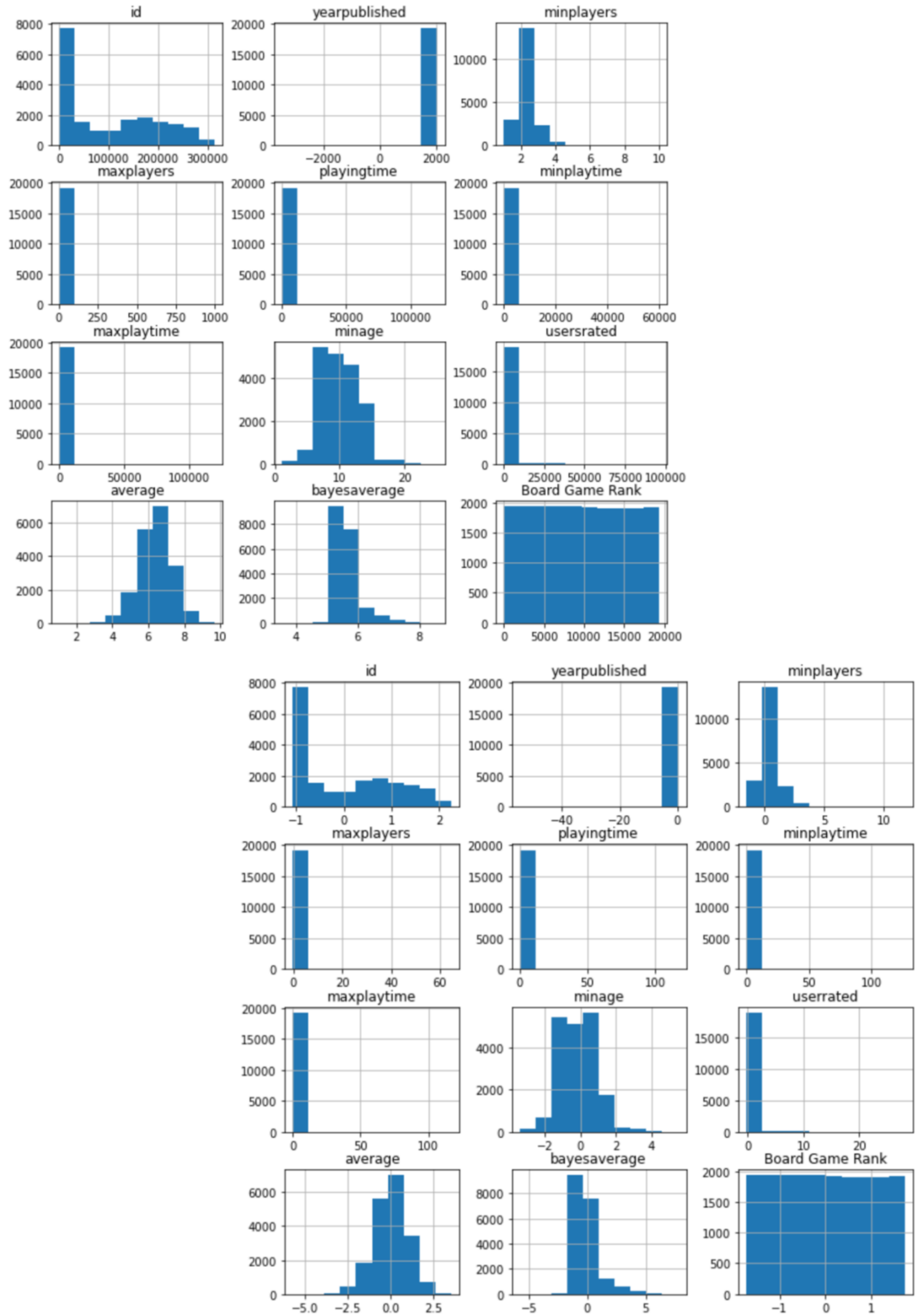
```python
#Calculating IQR
print((df < (Q1-1.5*IQR)) | (df > (Q3+1.5*IQR)))
```

```
          id  yearpublished  minplayers  maxplayers  playingtime  minplaytime  \
0      False          False       False       False        False        False
1      False          False       False       False        False        False
2      False          False        True       False        False        False
3      False          False       False       False        False        False
4      False          False       False       False        False        False
...      ...            ...         ...         ...          ...          ...
19225  False          False       False       False        False        False
19226  False          False       False       False        False        False
19227  False          False        True       False        False        False
19228  False          False       False       False        False        False
19229  False          False       False       False        False        False

       maxplaytime  minage  usersrated  average  bayesaverage  Board Game Rank
0            False   False        True    False          True            False
1            False   False        True    False          True            False
2            False   False        True    False          True            False
3            False   False        True    False          True            False
4            False   False        True    False          True            False
...            ...     ...         ...      ...           ...              ...
19225        False   False       False    False         False            False
19226        False   False       False    False         False            False
19227        False   False       False    False         False            False
19228        False   False       False    False         False            False
19229        False   False       False     True         False            False

[19230 rows x 12 columns]
```

For the binary values created above, we get the values whether they are outliers or not. We can remove all the outlier values, but we have decided to keep them for training the model. We don't want to lose any data because when we are getting recommendations, we will need good as well as bad game information.

For normalizing the dataset values, we used the z-score. To simply define, a z-score tells us the distance between mean and the datapoint value. But more technically it's a measure of how many standard deviations below or above the population mean a raw score is. We normalized the dataset using the pandas inbuilt function scaler.fit_transform. The rating attribute column was highly unnormalized. We compared the describe functions and histogram plots before and after we did normalization. The results proved that the dataset was not normalized. Please find the screenshots of the histograms function below.

After both the datasets where processed with the data cleaning techniques, we went ahead and saved the datasets into new csv files. We will be using these csv files as inputs for the next assignments and ultimately working on the model based on these csv files.



Snapshot of the histogram before and after normalizing the dataset.