In [217]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn import tree
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [2]:
```
history = pd.read_csv('/Users/hunterberberich/Desktop/DAAN862/click_history.csv')
product = pd.read_csv('/Users/hunterberberich/Desktop/DAAN862/product_features.csv')
user = pd.read_csv('/Users/hunterberberich/Desktop/DAAN862/user_features.csv')
```

In [3]:
```
history
```

Out[3]:

	user_id	product_id	clicked
0	104863	1350	False
1	108656	1321	True
2	100120	1110	False
3	104838	1443	True
4	107304	1397	True
...	...	...	...
35985	111177	1926	False
35986	111937	1699	True
35987	105921	1018	False
35988	106064	1082	True
35989	100664	1063	True

35990 rows × 3 columns

In [4]:
```
product
```

Out[4]:

	product_id	category	on_sale	number_of_reviews	avg_review_score
0	1134	tools	False	101	3.349452
1	1846	skincare	False	111	5.000000
2	1762	fragrance	False	220	4.882706
3	1254	hair	True	446	5.000000
4	1493	body	True	513	-1.000000
...	...	...	...	...	...
995	1541	fragrance	False	312	1.149903
996	1326	body	True	603	1.277216
997	1062	foot	True	705	3.602008
998	1721	makeup	True	306	1.846254
999	1995	men_skincare	True	199	3.576706

1000 rows × 5 columns

In [5]:
```
user
```

Out[5]:

	user_id	number_of_clicks_before	ordered_before	personal_interests
0	104939	2	True	['body', 'makeup', 'nail', 'hand', 'foot', 'me...
1	101562	2	True	['men_skincare', 'men_fragrance', 'tools', 'sk...
2	102343	2	True	['tools', 'makeup', 'foot', 'nail']
3	106728	5	True	['hand', 'men_skincare']
4	107179	0	True	['makeup', 'body', 'skincare', 'foot', 'men_sk...
...	...	...	...	...
11995	105121	6+	True	['fragrance', 'hand', 'makeup', 'men_fragrance...
11996	102607	1	False	['body', 'makeup', 'tools', 'fragrance', 'nail...
11997	106873	2	False	['hair', 'tools', 'men_skincare', 'hand', 'bod...
11998	107769	NaN	True	['hand', 'men_skincare', 'tools', 'makeup', 'h...
11999	103190	0	True	['men_fragrance', 'makeup', 'men_skincare', 'h...

12000 rows × 4 columns

In [6]:
```
#Merging Data sets in order to explore data
```

In [7]:
```
merged = pd.merge(history, user, on = 'user_id', how = 'left')
```

In [8]:
```
SephoraData = pd.merge(merged, product, on='product_id',
how='left')
```

In [9]:
```
SephoraData.describe()
```

Out[9]:

	user_id	product_id	number_of_reviews	avg_review_score
count	35990.000000	35990.000000	3.599000e+04	35990.000000
mean	106017.080161	1500.232898	1.138828e+05	2.650815
std	3483.480090	288.101984	4.987392e+05	1.736823
min	100001.000000	1000.000000	6.600000e+01	-1.000000
25%	102976.500000	1250.000000	2.570000e+02	1.430110
50%	106060.000000	1503.000000	4.710000e+02	2.761251
75%	109049.000000	1749.000000	7.000000e+02	4.165674
max	111999.000000	1999.000000	2.307390e+06	5.000000

In [10]:
```
SephoraData.isnull().sum()
```

Out[10]:
```
user_id                     0
product_id                  0
clicked                     0
number_of_clicks_before    1565
ordered_before              0
personal_interests          0
category                    0
on_sale                     0
number_of_reviews           0
avg_review_score            0
dtype: int64
```

In [11]:
```
SephoraData = SephoraData.dropna()
```

In [12]:
```
SephoraData.dtypes
```

Out[12]:
```
user_id                     int64
product_id                  int64
clicked                      bool
number_of_clicks_before    object
ordered_before               bool
personal_interests         object
category                   object
on_sale                      bool
number_of_reviews           int64
avg_review_score          float64
dtype: object
```

In [13]:
```
#Removed negative values from 'avg_review_score'
```

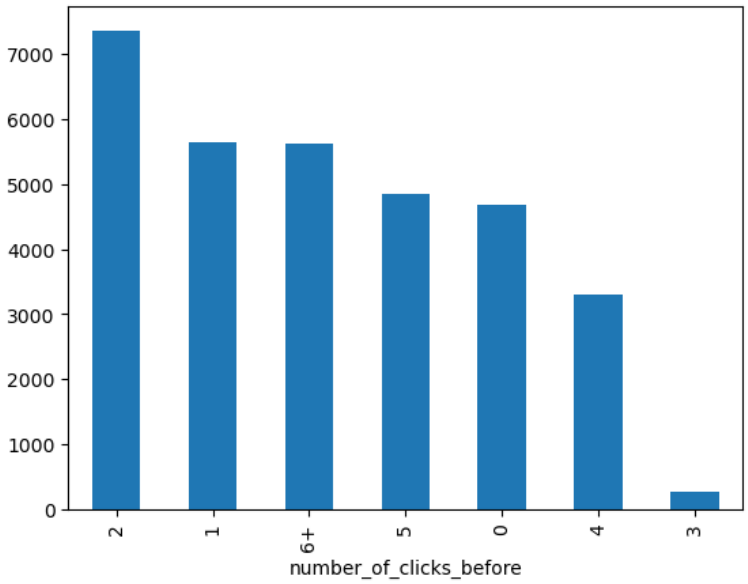
In [14]:
```
SephoraData = SephoraData[(SephoraData['avg_review_score'] > 0) & (SephoraData['avg_review_score'] <= 5)]
```

In [17]:
```
group = SephoraData['number_of_clicks_before'].value_counts()
```

In [18]:
```
group
```

Out[18]:
```
number_of_clicks_before
2     7357
1     5645
6+    5617
5     4854
0     4691
4     3294
3      270
Name: count, dtype: int64
```

In [19]:
```
group.plot(kind = "bar")
plt.show()
```



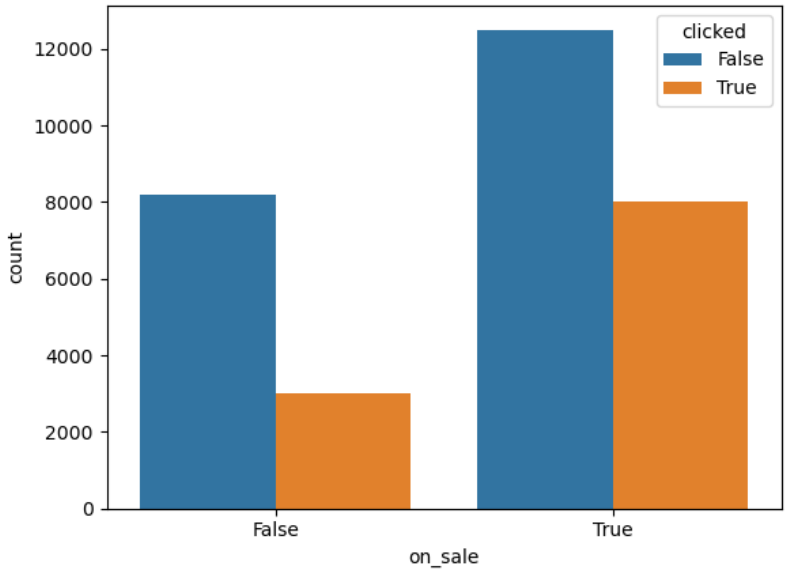
In [286]:
```
pd.crosstab(SephoraData['clicked'], SephoraData['on_sale'], margins = True)
```

Out[286]:

on_sale	False	True	All
clicked			
False	8207	12507	20714
True	3007	8007	11014
All	11214	20514	31728

In [287]:
```
sns.countplot(x = "on_sale", hue = "clicked", data = SephoraData)
```

Out[287]:
```
<Axes: xlabel='on_sale', ylabel='count'>
```



In [20]:
```
SephoraData['number_of_clicks_before'] = SephoraData['number_of_clicks_before'].str.replace("+", "")
```

In [22]:
```
SephoraData['number_of_clicks_before'] = pd.to_numeric(SephoraData['number_of_clicks_before'])
```

In [23]:
```
#removed '+' and turned column to numeric
```

In [24]:
```
SephoraData.describe()
```

Out[24]:

	user_id	product_id	number_of_clicks_before	number_of_reviews	avg_review_score
count	31728.000000	31728.000000	31728.000000	3.172800e+04	31728.000000
mean	106015.692511	1505.729671	2.909638	1.185787e+05	2.958195
std	3488.594637	289.836466	2.138991	5.084132e+05	1.437156
min	100001.000000	1000.000000	0.000000	6.600000e+01	0.008042
25%	102970.000000	1253.000000	1.000000	2.540000e+02	1.751859
50%	106059.000000	1512.000000	2.000000	4.710000e+02	2.969515
75%	109062.000000	1754.000000	5.000000	7.070000e+02	4.219748
max	111999.000000	1999.000000	6.000000	2.307390e+06	5.000000

In [25]:
```
#RemovingOutliers
```

In [26]:
```
from scipy import stats
```

In [27]:
```
z = np.abs(stats.zscore(SephoraData['number_of_reviews']))
threshold = 3
```

In [28]:
```
outliers = SephoraData[z > threshold]
```

In [29]:
```
print(outliers['number_of_reviews'])
```

```
79       2307064
87       2307275
112      2306733
123      2306407
124      2306318
          ...
35912    2306857
35926    2307240
35943    2306552
35969    2306762
35973    2307064
Name: number_of_reviews, Length: 1625, dtype: int64
```

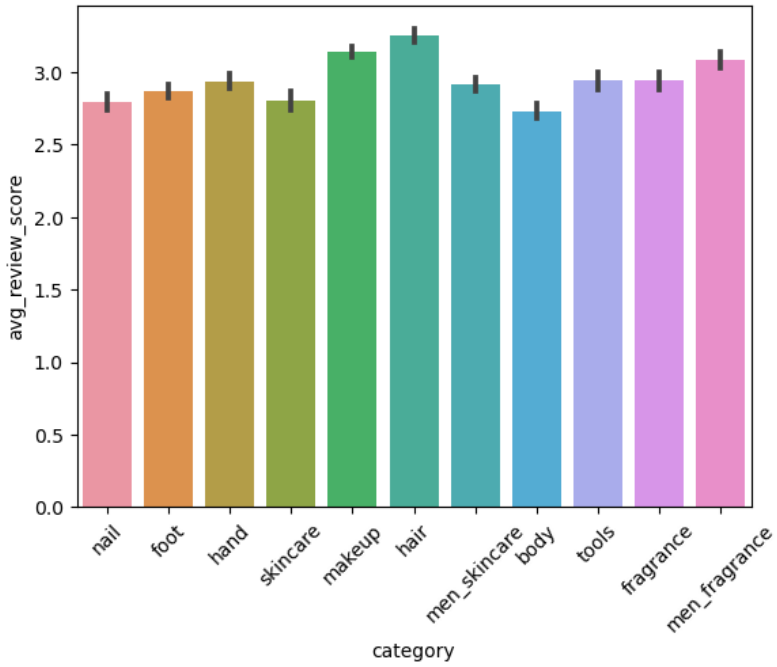
In [30]:
```
SephoraData2 = SephoraData.drop(outliers.index)
```

In [31]:
```
SephoraData2.describe()
```

Out[31]:

	user_id	product_id	number_of_clicks_before	number_of_reviews	avg_review_score
count	30103.000000	30103.000000	30103.00000	30103.000000	30103.000000
mean	106005.109325	1503.810883	2.91157	456.444275	2.993609
std	3490.839076	289.067396	2.13998	237.046228	1.430708
min	100001.000000	1000.000000	0.00000	66.000000	0.008042
25%	102958.000000	1251.000000	1.00000	249.000000	1.801611
50%	106044.000000	1511.000000	2.00000	443.000000	3.030499
75%	109058.000000	1751.000000	5.00000	670.000000	4.258360
max	111999.000000	1999.000000	6.00000	872.000000	5.000000

In [32]:
```
sns.barplot(x = SephoraData.category, y = SephoraData.avg_review_score)
plt.xticks(rotation = 45)
plt.show()
```



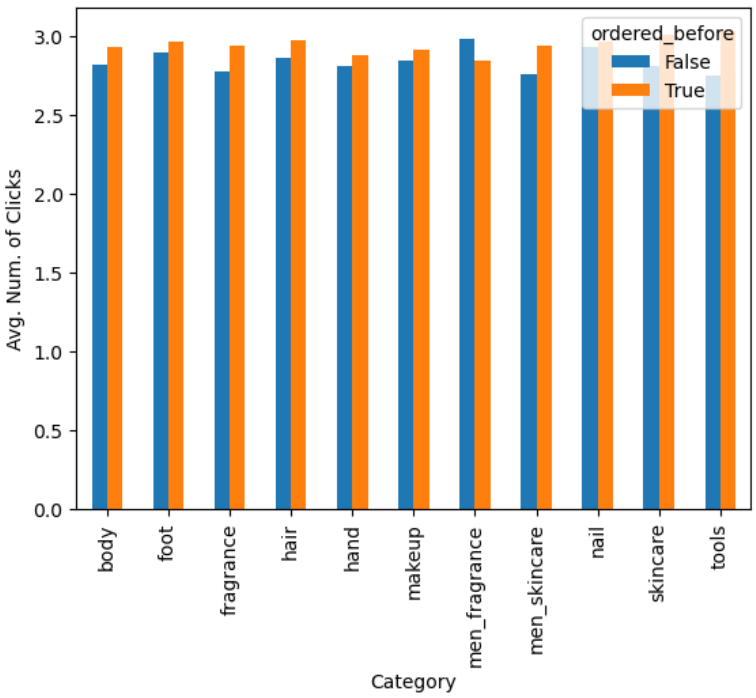
In [39]:
```
group1 = SephoraData2.groupby(["category", "ordered_before"])["number_of_clicks_before"].mean().unstack()
```

In [40]:
```
group1
```

Out[40]:

ordered_before	False	True
category		
body	2.817422	2.927320
foot	2.900187	2.965631
fragrance	2.779479	2.937538
hair	2.864583	2.977880
hand	2.811550	2.879234
makeup	2.845881	2.917677
men_fragrance	2.981586	2.847866
men_skincare	2.759909	2.940103
nail	2.929619	2.962033
skincare	2.811897	3.010623
tools	2.753507	3.030435

In [41]:
```
group1.plot(kind = "bar")
plt.xlabel("Category")
plt.ylabel("Avg. Num. of Clicks")
plt.show()
```



In [36]:
```
#On average, more users bought products the more number of clicks.
```

In [42]:
```
numeric_cols = ['number_of_clicks_before', 'number_of_reviews', 'avg_review_score']
```

In [235]:
```
scaler = MinMaxScaler()
```

In [150]:
```
SephoraData2[numeric_cols] = scaler.fit_transform(SephoraData2[numeric_cols])
```

In [151]:
```
SephoraData2
```

Out[151]:

	user_id	product_id	clicked	number_of_clicks_before	ordered_before	personal_interests	category	on_sale	number_of_reviews	avg_review_score
0	104863	1350	False	0.333333	True	['hair', 'body']	nail	False	0.086849	0.529916
1	108656	1321	True	0.166667	True	['tools', 'hair', 'skincare', 'nail', 'men_ski...	foot	True	0.415633	0.390205
2	100120	1110	False	0.000000	True	['tools', 'men_fragrance', 'hair', 'foot', 'na...	hand	True	0.024814	0.611695
3	104838	1443	True	0.000000	True	['tools', 'makeup', 'skincare', 'body', 'fragr...	skincare	False	0.679901	0.772656
4	107304	1397	True	0.833333	True	['hair', 'men_fragrance', 'foot', 'fragrance']	makeup	True	0.089330	1.000000
...	...	...	...	...	...	...	...	...	...	...
35984	109366	1148	False	0.166667	False	['men_fragrance', 'men_skincare', 'fragrance',...	foot	True	0.071960	0.180658
35985	111177	1926	False	0.166667	False	['skincare', 'nail', 'tools', 'body', 'hand', ...	fragrance	False	0.321340	0.960176
35987	105921	1018	False	0.833333	False	['tools', 'fragrance']	foot	True	0.177419	0.366834
35988	106064	1082	True	0.333333	False	['hair', 'foot', 'men_fragrance', 'tools', 'ha...	makeup	True	0.107940	0.536627
35989	100664	1063	True	0.000000	False	['men_fragrance', 'makeup', 'tools', 'nail', '...	makeup	True	0.151365	0.593249

30103 rows × 10 columns

In [152]:
```
#Multiple times I tried to use 'personal_interest' in my model but it never worked.
```

In [153]:
```
Sephora2 = SephoraData2.drop(columns = ['personal_interests', 'product_id', 'user_id'])
```

In [154]:
```
data = Sephora2
```

In [155]:
```
data
```

Out[155]:

	clicked	number_of_clicks_before	ordered_before	category	on_sale	number_of_reviews	avg_review_score
0	False	0.333333	True	nail	False	0.086849	0.529916
1	True	0.166667	True	foot	True	0.415633	0.390205
2	False	0.000000	True	hand	True	0.024814	0.611695
3	True	0.000000	True	skincare	False	0.679901	0.772656
4	True	0.833333	True	makeup	True	0.089330	1.000000
...	...	...	...	...	...	...	...
35984	False	0.166667	False	foot	True	0.071960	0.180658
35985	False	0.166667	False	fragrance	False	0.321340	0.960176
35987	False	0.833333	False	foot	True	0.177419	0.366834
35988	True	0.333333	False	makeup	True	0.107940	0.536627
35989	True	0.000000	False	makeup	True	0.151365	0.593249

30103 rows × 7 columns

In [156]:
```
dummy = pd.get_dummies(data['category'])
```

In [157]:
```
dummy
```

Out[157]:

	body	foot	fragrance	hair	hand	makeup	men_fragrance	men_skincare	nail	skincare	tools
0	False	False	False	False	False	False	False	False	True	False	False
1	False	True	False	False	False	False	False	False	False	False	False
2	False	False	False	False	True	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	True	False
4	False	False	False	False	False	True	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
35984	False	True	False	False	False	False	False	False	False	False	False
35985	False	False	True	False	False	False	False	False	False	False	False
35987	False	True	False	False	False	False	False	False	False	False	False
35988	False	False	False	False	False	True	False	False	False	False	False
35989	False	False	False	False	False	True	False	False	False	False	False

30103 rows × 11 columns

In [158]:
```
data = data.join(dummy)
```

In [159]:
```
data
```

Out[159]:

	clicked	number_of_clicks_before	ordered_before	category	on_sale	number_of_reviews	avg_review_score	body	foot	fragrance	hair	hand	ma
0	False	0.333333	True	nail	False	0.086849	0.529916	False	False	False	False	False	
1	True	0.166667	True	foot	True	0.415633	0.390205	False	True	False	False	False	
2	False	0.000000	True	hand	True	0.024814	0.611695	False	False	False	False	True	
3	True	0.000000	True	skincare	False	0.679901	0.772656	False	False	False	False	False	
4	True	0.833333	True	makeup	True	0.089330	1.000000	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	
35984	False	0.166667	False	foot	True	0.071960	0.180658	False	True	False	False	False	
35985	False	0.166667	False	fragrance	False	0.321340	0.960176	False	False	True	False	False	
35987	False	0.833333	False	foot	True	0.177419	0.366834	False	True	False	False	False	
35988	True	0.333333	False	makeup	True	0.107940	0.536627	False	False	False	False	False	
35989	True	0.000000	False	makeup	True	0.151365	0.593249	False	False	False	False	False	

30103 rows × 18 columns

In [160]:
```
data = data.replace({True:1, False:0})
```

In [161]:
```
move = data.pop("category")
move2 = data.pop("number_of_reviews")
move3 = data.pop("avg_review_score")
```

In [162]:
```
data.insert(1, "category", move)
data.insert(3, "number_of_reviews", move2)
data.insert(4, "avg_review_score", move3)
```

In [259]:
```
data
```

Out[259]:

	clicked	category	number_of_clicks_before	number_of_reviews	avg_review_score	ordered_before	on_sale	body	foot	fragrance	hair	hand	make
0	0	nail	0.333333	0.086849	0.529916	1	0	0	0	0	0	0	
1	1	foot	0.166667	0.415633	0.390205	1	1	0	1	0	0	0	
2	0	hand	0.000000	0.024814	0.611695	1	1	0	0	0	0	1	
3	1	skincare	0.000000	0.679901	0.772656	1	0	0	0	0	0	0	
4	1	makeup	0.833333	0.089330	1.000000	1	1	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
35984	0	foot	0.166667	0.071960	0.180658	0	1	0	1	0	0	0	
35985	0	fragrance	0.166667	0.321340	0.960176	0	0	0	0	1	0	0	
35987	0	foot	0.833333	0.177419	0.366834	0	1	0	1	0	0	0	
35988	1	makeup	0.333333	0.107940	0.536627	0	1	0	0	0	0	0	
35989	1	makeup	0.000000	0.151365	0.593249	0	1	0	0	0	0	0	

30103 rows × 18 columns

In [164]:
```
X = data.drop(['clicked', 'category'], axis = 1)
y = data.clicked
```

In [165]:
```
X
```

Out[165]:

	number_of_clicks_before	number_of_reviews	avg_review_score	ordered_before	on_sale	body	foot	fragrance	hair	hand	makeup	men_fragrance
0	0.333333	0.086849	0.529916	1	0	0	0	0	0	0	0	0
1	0.166667	0.415633	0.390205	1	1	0	1	0	0	0	0	0
2	0.000000	0.024814	0.611695	1	1	0	0	0	0	1	0	0
3	0.000000	0.679901	0.772656	1	0	0	0	0	0	0	0	0
4	0.833333	0.089330	1.000000	1	1	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
35984	0.166667	0.071960	0.180658	0	1	0	1	0	0	0	0	0
35985	0.166667	0.321340	0.960176	0	0	0	0	1	0	0	0	0
35987	0.833333	0.177419	0.366834	0	1	0	1	0	0	0	0	0
35988	0.333333	0.107940	0.536627	0	1	0	0	0	0	0	1	0
35989	0.000000	0.151365	0.593249	0	1	0	0	0	0	0	1	0

30103 rows × 16 columns

In [258]:
```
y
```

Out[258]:
```
0        0
1        1
2        0
3        1
4        1
       ..
35984    0
35985    0
35987    0
35988    1
35989    1
Name: clicked, Length: 30103, dtype: int64
```

In [166]:
```
data.xcolumns = ['number_of_clicks_before','number_of_reviews','avg_review_score','ordered_before', 'on_sale','bod
```

```
/var/folders/kp/5dkz8bcx6vx8fhrjwhg0gw0w0000gn/T/ipykernel_60543/969271643.py:1: UserWarning: Pandas doesn't allo
w columns to be created via a new attribute name - see https://pandas.pydata.org/pandas-docs/stable/indexing.html
#attribute-access (https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access)
  data.xcolumns = ['number_of_clicks_before','number_of_reviews','avg_review_score','ordered_before', 'on_sal
e','body','foot','fragrance','hair','hand','makeup','men_fragrance', 'men_skincare','nail','skincare','tools']
```

In [171]:
```
data.xcolumns
```

Out[171]:
```
['number_of_clicks_before',
 'number_of_reviews',
 'avg_review_score',
 'ordered_before',
 'on_sale',
 'body',
 'foot',
 'fragrance',
 'hair',
 'hand',
 'makeup',
 'men_fragrance',
 'men_skincare',
 'nail',
 'skincare',
 'tools']
```

In [172]:
```
from sklearn import tree
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [173]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 34)
```

In [200]:
```
Dtree = tree.DecisionTreeClassifier(max_depth = 10, min_samples_split = 5)
```

In [201]:
```
Dtree.fit(X_train, y_train)
```

Out[201]:
```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=10, min_samples_split=5)
```

In [277]:
```
pred_test = Dtree.predict(X_test)
pred_train = Dtree.predict(X_train)
```

In [276]:
```
accuracy_score(y_test, pred_test)
```

Out[276]:
```
0.7516332632045177
```

In [178]:
```
accuracy_score(y_test, y_pred, normalize = False)
```

Out[178]:
```
6392
```

In [278]:
```
metrics.accuracy_score(pred_test, y_test)
```

Out[278]:
```
0.7516332632045177
```

In [279]:
```
metrics.accuracy_score(pred_train, y_train)
```

Out[279]:
```
0.7824601366742597
```

In [280]:
```
DTreeTest_score = metrics.accuracy_score(pred_test, y_test)
DTeeTrain_score = metrics.accuracy_score(pred_train, y_train)
```

In [205]:
```
pd.DataFrame({'variable': data.xcolumns,
              'importance': Dtree.feature_importances_})
```

Out[205]:

	variable	importance
0	number_of_clicks_before	0.051080
1	number_of_reviews	0.792715
2	avg_review_score	0.056899
3	ordered_before	0.051061
4	on_sale	0.015309
5	body	0.003635
6	foot	0.002727
7	fragrance	0.003781
8	hair	0.001986
9	hand	0.004687
10	makeup	0.006423
11	men_fragrance	0.002658
12	men_skincare	0.002544
13	nail	0.002075
14	skincare	0.001660
15	tools	0.000760

In [214]:
```
dot_data = tree.export_graphviz(Dtree, out_file = None,
                                feature_names = X_train.columns)
```

In [216]:
```
Source(dot_data).render('tree')
```

Out[216]:
```
'tree.pdf'
```

In [179]:
```
lr = linear_model.LogisticRegression(max_iter = 100)
```

In [180]:
```
lr.fit(X_train, y_train)
```

Out[180]:
```
LogisticRegression
LogisticRegression()
```

In [181]:
```
lr.coef_
```

Out[181]:
```
array([[ 0.02906118, -1.16611891, -0.22604857,  0.64944778,  0.4469325 ,
        -0.03556149,  0.22888856, -0.0608526 , -0.0424    , -0.00151893,
         0.10362188,  0.20815086,  0.19390248, -0.2070896 , -0.20357749,
        -0.18396769]])
```

In [182]:
```
lr_train_pred = lr.predict(X_train)
lr_test_pred = lr.predict(X_test)
```

In [183]:
```
metrics.accuracy_score(y_train, lr_train_pred)
```

Out[183]:
```
0.6191628701594533
```

In [263]:
```
LR_train_score = metrics.accuracy_score(y_train, lr_train_pred)
LR_test_score = metrics.accuracy_score(y_test, lr_test_pred)
```

In [184]:
```
metrics.accuracy_score(y_test, lr_test_pred)
```

Out[184]:
```
0.6248477466504263
```

In [185]:
```
train_cm = metrics.confusion_matrix(y_train, lr_train_pred)
test_cm = metrics.confusion_matrix(y_test, lr_test_pred)
```

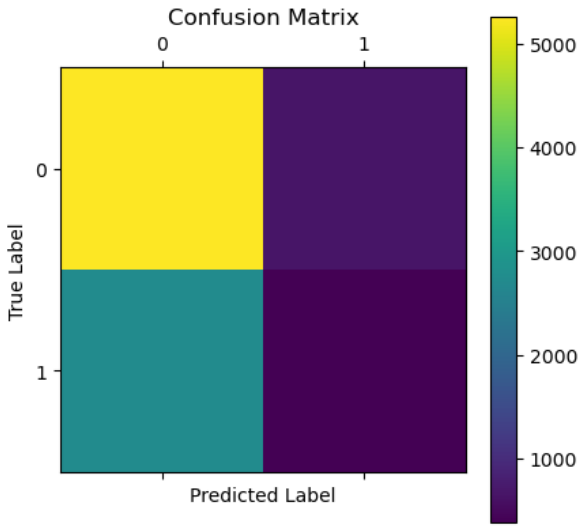
In [186]:
```
train_cm
```

Out[186]:
```
array([[12177,  1596],
       [ 6429,   870]])
```

In [260]:
```
plt.figure()
plt.matshow(test_cm)
plt.title('Confusion Matrix')
plt.colorbar()
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
```

Out[260]:
```
Text(0.5, 0, 'Predicted Label')
```

```
<Figure size 640x480 with 0 Axes>
```



In [187]:
```
RF = RandomForestClassifier (n_estimators = 100, random_state = 0)
```

In [188]:
```
RF.fit(X_train, y_train)
```

Out[188]:
```
RandomForestClassifier
RandomForestClassifier(random_state=0)
```

In [264]:
```
RF_pred_test = RF.predict(X_test)
RF_pred_train = RF.predict(X_train)
```

In [265]:
```
metrics.accuracy_score(RF_pred_test, y_test)
```

Out[265]:
```
0.7179714317351346
```

In [267]:
```
metrics.accuracy_score(RF_pred_train, y_train)
```

Out[267]:
```
0.8484719058466211
```

In [191]:
```
print(metrics.classification_report(y_test, RF_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.81      0.79      5914
           1       0.60      0.55      0.57      3117

    accuracy                           0.72      9031
   macro avg       0.69      0.68      0.68      9031
weighted avg       0.71      0.72      0.71      9031
```

In [268]:
```
RfTrainScore = metrics.accuracy_score(RF_pred_train, y_train)
RfTestScore = metrics.accuracy_score(RF_pred_test, y_test)
```

In [192]:
```
from sklearn.naive_bayes import GaussianNB
```

In [193]:
```
NB = GaussianNB()
```

In [194]:
```
NB.fit(X_train, y_train)
```

Out[194]:
```
GaussianNB
GaussianNB()
```

In [195]:
```
NB_train_pred = NB.predict(X_train)
NB_test_pred = NB.predict(X_test)
```

In [196]:
```
metrics.accuracy_score(y_train, NB_train_pred)
```

Out[196]:
```
0.6439350797266514
```

In [197]:
```
metrics.accuracy_score(y_test, NB_test_pred)
```

Out[197]:
```
0.6476580666592847
```

In [269]:
```
NB_TrainScore = metrics.accuracy_score(y_train, NB_train_pred)
NB_TestScore = metrics.accuracy_score(y_test, NB_test_pred)
```

In [198]:
```
np.set_printoptions(precision =2)
```

In [199]:
```
NB.predict_proba(X_test[:5])
```

Out[199]:
```
array([[0.99, 0.01],
       [0.97, 0.03],
       [0.41, 0.59],
       [0.28, 0.72],
       [0.18, 0.82]])
```

In [244]:
```
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import AdaBoostClassifier
```

In [253]:
```
abc = AdaBoostClassifier(n_estimators=50,
                         learning_rate =0.5)
```

In [254]:
```
model = abc.fit(X_train, y_train)
```

In [270]:
```
test_pred = model.predict(X_test)
train_pred = model.predict(X_train)
```

In [272]:
```
metrics.accuracy_score(y_test, test_pred)
```

Out[272]:
```
0.7619311261211383
```

In [273]:
```
metrics.accuracy_score(y_train, train_pred)
```

Out[273]:
```
0.7664673500379651
```

In [274]:
```
AdaTrainScore = metrics.accuracy_score(y_train, train_pred)
AdaTestScore = metrics.accuracy_score(y_test, test_pred)
```

In [282]:
```
AdaTrainScore
```

Out[282]:
```
0.7664673500379651
```

In [283]:
```
AdaTestScore
```

Out[283]:
```
0.7619311261211383
```

In [284]:
```
Accuracy = pd.read_csv('/Users/hunterberberich/Desktop/DAAN862/AccuracyScore.csv')
```

In [285]:
```
Accuracy
```

Out[285]:

	Model	Metrics Accuracy Score(Train)	Metrics Accuracy Score(Test)
0	DecisionTreeClassifier	0.75	0.78
1	LogisticRegression	0.62	0.62
2	RandomForestClassifier	0.72	0.85
3	NaïveBayes	0.64	0.65
4	AdaBoostClassifier	0.77	0.76

In [ ]:
```
#All models had decent performances once I figured out how to preprocess the data correctly (took me a very long t
#I learned that building a model from scratch with multiple data sets is extremely difficult. I also learned some
#I also gained knowledge of new tricks and functions I can use in the future I found in going through the textbook
#but seems could work well in some cases.
```