In [1]:
```
import pandas as pd
```

In [2]:
```
cust=pd.read_excel("cust_data.xlsx")
cust
```

Out[2]:

	Cust_ID	Gender	Orders	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	...	LG	Dior	Scabal	Tommy Hilfiger	Hollister	Foreve 2
0	1	M	7	0	0	0	0	0	0	0	...	0	0	0	0	0	(
1	2	F	0	0	1	0	0	0	0	0	...	0	1	0	0	0	(
2	3	M	7	0	1	0	0	0	0	0	...	0	0	0	0	0	(
3	4	F	0	0	0	0	0	0	0	0	...	0	0	0	0	0	(
4	5	NaN	10	0	0	0	0	0	0	0	...	0	0	2	0	0	(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
29995	29996	M	0	0	0	0	0	0	0	0	...	0	0	0	0	0	(
29996	29997	M	1	0	1	0	0	0	0	0	...	0	0	0	0	0	(
29997	29998	M	0	0	1	0	0	0	0	0	...	0	0	0	0	0	(
29998	29999	M	0	0	0	0	0	0	0	0	...	0	0	0	0	0	(
29999	30000	F	3	2	0	0	1	0	0	0	...	0	0	0	0	0	(

30000 rows × 38 columns

In [3]:
```
cust.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 38 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Cust_ID                 30000 non-null  int64
 1   Gender                  27276 non-null  object
 2   Orders                  30000 non-null  int64
 3   Jordan                  30000 non-null  int64
 4   Gatorade                30000 non-null  int64
 5   Samsung                 30000 non-null  int64
 6   Asus                    30000 non-null  int64
 7   Udis                    30000 non-null  int64
 8   Mondelez International  30000 non-null  int64
 9   Wrangler                30000 non-null  int64
 10  Vans                    30000 non-null  int64
 11  Fila                    30000 non-null  int64
 12  Brooks                  30000 non-null  int64
 13  H&M                     30000 non-null  int64
 14  Dairy Queen             30000 non-null  int64
 15  Fendi                   30000 non-null  int64
 16  Hewlett Packard         30000 non-null  int64
 17  Pladis                  30000 non-null  int64
 18  Asics                   30000 non-null  int64
 19  Siemens                 30000 non-null  int64
 20  J.M. Smucker            30000 non-null  int64
 21  Pop Chips               30000 non-null  int64
 22  Juniper                 30000 non-null  int64
 23  Huawei                  30000 non-null  int64
 24  Compaq                  30000 non-null  int64
 25  IBM                     30000 non-null  int64
 26  Burberry                30000 non-null  int64
 27  Mi                      30000 non-null  int64
 28  LG                      30000 non-null  int64
 29  Dior                    30000 non-null  int64
 30  Scabal                  30000 non-null  int64
 31  Tommy Hilfiger          30000 non-null  int64
 32  Hollister               30000 non-null  int64
 33  Forever 21              30000 non-null  int64
```

```
 34  Colavita                30000 non-null  int64
 35  Microsoft               30000 non-null  int64
 36  Jiffy mix               30000 non-null  int64
 37  Kraft                   30000 non-null  int64
dtypes: int64(37), object(1)
memory usage: 8.7+ MB
```

In [4]:
```
cust.isnull().sum()
```

Out[4]:
```
Cust_ID                     0
Gender                   2724
Orders                      0
Jordan                      0
Gatorade                    0
Samsung                     0
Asus                        0
Udis                        0
Mondelez International      0
Wrangler                    0
Vans                        0
Fila                        0
Brooks                      0
H&M                         0
Dairy Queen                 0
Fendi                       0
Hewlett Packard             0
Pladis                      0
Asics                       0
Siemens                     0
J.M. Smucker                0
Pop Chips                   0
Juniper                     0
Huawei                      0
Compaq                      0
IBM                         0
Burberry                    0
Mi                          0
LG                          0
Dior                        0
Scabal                      0
Tommy Hilfiger              0
Hollister                   0
Forever 21                  0
Colavita                    0
Microsoft                   0
Jiffy mix                   0
Kraft                       0
dtype: int64
```

In [5]:
```
cust=pd.get_dummies(cust,drop_first=True)
```

In [6]:
```
cust
```

Out[6]:

	Cust_ID	Orders	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	Vans	...	Dior	Scabal	Tommy Hilfiger	Hollister	Forever 21	Cola
0	1	7	0	0	0	0	0	0	0	2	...	0	0	0	0	0	
1	2	0	0	1	0	0	0	0	0	0	...	1	0	0	0	0	
2	3	7	0	1	0	0	0	0	0	0	...	0	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	5	10	0	0	0	0	0	0	0	0	...	0	2	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
29996	29997	1	0	1	0	0	0	0	0	0	...	0	0	0	0	0	
29997	29998	0	0	1	0	0	0	0	0	1	...	0	0	0	0	0	
29998	29999	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
29999	30000	3	2	0	0	1	0	0	0	0	...	0	0	0	0	0	

30000 rows × 38 columns

In [7]:
```
cust.columns
```

Out[7]:
```
Index(['Cust_ID', 'Orders', 'Jordan', 'Gatorade', 'Samsung', 'Asus', 'Udis',
       'Mondelez International', 'Wrangler', 'Vans', 'Fila', 'Brooks', 'H&M',
       'Dairy Queen', 'Fendi', 'Hewlett Packard', 'Pladis', 'Asics', 'Siemens',
       'J.M. Smucker', 'Pop Chips', 'Juniper', 'Huawei', 'Compaq', 'IBM',
       'Burberry', 'Mi', 'LG', 'Dior', 'Scabal', 'Tommy Hilfiger', 'Hollister',
       'Forever 21', 'Colavita', 'Microsoft', 'Jiffy mix', 'Kraft',
       'Gender_M'],
      dtype='object')
```

In [8]:
```
cust.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 38 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Cust_ID                30000 non-null  int64
 1   Orders                 30000 non-null  int64
 2   Jordan                 30000 non-null  int64
 3   Gatorade               30000 non-null  int64
 4   Samsung                30000 non-null  int64
 5   Asus                   30000 non-null  int64
 6   Udis                   30000 non-null  int64
 7   Mondelez International 30000 non-null  int64
 8   Wrangler               30000 non-null  int64
 9   Vans                   30000 non-null  int64
 10  Fila                   30000 non-null  int64
 11  Brooks                 30000 non-null  int64
 12  H&M                    30000 non-null  int64
 13  Dairy Queen            30000 non-null  int64
 14  Fendi                  30000 non-null  int64
 15  Hewlett Packard        30000 non-null  int64
 16  Pladis                 30000 non-null  int64
 17  Asics                  30000 non-null  int64
 18  Siemens                30000 non-null  int64
 19  J.M. Smucker           30000 non-null  int64
 20  Pop Chips              30000 non-null  int64
 21  Juniper                30000 non-null  int64
 22  Huawei                 30000 non-null  int64
 23  Compaq                 30000 non-null  int64
 24  IBM                    30000 non-null  int64
 25  Burberry               30000 non-null  int64
 26  Mi                     30000 non-null  int64
 27  LG                     30000 non-null  int64
 28  Dior                   30000 non-null  int64
 29  Scabal                 30000 non-null  int64
 30  Tommy Hilfiger         30000 non-null  int64
 31  Hollister              30000 non-null  int64
 32  Forever 21             30000 non-null  int64
 33  Colavita               30000 non-null  int64
```

```
 34  Microsoft           30000 non-null  int64
 35  Jiffy mix           30000 non-null  int64
 36  Kraft               30000 non-null  int64
 37  Gender_M            30000 non-null  uint8
dtypes: int64(37), uint8(1)
memory usage: 8.5 MB
```

In [9]:

```
cust.describe()
```

Out[9]:

	Cust_ID	Orders	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	Vans
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.00000
mean	15000.500000	4.169800	0.267433	0.252333	0.222933	0.161333	0.143533	0.139767	0.106933	0.11143
std	8660.398374	3.590311	0.804778	0.705368	0.917494	0.740038	0.641258	0.525840	0.515921	0.54799
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	7500.750000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
50%	15000.500000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
75%	22500.250000	7.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
max	30000.000000	12.000000	24.000000	15.000000	27.000000	17.000000	14.000000	31.000000	9.000000	16.00000

8 rows × 38 columns

In [10]:
```
cust_corr=cust.corr()
cust_corr
```

Out[10]:

	Cust_ID	Orders	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	Vans	...	Dior	Scabal
Cust_ID	1.000000	0.029132	0.064277	0.021821	0.057206	0.049191	0.060677	0.035560	0.043043	0.002158	...	0.062489	0.057356
Orders	0.029132	1.000000	0.016090	0.034310	0.017885	0.015584	0.009018	0.008741	0.003856	0.006343	...	0.009027	0.020050
Jordan	0.064277	0.016090	1.000000	0.177254	0.071258	0.123731	0.135673	0.021950	0.040710	0.115796	...	0.061185	0.062582
Gatorade	0.021821	0.034310	0.177254	1.000000	0.063890	0.046215	0.058180	0.031272	0.088166	0.170620	...	0.056571	0.045958
Samsung	0.057206	0.017885	0.071258	0.063890	1.000000	0.012274	0.028785	0.035807	0.014565	0.029155	...	0.017369	0.046561
Asus	0.049191	0.015584	0.123731	0.046215	0.012274	1.000000	0.114588	0.018120	0.026668	0.046990	...	0.028911	0.043168
Udis	0.060677	0.009018	0.135673	0.058180	0.028785	0.114588	1.000000	0.040251	0.023128	0.020696	...	0.029567	0.068091
Mondelez International	0.035560	0.008741	0.021950	0.031272	0.035807	0.018120	0.040251	1.000000	0.021211	0.020795	...	0.034783	0.100657
Wrangler	0.043043	0.003856	0.040710	0.088166	0.014565	0.026668	0.023128	0.021211	1.000000	0.028595	...	0.054262	0.099995
Vans	0.002158	0.006343	0.115796	0.170620	0.029155	0.046990	0.020696	0.020795	0.028595	1.000000	...	0.014776	-0.011961
Fila	-0.000450	-0.009627	0.031611	0.026350	0.024847	-0.004766	0.021717	0.026681	-0.002237	0.002731	...	0.001074	0.021368
Brooks	0.039574	0.015389	0.165471	0.154345	0.089334	0.114384	0.089850	0.039200	0.021861	0.122174	...	0.030597	0.027999
H&M	0.023426	0.030833	0.038302	0.066794	0.031444	0.025349	0.037187	0.043001	0.070330	0.024992	...	0.138741	0.149119
Dairy Queen	-0.005785	-0.002705	0.043857	0.055532	0.014086	0.028589	0.013806	0.022947	0.030978	0.031384	...	0.047956	0.037345
Fendi	0.044053	0.017077	0.030369	0.029120	0.015095	0.006580	0.011772	0.001657	0.024073	0.012768	...	0.038278	0.021712
Hewlett Packard	0.048083	0.006867	0.008800	0.016680	0.011988	0.006500	0.014293	0.012106	0.034098	0.002760	...	0.044929	0.031425
Pladis	0.012790	-0.000602	0.062050	0.062749	0.025422	0.030216	0.022155	0.053772	0.040609	0.032070	...	0.076033	0.053528
Asics	0.000715	0.022064	0.009816	0.042912	0.004546	0.010339	-0.008444	0.018303	-0.002145	-0.000852	...	0.012582	0.009692
Siemens	-0.039673	0.007843	-0.011082	0.006296	-0.011741	-0.000218	-0.014267	-0.002722	-0.010719	-0.008563	...	-0.013903	-0.013835
J.M. Smucker	0.011654	0.030807	0.054025	0.060775	0.026365	0.023046	0.013708	0.059322	0.020302	0.030542	...	0.057555	0.082324
Pop Chips	0.046623	0.018774	0.081462	0.087545	0.048772	0.022716	0.020427	0.031040	0.043520	0.085626	...	0.068831	0.028940

	Cust_ID	Orders	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	Vans	...	Dior	Scabal
Juniper	-0.015516	-0.002741	0.025708	0.030956	0.001621	0.017840	0.014118	0.019580	0.015098	0.010213	...	0.011408	0.017918
Huawei	0.018493	0.008031	0.051468	0.042281	0.024820	0.033468	0.024393	0.021313	0.030264	0.028003	...	0.073317	0.083139
Compaq	-0.009158	0.002765	0.007495	0.008773	0.004358	0.005892	0.002922	0.002441	-0.000865	0.015774	...	-0.006872	0.000107
IBM	-0.026589	-0.007647	0.001963	-0.009290	-0.005467	-0.007446	-0.003076	0.000471	-0.006730	-0.003649	...	-0.020813	-0.017099
Burberry	0.061117	0.015813	0.096492	0.102216	0.047852	0.047276	0.040914	0.065318	0.074666	0.058406	...	0.162370	0.126864
Mi	-0.031839	0.010369	0.022963	0.033103	0.004060	0.016904	0.013516	0.011150	0.024924	-0.009476	...	0.078787	0.097718
LG	0.029693	-0.006382	0.070205	0.066443	0.058584	0.025722	0.030406	0.036895	0.079583	0.073194	...	0.064059	0.054640
Dior	0.062489	0.009027	0.061185	0.056571	0.017369	0.028911	0.029567	0.034783	0.054262	0.014776	...	1.000000	0.154839
Scabal	0.057356	0.020050	0.062582	0.045958	0.046561	0.043168	0.068091	0.100657	0.099995	-0.011961	...	0.154839	1.000000
Tommy Hilfiger	0.016463	0.003550	0.063739	0.058190	0.015502	0.039139	0.023367	0.056306	0.045565	0.021992	...	0.124796	0.155917
Hollister	0.084793	0.001680	0.026350	0.050302	0.021566	0.017283	0.032484	0.036270	0.075141	0.007039	...	0.091280	0.136622
Forever 21	0.049231	-0.003436	0.024710	0.031495	0.016170	0.014186	0.035360	0.044912	0.066978	0.002500	...	0.082791	0.117239
Colavita	0.002061	0.007455	0.015564	0.018279	0.005584	0.009605	0.000437	0.014344	0.006188	0.000045	...	0.044994	0.008717
Microsoft	-0.005614	0.015307	0.015804	0.016625	0.003640	0.019748	0.012679	0.013070	0.008203	-0.002454	...	0.034829	0.019889
Jiffy mix	-0.019145	0.011268	0.021651	0.038655	0.009303	0.008001	0.004808	0.025066	0.015954	0.008639	...	0.055948	0.030452
Kraft	0.022508	-0.007160	0.018918	0.018553	0.006633	0.007530	0.006795	0.019727	0.014749	0.013326	...	0.064123	0.014600
Gender_M	-0.060798	0.016879	0.027470	0.127665	-0.006435	-0.094381	-0.077943	-0.045117	-0.087144	0.140070	...	-0.074158	-0.180412

38 rows × 38 columns

In [11]:
```
from sklearn.cluster import KMeans
```

In [12]:
```
kmeans_cluster=KMeans(2)
```

In [13]:
```
kmeans_cluster.fit(cust)
```

Out[13]:
```
KMeans(n_clusters=2)
```

In [14]:
```
cust_data=cust.copy()
```

In [15]:
```
cust_data["km_predicted"]=kmeans_cluster.predict(cust)
```

In [16]:
```
cust_data
```

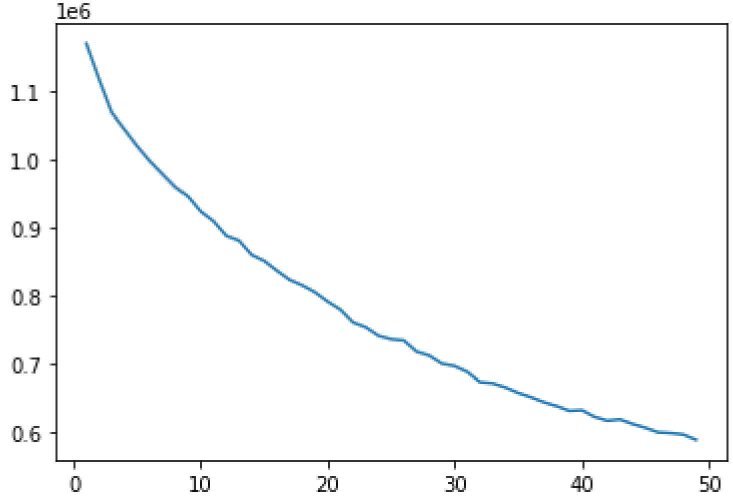
Out[16]:

	Cust_ID	Orders	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	Vans	...	Scabal	Tommy Hilfiger	Hollister	Forever 21	Colavita	I
0	1	7	0	0	0	0	0	0	0	2	...	0	0	0	0	0	
1	2	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	
2	3	7	0	1	0	0	0	0	0	0	...	0	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	5	10	0	0	0	0	0	0	0	0	...	2	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
29996	29997	1	0	1	0	0	0	0	0	0	...	0	0	0	0	0	
29997	29998	0	0	1	0	0	0	0	0	1	...	0	0	0	0	0	
29998	29999	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
29999	30000	3	2	0	0	1	0	0	0	0	...	0	0	0	0	0	

30000 rows × 39 columns

In [17]:
```
import matplotlib.pyplot as plt
from sklearn import preprocessing
x_scaled=preprocessing.scale(cust_data)
```

In [18]:
```
a=[]
for i in range(1,50):
    kmeans_em=KMeans(i)
    kmeans_em.fit(x_scaled)
    a.append(kmeans_em.inertia_)
plt.plot(range(1,50),a);
```



In [19]:
```
kmeans_cluster_new=KMeans(5)
kmeans_cluster_new.fit(x_scaled)
cust_data["km_predicted_new"]=kmeans_cluster_new.predict(x_scaled)
cust_data
```

Out[19]:

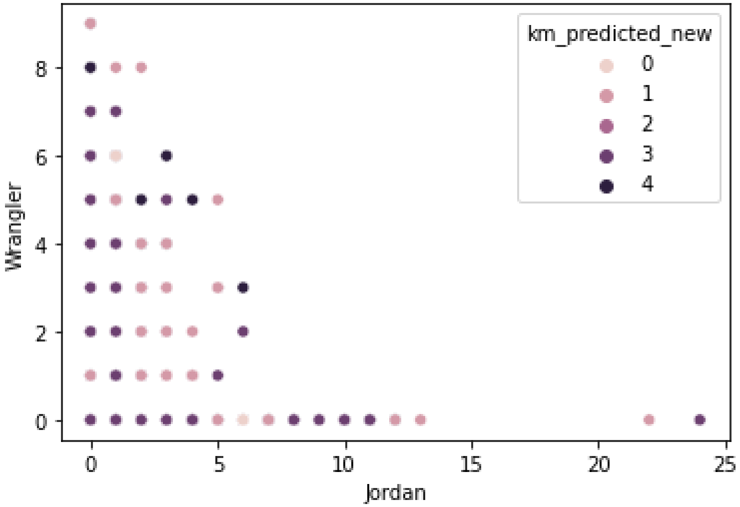
	Cust_ID	Orders	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	Vans	...	Tommy Hilfiger	Hollister	Forever 21	Colavita	Microsoft
0	1	7	0	0	0	0	0	0	0	2	...	0	0	0	0	0
1	2	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0
2	3	7	0	1	0	0	0	0	0	0	...	0	0	0	0	1
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	5	10	0	0	0	0	0	0	0	0	...	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	29996	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
29996	29997	1	0	1	0	0	0	0	0	0	...	0	0	0	0	0
29997	29998	0	0	1	0	0	0	0	0	1	...	0	0	0	0	0
29998	29999	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
29999	30000	3	2	0	0	1	0	0	0	0	...	0	0	0	0	0

30000 rows × 40 columns

In [20]:
```
import seaborn as sns
sns.scatterplot(cust_data["Jordan"],cust_data['Wrangler'],hue='km_predicted_new',data=cust_data)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as ke
yword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[20]:
```
<AxesSubplot:xlabel='Jordan', ylabel='Wrangler'>
```



In [21]:
```
from sklearn.metrics import silhouette_score
```

In [22]:
```
for i in range(3,10):
    kmeans=KMeans(n_clusters=i,max_iter=50)
    kmeans.fit(x_scaled)
    cluster_label=kmeans.labels_
    silhouette_avg=silhouette_score(x_scaled,cluster_label)
    print("n_cluster={0},the silhouette_score {1}".format(i,silhouette_avg))
```

```
n_cluster=3,the silhouette_score 0.0676564665775785
n_cluster=4,the silhouette_score 0.06928350797513552
n_cluster=5,the silhouette_score 0.06961935951720659
n_cluster=6,the silhouette_score 0.07191643986482621
n_cluster=7,the silhouette_score 0.06992694548223002
n_cluster=8,the silhouette_score 0.07391793017197527
n_cluster=9,the silhouette_score 0.07585043357168024
```

In [25]:
```
!pip install yellowbrick
```

```
Collecting yellowbrick
  Using cached yellowbrick-1.3.post1-py3-none-any.whl (271 kB)
Requirement already satisfied: scikit-learn>=0.20 in c:\programdata\anaconda3\lib\site-packages (from yellowbrick) (0.2
4.1)
Collecting numpy<1.20,>=1.16.0
  Using cached numpy-1.19.5-cp38-cp38-win_amd64.whl (13.3 MB)
Requirement already satisfied: cycler>=0.10.0 in c:\programdata\anaconda3\lib\site-packages (from yellowbrick) (0.10.0)
Requirement already satisfied: scipy>=1.0.0 in c:\programdata\anaconda3\lib\site-packages (from yellowbrick) (1.6.2)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in c:\programdata\anaconda3\lib\site-packages (from yellowbric
k) (3.3.4)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages (from cycler>=0.10.0->yellowbrick) (1.
15.0)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib!=3.0.0,>=2.
0.2->yellowbrick) (8.2.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib!=3.
0.0,>=2.0.2->yellowbrick) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib!=3.0.0,
>=2.0.2->yellowbrick) (1.3.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\programdata\anaconda3\lib\site-packages
(from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.4.7)
Requirement already satisfied: joblib>=0.11 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn>=0.20->yel
lowbrick) (1.0.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn>=
0.20->yellowbrick) (2.1.0)
Installing collected packages: numpy, yellowbrick
  Attempting uninstall: numpy
    Found existing installation: numpy 1.20.1
    Uninstalling numpy-1.20.1:


ERROR: Could not install packages due to an OSError: [WinError 5] Access is denied: 'c:\\programdata\\anaconda3\\lib\\s
ite-packages\\numpy-1.20.1.dist-info\\direct_url.json'
Consider using the `--user` option or check the permissions.
```

In [ ]:
```

```