Introduction

Use linear_model to predict the score of app in Google Play Store

Data cleaning

Our goal is to use the other given data except the score to predict the score of application.Although in the given data,score Y is not a continuous value,we set the score Y a continuous value to do the machine learning as a matter of convenience.

First of all,we pre-process the data set to remove the duplicate data,empty data and the NaN value.

```
d = pd.read_csv('googleplaystore.csv')

d = d.astype(str)
d.drop_duplicates(subset='App', inplace=True)
d = d[d['Android Ver'] != np.nan]
d = d[d['Android Ver'] != 'NaN']
d = d[d['Installs'] != 'Free']
d = d[d['Installs'] != 'Paid']
d = d[d['Size'] != 'Varies with device']
d = d[d['Size'] != '']
d = d[d['Rating'] != 'NaN']
d = d[d['Rating'] != 'nan']
```

Then we transform the Rating,Installs,Price,Reviews to the numerical value,each maintains the same unit.

```
d['Installs'] = d['Installs'].apply(lambda x: x.replace('+', '')
d['Installs'] = d['Installs'].apply(lambda x: x.replace(',', '')
d['Installs'] = d['Installs'].apply(lambda x: int(x))
d['Size'] = d['Size'].apply(lambda x: str(x).replace('M', '') if
d['Size'] = d['Size'].apply(lambda x: str(x).replace(',', '') if
d['Size'] = d['Size'].apply(lambda x: float(str(x).replace('k',
d['Size'] = d['Size'].apply(lambda x: float(x))
d['Installs'] = d['Installs'].apply(lambda x: float(x))
d['Price'] = d['Price'].apply(lambda x: str(x).replace('$', '')
d['Price'] = d['Price'].apply(lambda x: float(x))
d['Reviews'] = d['Reviews'].apply(lambda x: int(x))
```

For the prediction of machine learning,we transform the character value to the numerical value and classify the same to one class,then we choose the enumerated index as the input of machine learning.

```
def get_index_value(name, x):
    a = d.groupby([name]).size().reset_index()
    for index, i in enumerate(a.values):
        if i[0] == str(x):
            return index
    return -1
d['Category Val'] = d['Category'].apply(lambda x: get_index_valu
d['Type Val'] = d['Type'].apply(lambda x: get_index_value('Type'
d['Content Rating Val'] = d['Content Rating'].apply(lambda x: ge
d['Android Ver Val'] = d['Android Ver'].apply(lambda x: get_inde
```

Check Data

Data description,it will show the maximum and the minimum value,average value and so on,we can easily judge whether there are abnormal value.

```
print(examDf.describe())
```

```
         Unnamed: 0       Rating  Category Val     Rating.1
count   7027.000000  7027.000000   7027.000000  7027.000000  7.0
mean    5637.058631     4.160623     16.644372     4.160623  1.4
std     3079.050077     0.559145      8.205449     0.559145  1.0
min        0.000000     1.000000      0.000000     1.000000  1.0
25%     3086.500000     4.000000     11.000000     4.000000  8.4
50%     5715.000000     4.300000     14.000000     4.300000  1.5
75%     8288.500000     4.500000     24.000000     4.500000  2.6
max    10840.000000     5.000000     32.000000     5.000000  4.4

              Size      Installs     Type Val        Price  \
count  7027.000000  7.027000e+03  7027.000000  7027.000000
mean     21.754427  4.468208e+06     0.076989     1.173572
std      22.726503  2.713777e+07     0.266593    18.197602
min       0.008500  1.000000e+00     0.000000     0.000000
25%       4.900000  1.000000e+04     0.000000     0.000000
50%      13.000000  1.000000e+05     0.000000     0.000000
75%      31.000000  1.000000e+06     0.000000     0.000000
max     100.000000  1.000000e+09     1.000000   400.000000

  Content Rating Val  Android Ver Val
```

```
count         7027.000000      7027.000000
mean             1.456240        14.235378
std              1.001088         4.997960
min              0.000000         0.000000
25%              1.000000        12.000000
50%              1.000000        14.000000
75%              1.000000        16.000000
max              5.000000        31.000000
```

Check the loss value,if the output is 0,it means this column doesn't have loss value.

```
print(examDf[examDf.isnull() == True].count())
```

```
Unnamed: 0            0
Rating                0
Category Val          0
Rating.1              0
Reviews               0
Size                  0
Installs              0
Type Val              0
Price                 0
Content Rating Val    0
Android Ver Val       0
dtype: int64
```

In the normal situation,0-0.3 means weak correlation;0.3-0.6 means normal correlation;0.6-1 means strong correlation.

```
print(examDf.corr())
```

Print the correlation coefficient,judge whether it deserves to do linear-regression model.

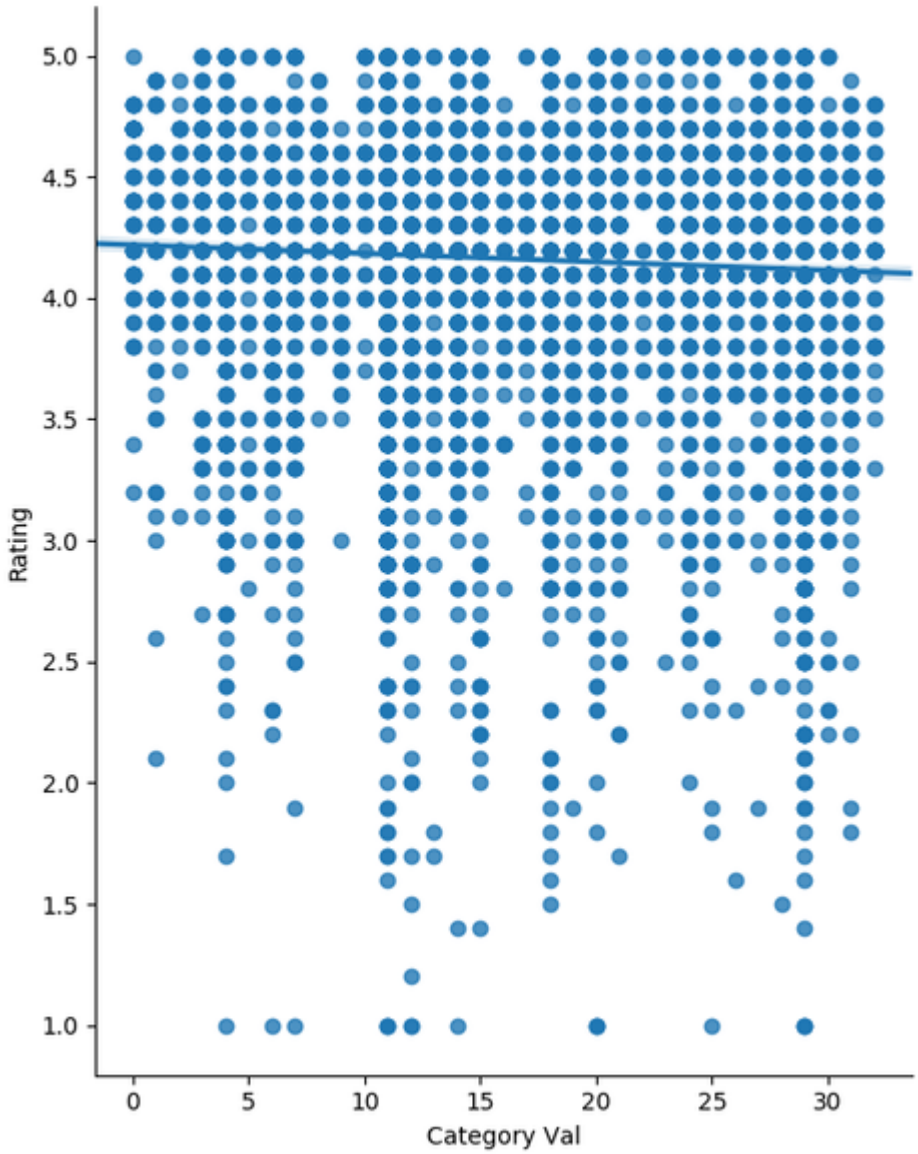
```
                  Unnamed: 0    Rating  Category Val  Rating.1
Unnamed: 0          1.000000 -0.112847      0.152614 -0.112847
Rating             -0.112847  1.000000     -0.051239  1.000000
Category Val        0.152614 -0.051239      1.000000 -0.051239
Rating.1           -0.112847  1.000000     -0.051239  1.000000
Reviews            -0.095687  0.067589     -0.000140  0.067589
Size               -0.077006  0.063067     -0.150375  0.063067
```

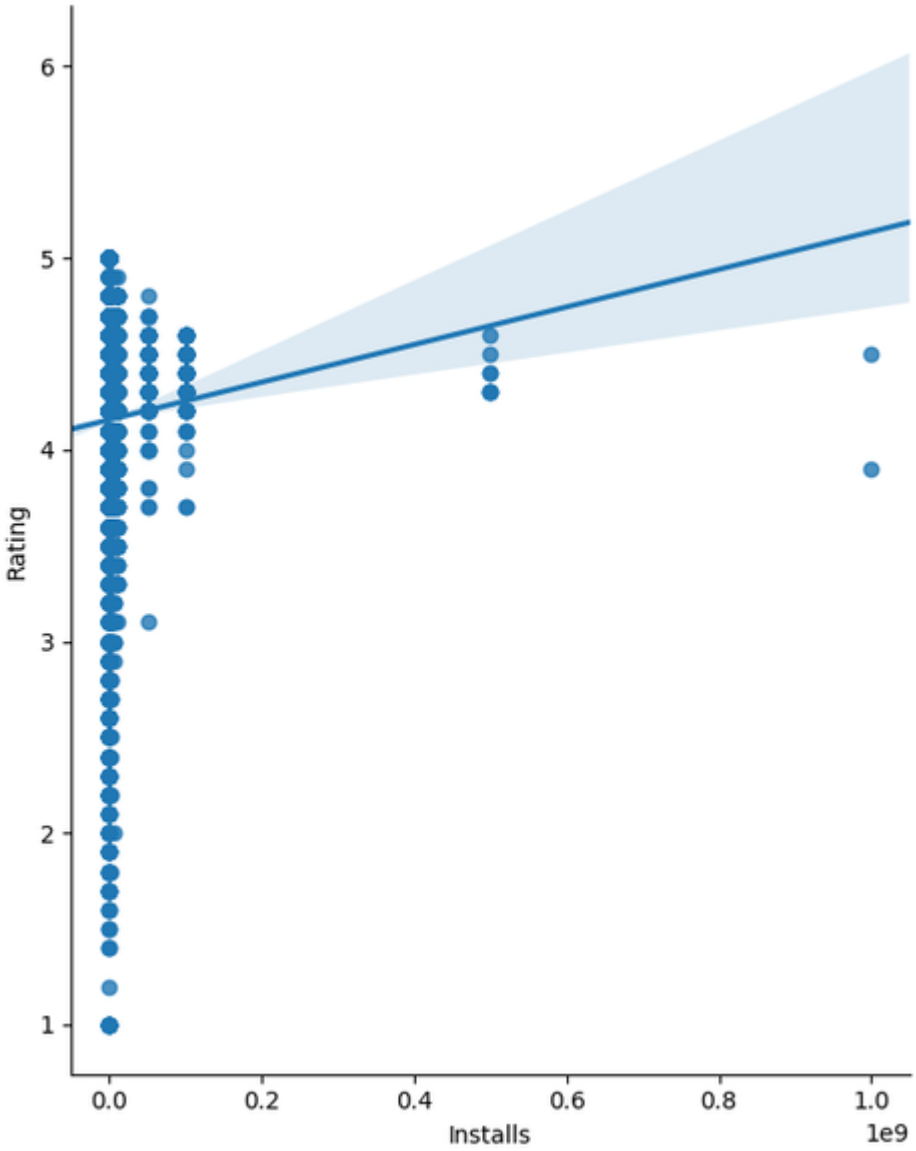
```
Installs             -0.105900  0.047610      0.012424  0.047610
Type Val              0.033900  0.043351      0.024750  0.043351
Price                -0.008735 -0.021140     -0.015356 -0.021140
Content Rating Val   -0.008309  0.034896     -0.111235  0.034896
Android Ver Val      -0.157756  0.039528      0.056668  0.039528

                        Size  Installs  Type Val     Price  \
Unnamed: 0         -0.077006 -0.105900  0.033900 -0.008735
Rating              0.063067  0.047610  0.043351 -0.021140
Category Val       -0.150375  0.012424  0.024750 -0.015356
Rating.1            0.063067  0.047610  0.043351 -0.021140
Reviews             0.180522  0.595755 -0.038898 -0.008994
Size                1.000000  0.131766 -0.016967 -0.025703
Installs            0.131766  1.000000 -0.046723 -0.010562
Type Val           -0.016967 -0.046723  1.000000  0.223314
Price              -0.025703 -0.010562  0.223314  1.000000
Content Rating Val  0.210481  0.045480 -0.031905 -0.012452
Android Ver Val     0.162624  0.034764 -0.099485  0.008559

                    Content Rating Val  Android Ver Val
Unnamed: 0                   -0.008309        -0.157756
Rating                        0.034896         0.039528
Category Val                 -0.111235         0.056668
Rating.1                      0.034896         0.039528
Reviews                       0.056289         0.014635
Size                          0.210481         0.162624
Installs                      0.045480         0.034764
Type Val                     -0.031905        -0.099485
Price                        -0.012452         0.008559
Content Rating Val            1.000000        -0.012705
Android Ver Val              -0.012705         1.000000
```

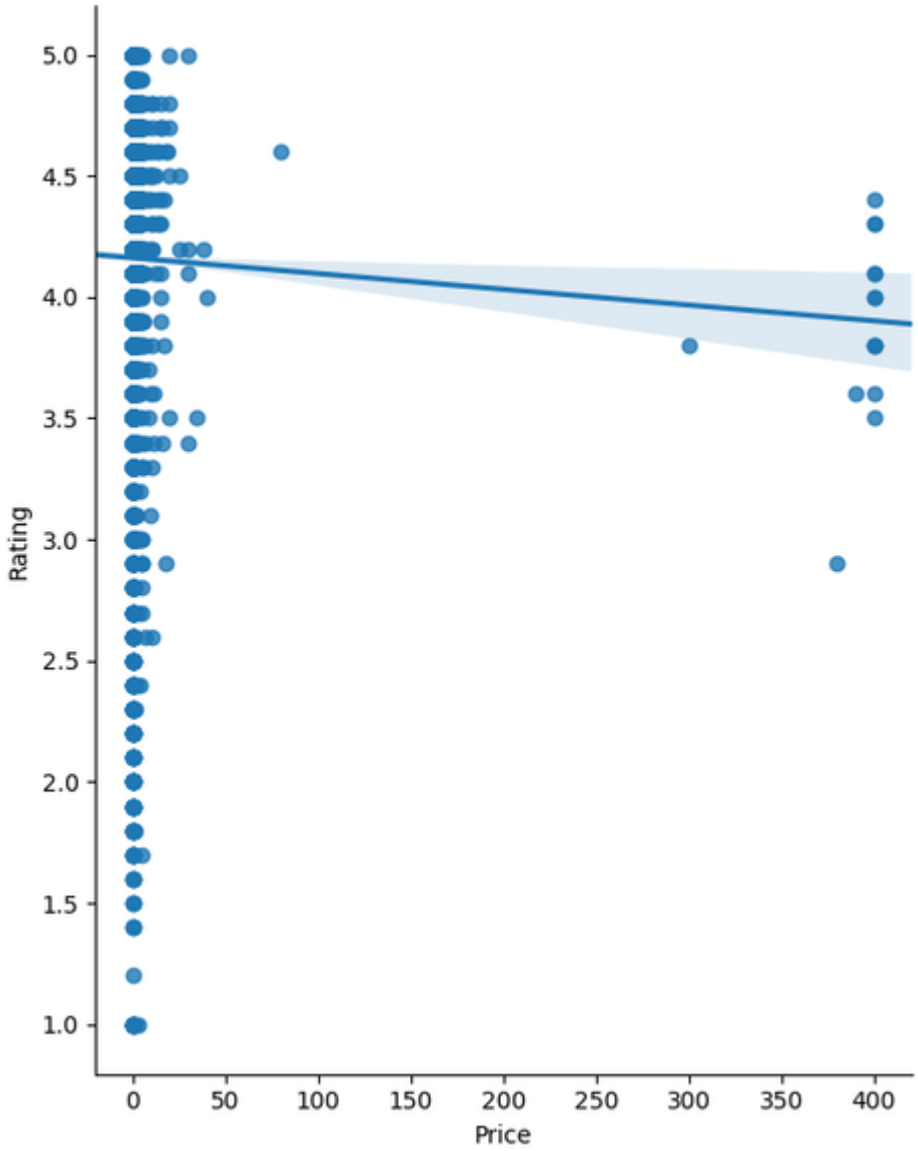
Add the best fitting straight lines and confidence bands through seaborn,judge the relationship Intuitively.

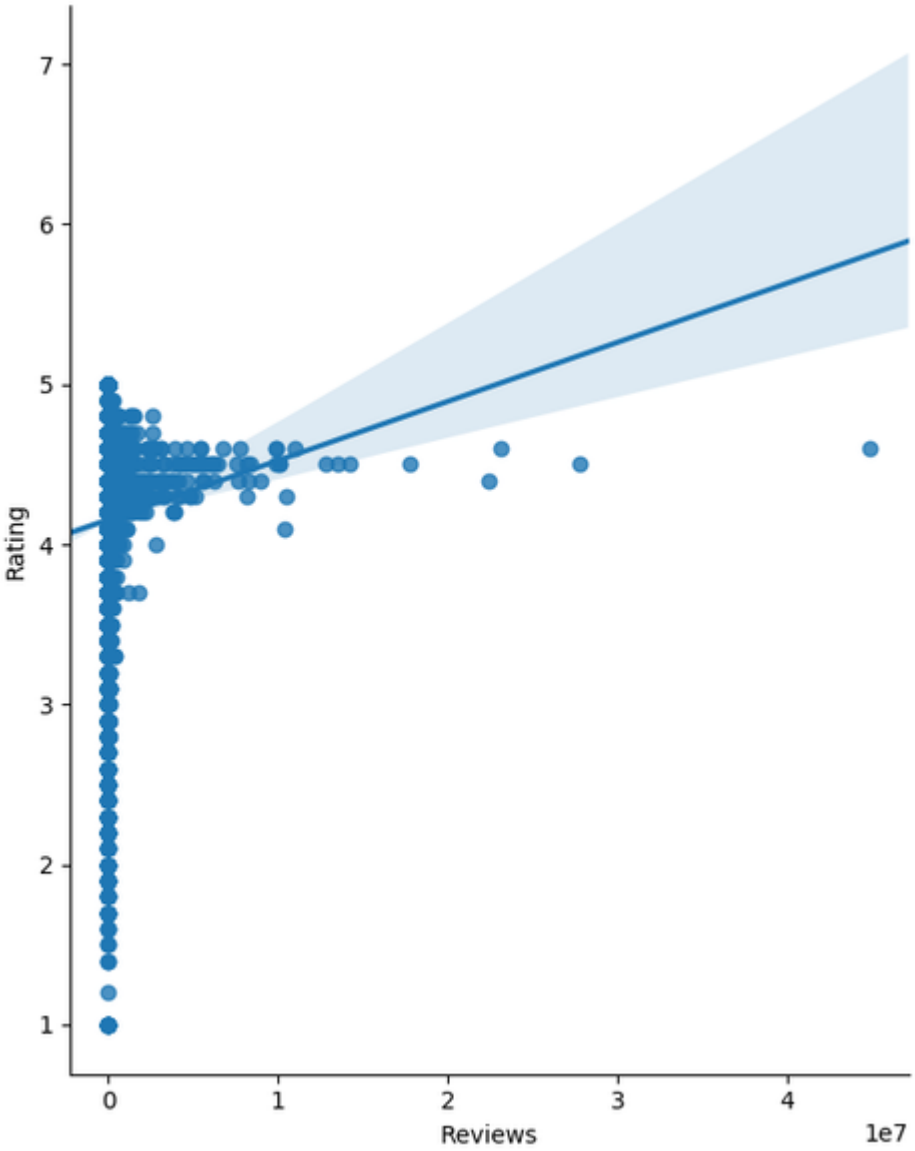
```
sns.pairplot(data, x_vars=['Category Val', 'Reviews','Installs',
], y_vars='Rating', height=7, aspect=0.8, kind='reg')
plt.show()
```

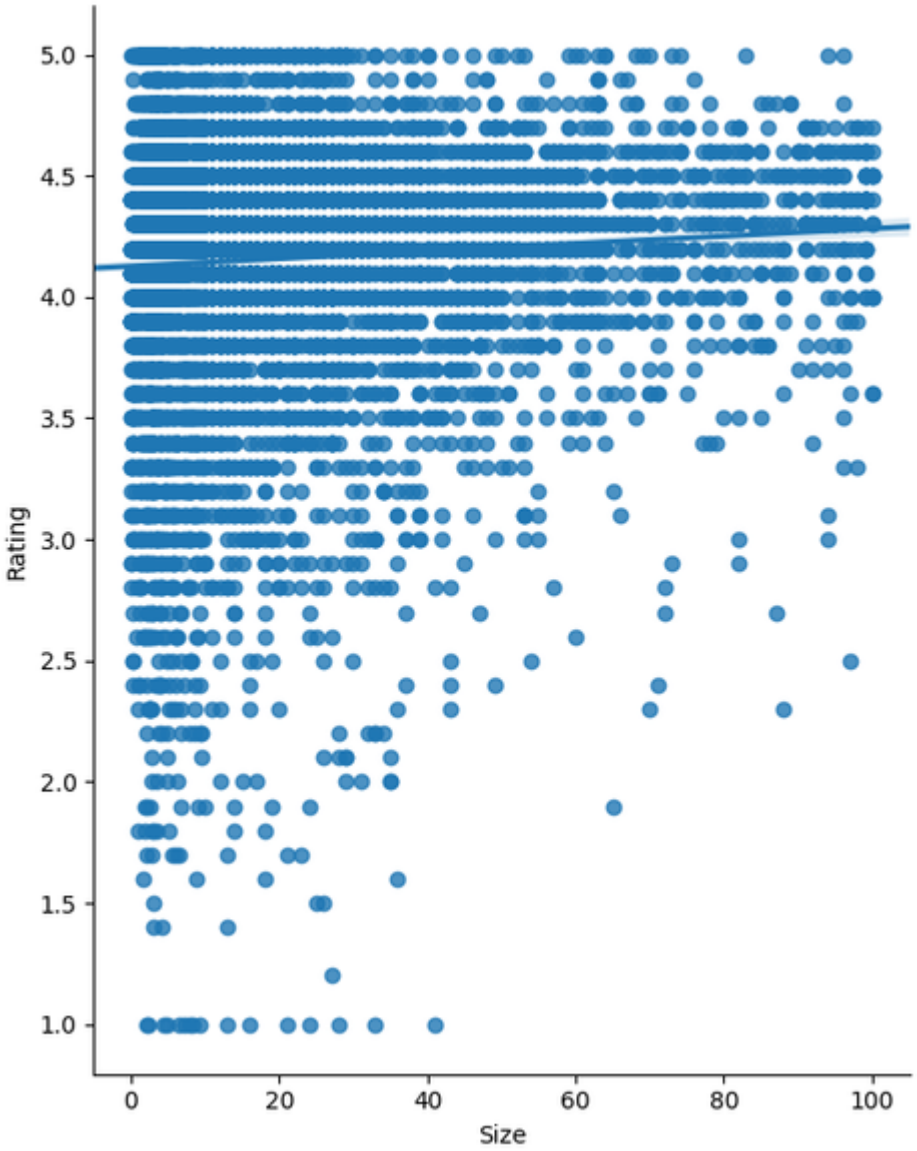
Result is shown as follows

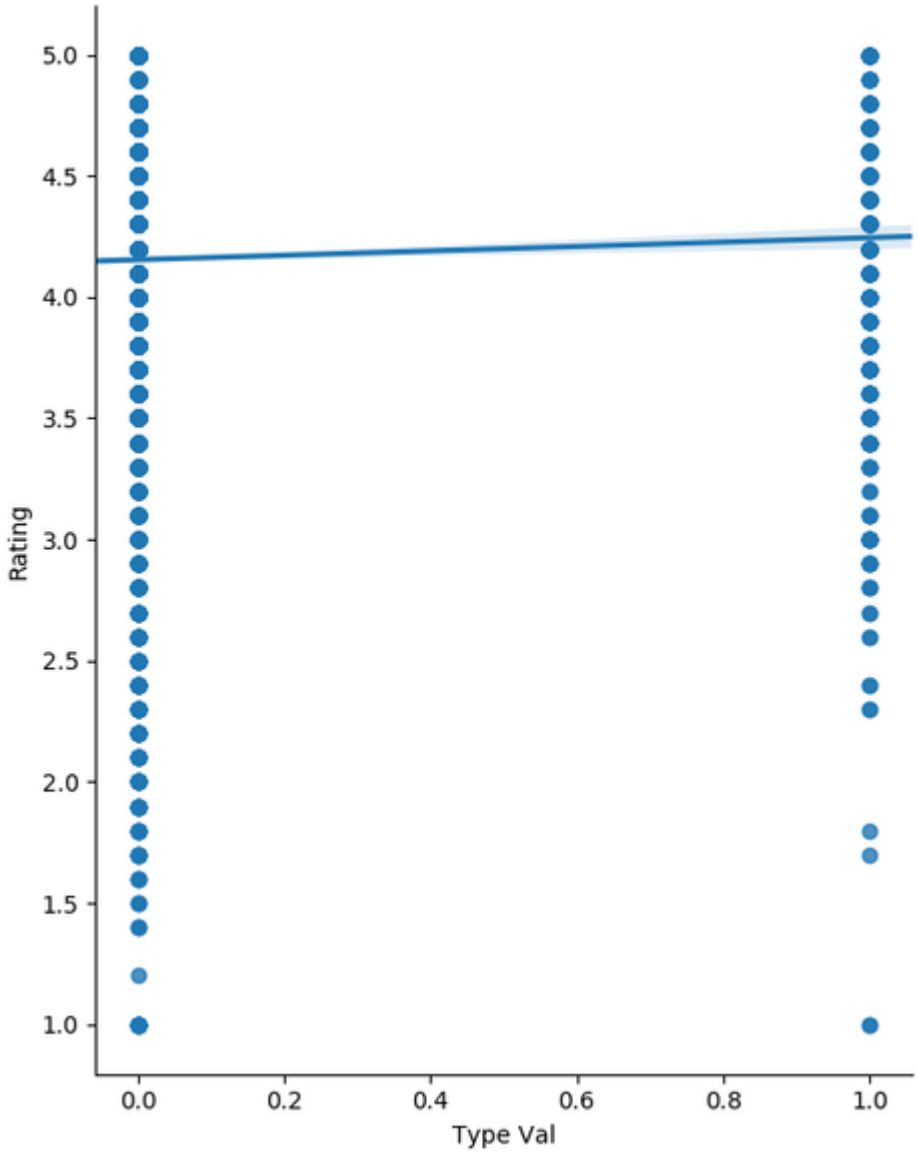



Rating
6
5
4
3
2
1
0.0
0.2
0.4
0.6
0.8
1.0
Installs
1e9


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Rating
0
50
100
150
200
250
300
350
400
Price


7
6
5
4
3
2
1
Rating
0
1
2
3
4
Reviews
1e7


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Rating
0
20
40
60
80
100
Size



Improvements:use linear_model to predict the score of app in Google Play Store,the results of the predications are not particularly satisfactory.The score's correlations with each dimension are not high,we should mining other data information for the further predication of the score