Ханунов Рк1 Вариант 15

rk1_TMO_khanunov

May 4, 2023

[25]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
import plotly.express as px
```

[26]:
```
df = pd.read_csv('googleplaystore.csv')
```

[27]:
```
df.head()
```

[27]:

	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	
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	
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	
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	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	

```
0  4.0.3 and up
1  4.0.3 and up
2  4.0.3 and up
3    4.2 and up
4    4.4 and up
```

[28]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   App             10841 non-null  object
 1   Category        10841 non-null  object
 2   Rating          9367 non-null   float64
 3   Reviews         10841 non-null  object
 4   Size            10841 non-null  object
 5   Installs        10841 non-null  object
 6   Type            10840 non-null  object
 7   Price           10841 non-null  object
 8   Content Rating  10840 non-null  object
 9   Genres          10841 non-null  object
 10  Last Updated    10841 non-null  object
 11  Current Ver     10833 non-null  object
 12  Android Ver     10838 non-null  object
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

[29]:
```
def miss(df):
    total = df.isnull().sum().sort_values(ascending=False)
    percent = (df.isnull().sum()/df.isnull().count()).sort_values(ascending=␣
  ↪False)
    missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Perce'])
    return missing_data
miss(df)
```

[29]:

	Total	Perce
Rating	1474	0.135965
Current Ver	8	0.000738
Android Ver	3	0.000277
Type	1	0.000092
Content Rating	1	0.000092
App	0	0.000000
Category	0	0.000000
Reviews	0	0.000000
Size	0	0.000000
Installs	0	0.000000

```
Price                0  0.000000
Genres               0  0.000000
Last Updated         0  0.000000
```

[30]:
```
total = df.shape[0]
```

[31]:
```
numb_col = []
for col in df.columns:
#
    temp_null_count = df[df[col].isnull()].shape[0]
    dt = str(df[col].dtype)
    if temp_null_count>0 and (dt=='float64' or dt =='int64') :
        numb_col.append(col)
        temp_perc = round((temp_null_count / total) * 100.0, 2)
        print('     {}.        {}.                 {}, {}%.'.format(numb_col,␣
 ↪dt, temp_null_count, temp_perc))
```

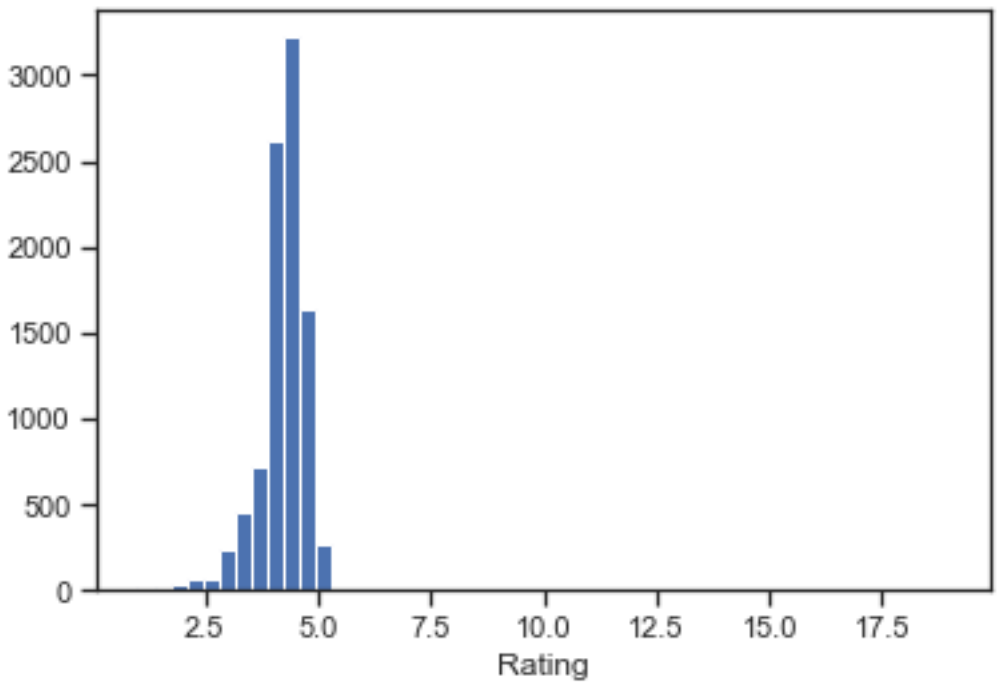
```
    ['Rating'].        float64.                 1474, 13.6%.
```

[32]:
```
df_num =df[numb_col]
df_num
```

[32]:
```
        Rating
0          4.1
1          3.9
2          4.7
3          4.5
4          4.3
…          …
10836      4.5
10837      5.0
10838      NaN
10839      4.5
10840      4.5

[10841 rows x 1 columns]
```

[33]:
```
for col in df_num:
    plt.hist(df[col], 50)
    plt.xlabel(col)
    plt.show()
```



[34]:
```
strategies=['mean', 'median', 'most_frequent']
```

[35]:
```
def test_num_impute_col(dataset, column, strategy_param):
    temp_data = dataset[[column]]
    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)
    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(temp_data)
    filled_data = data_num_imp[mask_missing_values_only]
    return column, strategy_param, filled_data.size, filled_data[0],␣
 ↪filled_data[1]
```

[36]:
```
test_num_impute_col(df, 'Rating', strategies[0])
```

[36]:
```
('Rating', 'mean', 1474, 4.193338315362443, 4.193338315362443)
```

[37]:
```
test_num_impute_col(df, 'Rating', strategies[1])
```

[37]:
```
('Rating', 'median', 1474, 4.3, 4.3)
```

[38]:
```
test_num_impute_col(df, 'Rating', strategies[2])
```

[38]:
```
('Rating', 'most_frequent', 1474, 4.4, 4.4)
```

[39]:
```
df['Rating'] = df['Rating'].fillna(df['Rating'].median())
```

[40]:
```
numb_col = []
for col in df.columns:
#
    temp_null_count = df[df[col].isnull()].shape[0]
    dt = str(df[col].dtype)
    if temp_null_count>0 and dt=='object' :
        numb_col.append(col)
        temp_perc = round((temp_null_count / total) * 100.0, 2)
        print('     {}.        {}.                 {}, {}%.'.format(numb_col,␣
 ↪dt, temp_null_count, temp_perc))
```

```
    ['Type'].        object.                 1, 0.01%.
    ['Type', 'Content Rating'].        object.
     1, 0.01%.
    ['Type', 'Content Rating', 'Current Ver'].        object.
         8, 0.07%.
    ['Type', 'Content Rating', 'Current Ver', 'Android Ver'].
object.                 3, 0.03%.
```

[41]:
```
df_num = df[numb_col]
df_num
```

[41]:

	Type	Content Rating	Current Ver	Android Ver
0	Free	Everyone	1.0.0	4.0.3 and up
1	Free	Everyone	2.0.0	4.0.3 and up
2	Free	Everyone	1.2.4	4.0.3 and up
3	Free	Teen	Varies with device	4.2 and up
4	Free	Everyone	1.1	4.4 and up
…	…	…	…	…
10836	Free	Everyone	1.48	4.1 and up
10837	Free	Everyone	1.0	4.1 and up
10838	Free	Everyone	1.0	2.2 and up
10839	Free	Mature 17+	Varies with device	Varies with device
10840	Free	Everyone	Varies with device	Varies with device

[10841 rows x 4 columns]

[42]:
```
cat_temp_data = df[['Type']]
cat_temp_data.head()
```

[42]:

	Type
0	Free
1	Free
2	Free
3	Free
4	Free

[43]:
```
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(cat_temp_data)
data_imp2
```

[43]:
```
array([['Free'],
       ['Free'],
       ['Free'],
       …,
       ['Free'],
       ['Free'],
       ['Free']], dtype=object)
```

[44]:
```
np.unique(data_imp2)
```

[44]:
```
array(['0', 'Free', 'Paid'], dtype=object)
```

[45]:
```
col = ['0', 'Free', 'Paid']
for i in col:
    k = data_imp2[data_imp2==i].size
    print('               {}     {}'.format(i, k))
```

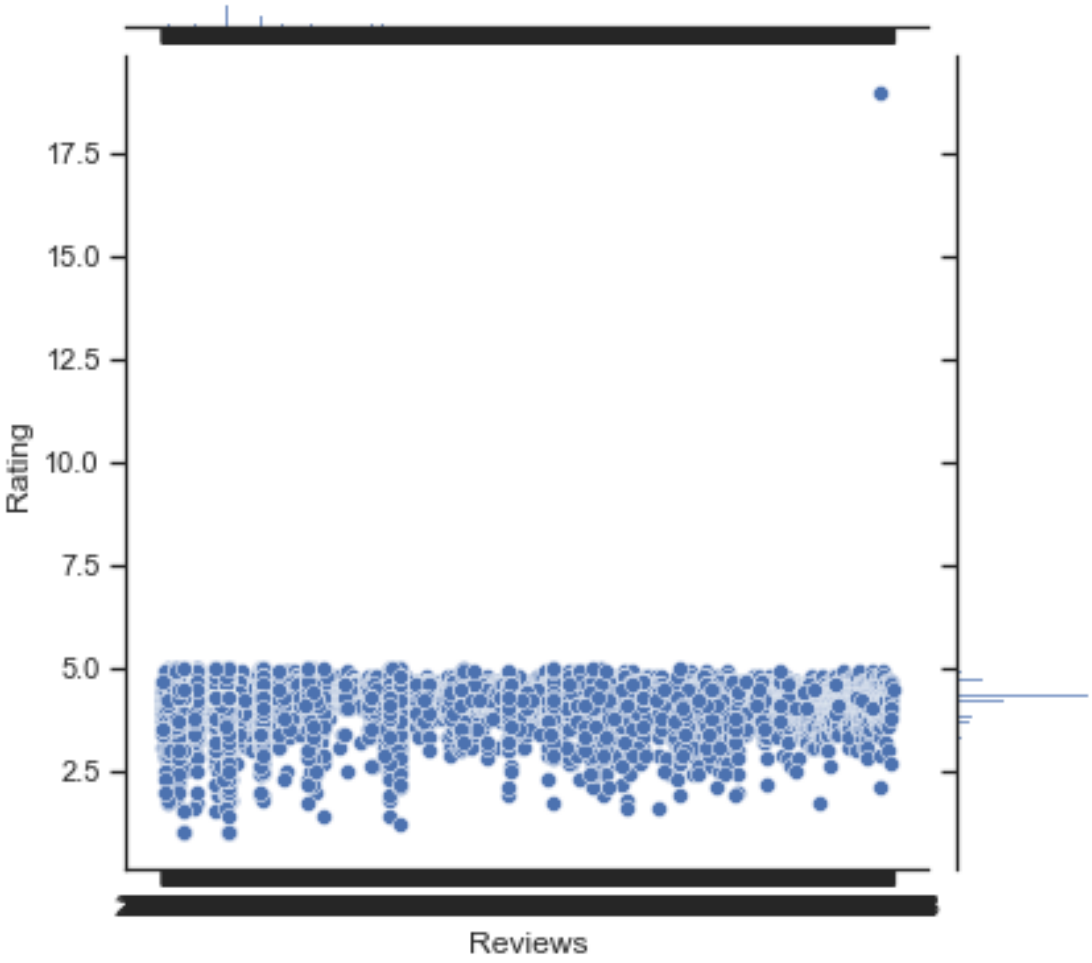
```
              0     1
              Free     10040
              Paid     800
```

[46]:
```
imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value=i)
data_imp3 = imp3.fit_transform(cat_temp_data)
data_imp3
```

[46]:
```
array([['Free'],
       ['Free'],
       ['Free'],
       …,
       ['Free'],
       ['Free'],
       ['Free']], dtype=object)
```

[47]:
```
df['Type']= df['Type'].fillna('unk')
```

[48]:
```
sns.jointplot(y='Rating', x='Reviews', data=df)
plt.show()
```


17.5
15.0
12.5
10.0
7.5
5.0
2.5
Rating
Reviews