Customer Propensity Modeling

Wyatt Rasmussen

DSC 680

Final Project 1

Data Imports

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

In [2]:
```
training = pd.read_csv('data/training_sample.csv')
testing = pd.read_csv('data/testing_sample.csv')
```

Cleaning and Exploring Data

In [3]:
```
print('Size of training dataset:', training.shape)
print('Size of testing dataset:', testing.shape)
```

```
Size of training dataset: (455401, 25)
Size of testing dataset: (151655, 25)
```

In [4]:
```
print('Null count of training dataset:', sum(training.isna().sum()))
print('Null count of testing dataset:', sum(testing.isna().sum()))
```

```
Null count of training dataset: 0
Null count of testing dataset: 0
```

In [5]:
```
training.columns
```

Out[5]:
```
Index(['UserID', 'basket_icon_click', 'basket_add_list', 'basket_add_detail',
       'sort_by', 'image_picker', 'account_page_click', 'promo_banner_click',
       'detail_wishlist_add', 'list_size_dropdown', 'closed_minibasket_click',
       'checked_delivery_detail', 'checked_returns_detail', 'sign_in',
       'saw_checkout', 'saw_sizecharts', 'saw_delivery', 'saw_account_upgrade',
       'saw_homepage', 'device_mobile', 'device_computer', 'device_tablet',
       'returning_user', 'loc_uk', 'ordered'],
      dtype='object')
```

In [6]:
```
testing.columns
```

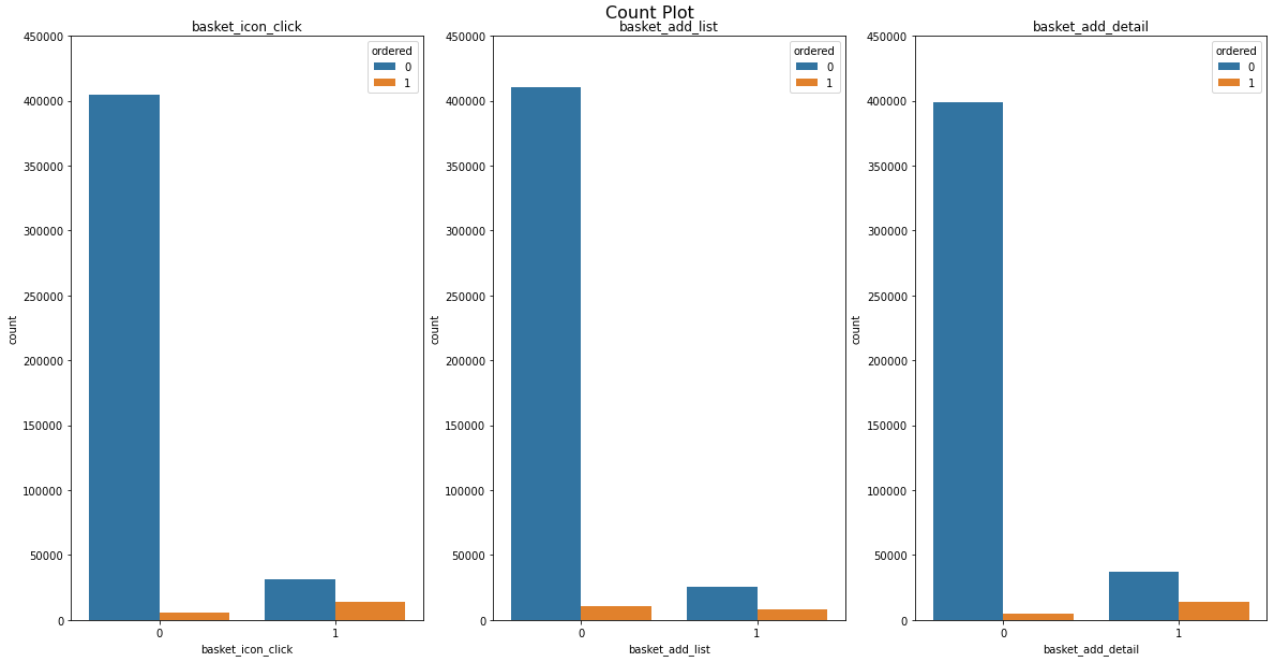
Out[6]:
```
Index(['UserID', 'basket_icon_click', 'basket_add_list', 'basket_add_detail',
       'sort_by', 'image_picker', 'account_page_click', 'promo_banner_click',
```

```
       'detail_wishlist_add', 'list_size_dropdown', 'closed_minibasket_click',
       'checked_delivery_detail', 'checked_returns_detail', 'sign_in',
       'saw_checkout', 'saw_sizecharts', 'saw_delivery', 'saw_account_upgrade',
       'saw_homepage', 'device_mobile', 'device_computer', 'device_tablet',
       'returning_user', 'loc_uk', 'ordered'],
      dtype='object')
```

In [7]:
```
training.dtypes
```

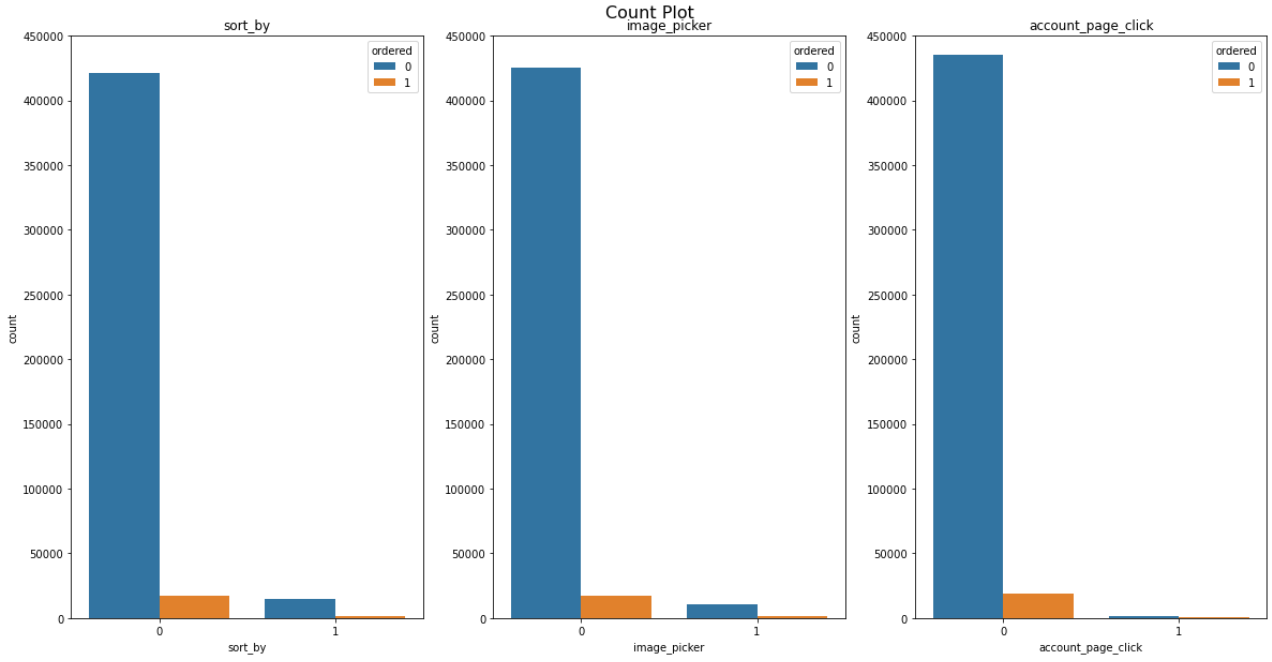
Out[7]:
```
UserID                    object
basket_icon_click          int64
basket_add_list            int64
basket_add_detail          int64
sort_by                    int64
image_picker               int64
account_page_click         int64
promo_banner_click         int64
detail_wishlist_add        int64
list_size_dropdown         int64
closed_minibasket_click    int64
checked_delivery_detail    int64
checked_returns_detail     int64
sign_in                    int64
saw_checkout               int64
saw_sizecharts             int64
saw_delivery               int64
saw_account_upgrade        int64
saw_homepage               int64
device_mobile              int64
device_computer            int64
device_tablet              int64
returning_user             int64
loc_uk                     int64
ordered                    int64
dtype: object
```

In [8]:
```
## for multiple columns
fig, ax = plt.subplots(1, 3, figsize=(20, 10))
fig.suptitle('Count Plot', fontsize=16, y=0.92)

columns = ['basket_icon_click', 'basket_add_list', 'basket_add_detail']
for i, col in enumerate(columns):
    graph = sns.countplot(x=training[col], hue=training["ordered"], ax=ax[i])
    graph.set_ylim(0,450000)
    ax[i].set_title(*[col])
```



In [9]:

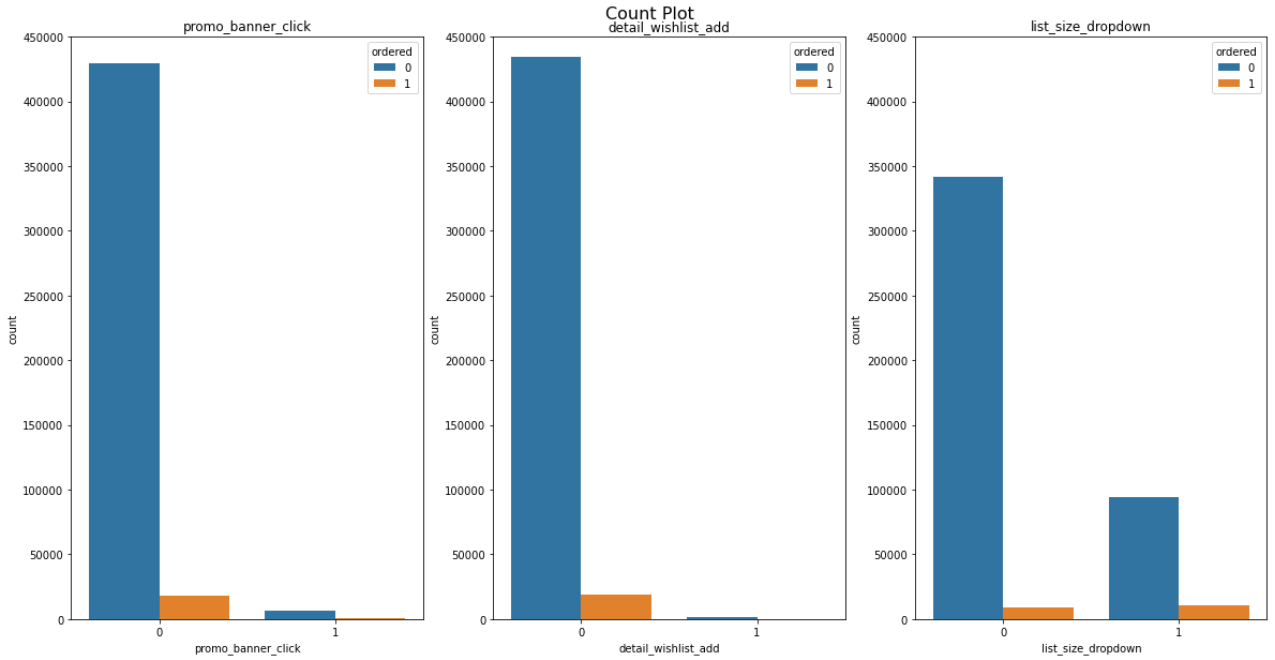
```
## for multiple columns
fig, ax = plt.subplots(1, 3, figsize=(20, 10))
fig.suptitle('Count Plot', fontsize=16, y=0.92)

columns = ['sort_by', 'image_picker', 'account_page_click']
for i, col in enumerate(columns):
    graph = sns.countplot(x=training[col], hue=training["ordered"], ax=ax[i])
    graph.set_ylim(0,450000)
    ax[i].set_title(*[col])
```



In [10]:

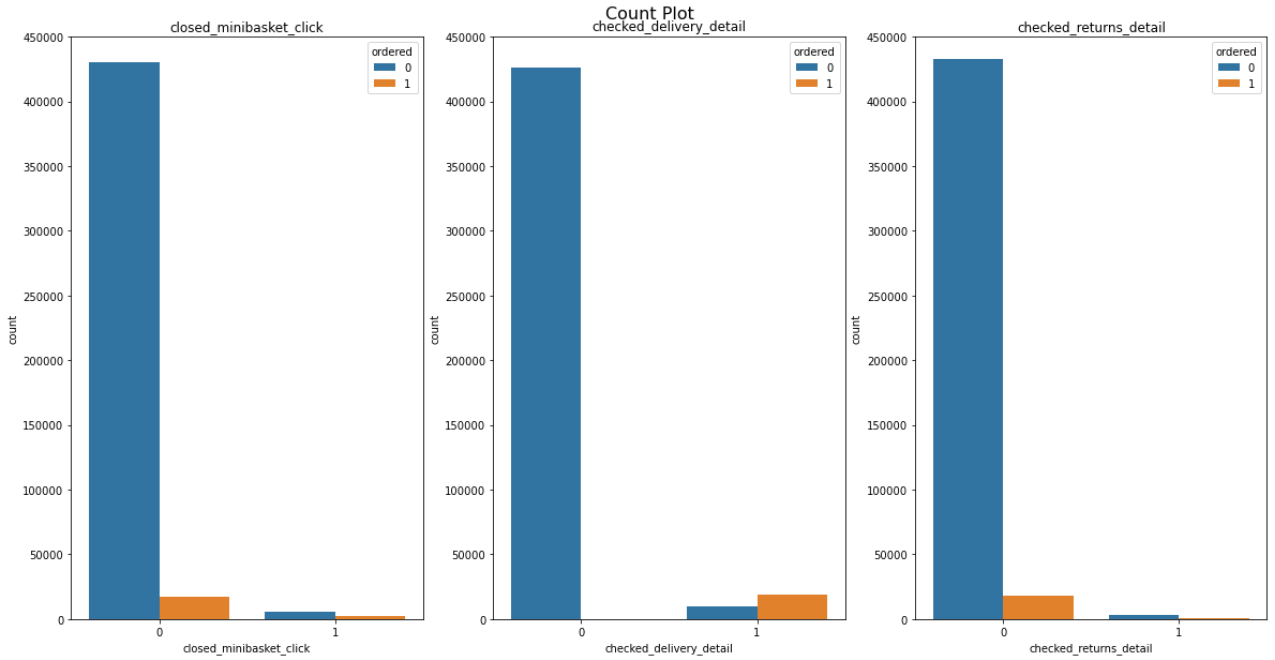
```
## for multiple columns
fig, ax = plt.subplots(1, 3, figsize=(20, 10))
fig.suptitle('Count Plot', fontsize=16, y=0.92)

columns = ['promo_banner_click', 'detail_wishlist_add', 'list_size_dropdown']
for i, col in enumerate(columns):
```

```
    graph = sns.countplot(x=training[col], hue=training["ordered"], ax=ax[i])
    graph.set_ylim(0,450000)
    ax[i].set_title(*[col])
```



In [11]:

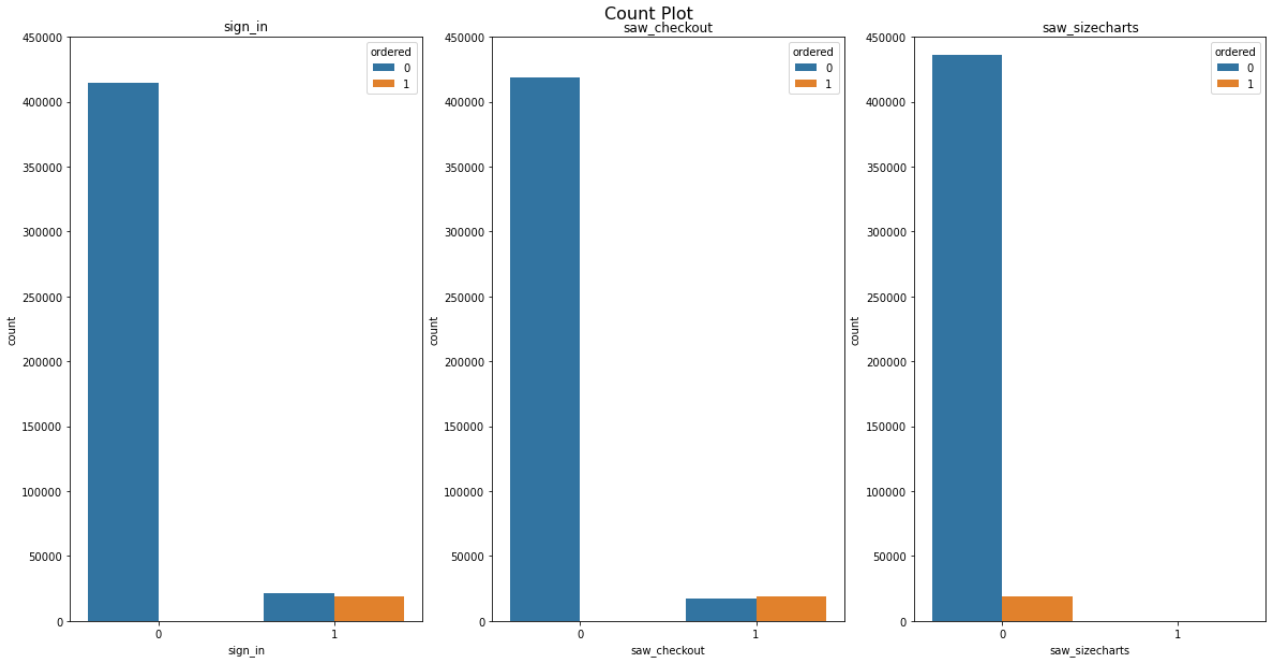
```
## for multiple columns
fig, ax = plt.subplots(1, 3, figsize=(20, 10))
fig.suptitle('Count Plot', fontsize=16, y=0.92)

columns = ['closed_minibasket_click', 'checked_delivery_detail', 'checked_return
for i, col in enumerate(columns):
    graph = sns.countplot(x=training[col], hue=training["ordered"], ax=ax[i])
    graph.set_ylim(0,450000)
    ax[i].set_title(*[col])
```



In [12]:

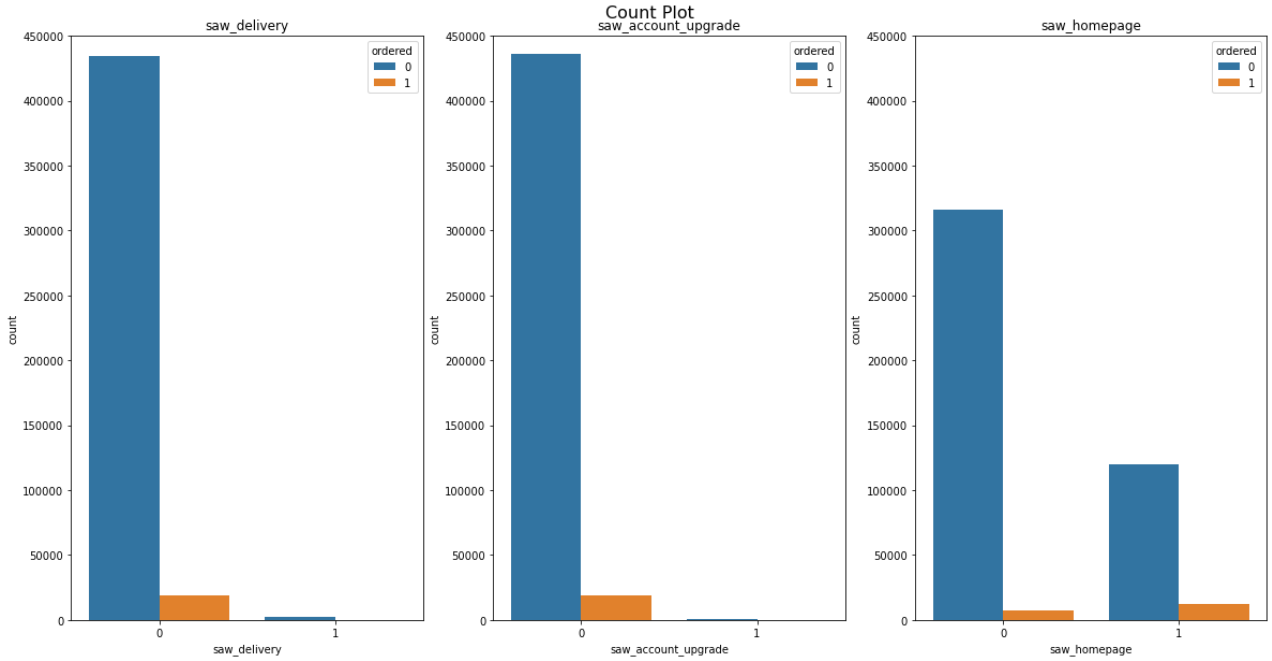
```
## for multiple columns
```

```
fig, ax = plt.subplots(1, 3, figsize=(20, 10))
fig.suptitle('Count Plot', fontsize=16, y=0.92)

columns = ['sign_in', 'saw_checkout', 'saw_sizecharts']
for i, col in enumerate(columns):
    graph = sns.countplot(x=training[col], hue=training["ordered"], ax=ax[i])
    graph.set_ylim(0,450000)
    ax[i].set_title(*[col])
```



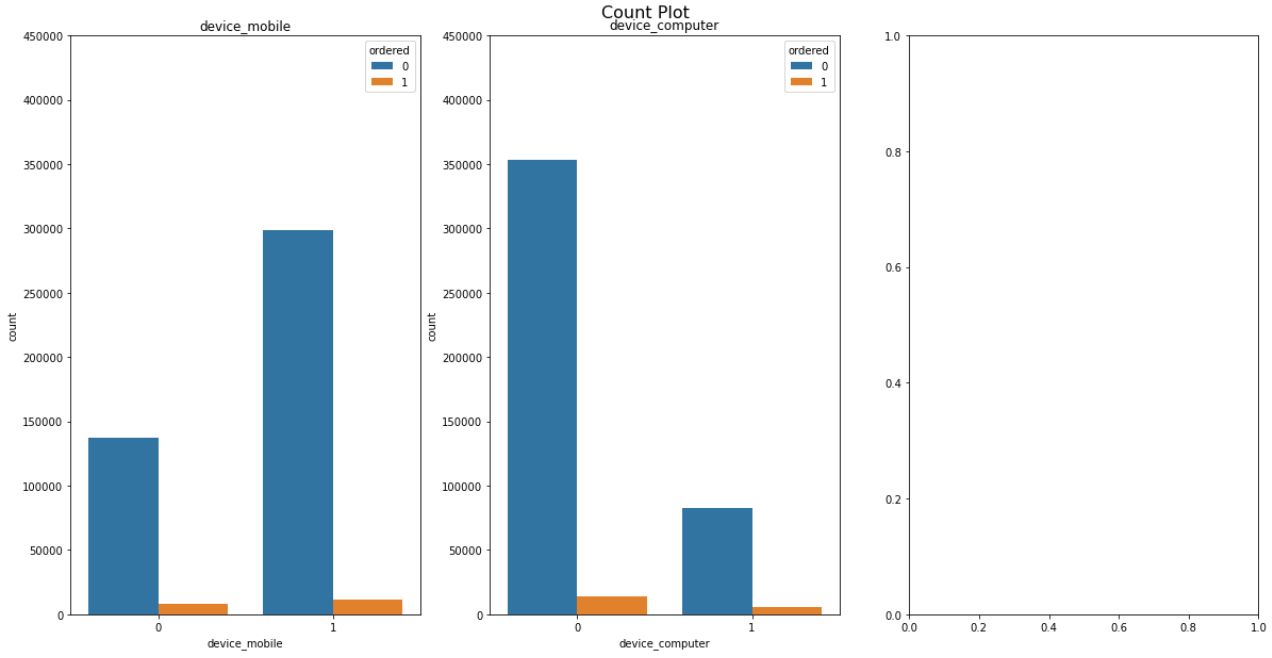
In [13]:

```
## for multiple columns
fig, ax = plt.subplots(1, 3, figsize=(20, 10))
fig.suptitle('Count Plot', fontsize=16, y=0.92)

columns = ['saw_delivery', 'saw_account_upgrade', 'saw_homepage']
for i, col in enumerate(columns):
    graph = sns.countplot(x=training[col], hue=training["ordered"], ax=ax[i])
    graph.set_ylim(0,450000)
    ax[i].set_title(*[col])
```



In [14]:

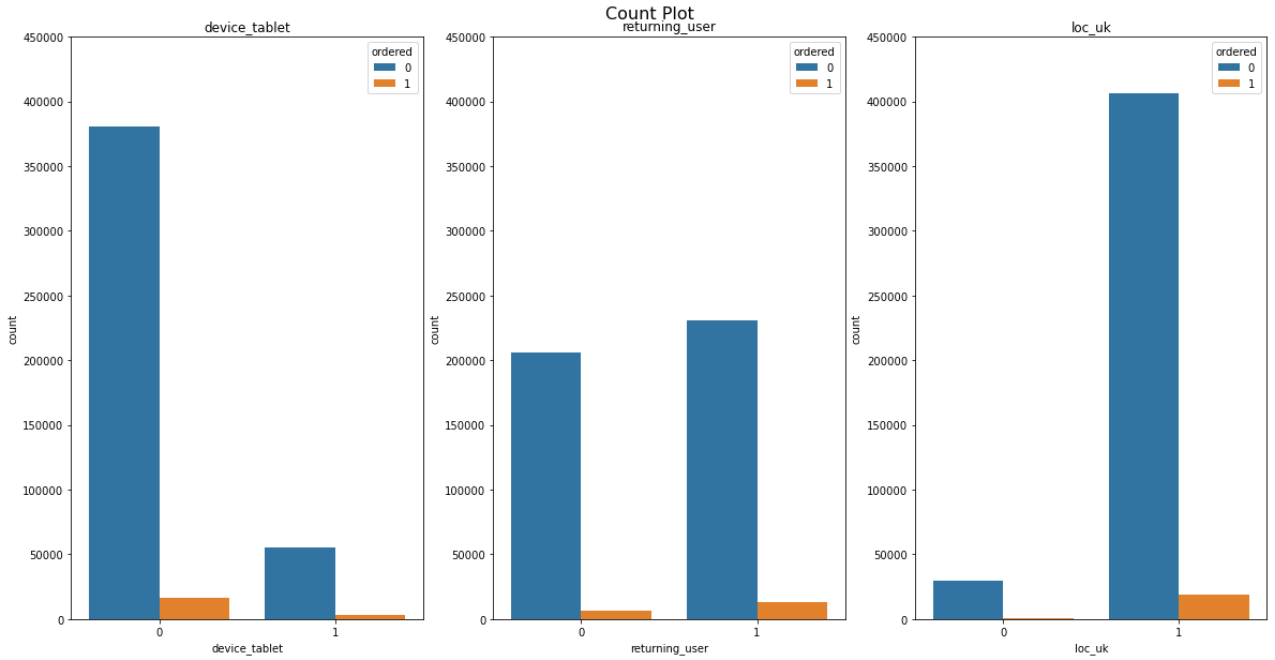
```
## for multiple columns
fig, ax = plt.subplots(1, 3, figsize=(20, 10))
fig.suptitle('Count Plot', fontsize=16, y=0.92)

columns = ['device_mobile', 'device_computer']
for i, col in enumerate(columns):
    graph = sns.countplot(x=training[col], hue=training["ordered"], ax=ax[i])
    graph.set_ylim(0,450000)
    ax[i].set_title(*[col])
```



In [15]:

```
## for multiple columns
fig, ax = plt.subplots(1, 3, figsize=(20, 10))
fig.suptitle('Count Plot', fontsize=16, y=0.92)

columns = ['device_tablet', 'returning_user', 'loc_uk']
for i, col in enumerate(columns):
```

```
    graph = sns.countplot(x=training[col], hue=training["ordered"], ax=ax[i])
    graph.set_ylim(0,450000)
    ax[i].set_title(*[col])
```



In [16]:

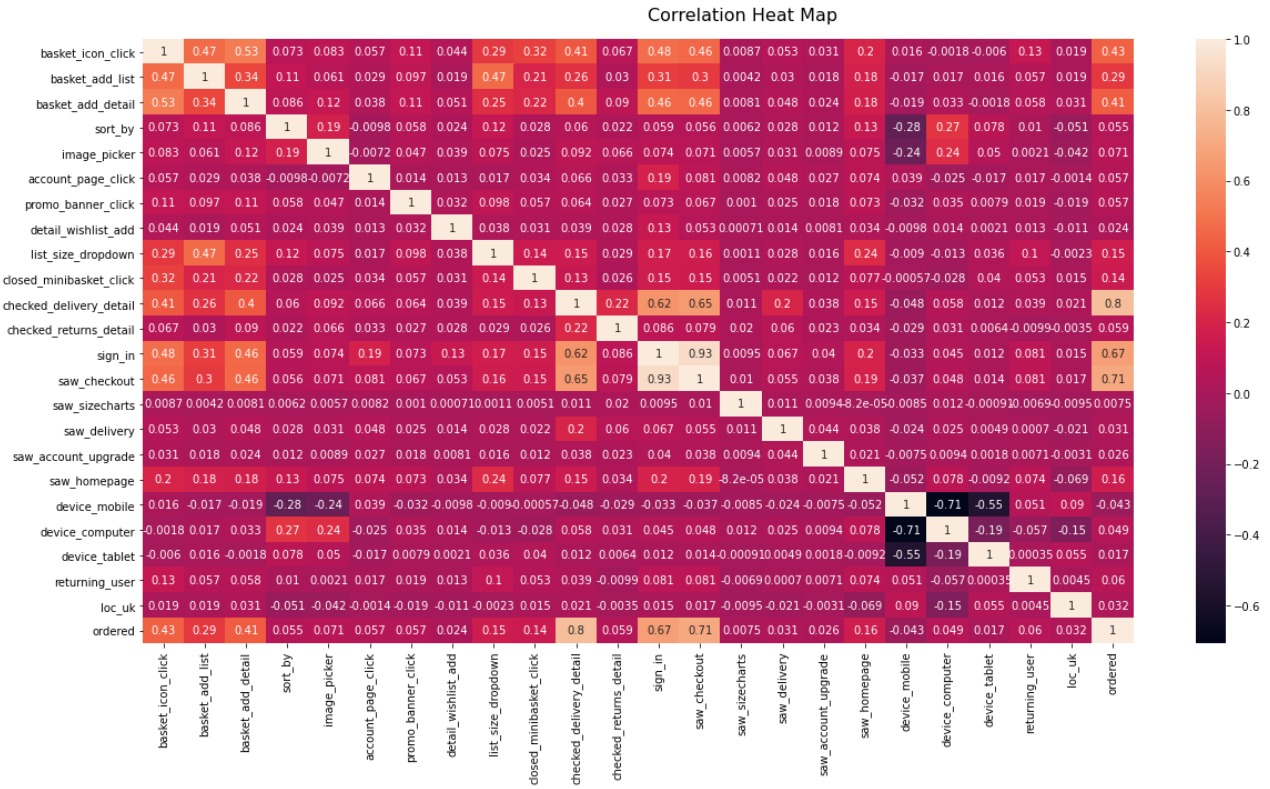
```
corr = training.corr()

corr.style.background_gradient(cmap='coolwarm')
```

Out[16]:

	basket_icon_click	basket_add_list	basket_add_detail	sort_by	image_
basket_icon_click	1.000000	0.466671	0.529947	0.073016	0.(
basket_add_list	0.466671	1.000000	0.340968	0.106852	0.
basket_add_detail	0.529947	0.340968	1.000000	0.085854	0.
sort_by	0.073016	0.106852	0.085854	1.000000	0
image_picker	0.082893	0.061462	0.124230	0.185661	1.(
account_page_click	0.057253	0.028994	0.037502	-0.009754	-0.
promo_banner_click	0.109342	0.096608	0.109043	0.058155	0.(
detail_wishlist_add	0.044153	0.019061	0.050724	0.024056	0.
list_size_dropdown	0.291608	0.469625	0.247205	0.124273	0.
closed_minibasket_click	0.323940	0.208082	0.222444	0.028453	0.
checked_delivery_detail	0.405787	0.264766	0.404134	0.059635	0
checked_returns_detail	0.067149	0.030469	0.090434	0.022364	0.(
sign_in	0.478834	0.312276	0.461659	0.058662	0.
saw_checkout	0.458774	0.297681	0.456713	0.055959	(
saw_sizecharts	0.008741	0.004161	0.008101	0.006196	0.
saw_delivery	0.052922	0.030286	0.048410	0.028102	0.

	basket_icon_click	basket_add_list	basket_add_detail	sort_by	image_
saw_account_upgrade	0.030764	0.018150	0.024255	0.012194	0.
saw_homepage	0.203087	0.180221	0.175138	0.128205	0.
device_mobile	0.016203	-0.017202	-0.018800	-0.278043	-0.
device_computer	-0.001757	0.016629	0.032794	0.269589	0
device_tablet	-0.006019	0.015516	-0.001799	0.078088	0.
returning_user	0.126640	0.057443	0.057680	0.010366	0.
loc_uk	0.018518	0.018797	0.030956	-0.051148	-0.
ordered	0.428334	0.287666	0.414420	0.054636	0.

In [17]:
```
fig, ax = plt.subplots(1, 1, figsize=(20, 10))
fig.suptitle('Correlation Heat Map', fontsize=16, y=0.92)

sns.heatmap(training.corr(), annot = True)
plt.show()
```



In [18]:
```
training.corr()['ordered']
```

Out[18]:
```
basket_icon_click          0.428334
basket_add_list            0.287666
basket_add_detail          0.414420
sort_by                    0.054636
image_picker               0.071492
account_page_click         0.057279
promo_banner_click         0.056533
detail_wishlist_add        0.023516
list_size_dropdown         0.154867
```

```
closed_minibasket_click    0.140011
checked_delivery_detail    0.798720
checked_returns_detail     0.059484
sign_in                    0.665556
saw_checkout               0.708986
saw_sizecharts             0.007548
saw_delivery               0.031461
saw_account_upgrade        0.025857
saw_homepage               0.157778
device_mobile             -0.042907
device_computer            0.049208
device_tablet              0.016939
returning_user             0.060295
loc_uk                     0.031643
ordered                    1.000000
Name: ordered, dtype: float64
```

In [19]:
```
training.corr()['ordered'] > 0.15
```

Out[19]:
```
basket_icon_click           True
basket_add_list             True
basket_add_detail           True
sort_by                    False
image_picker               False
account_page_click         False
promo_banner_click         False
detail_wishlist_add        False
list_size_dropdown          True
closed_minibasket_click    False
checked_delivery_detail     True
checked_returns_detail     False
sign_in                     True
saw_checkout                True
saw_sizecharts             False
saw_delivery               False
saw_account_upgrade        False
saw_homepage                True
device_mobile              False
device_computer            False
device_tablet              False
returning_user             False
loc_uk                     False
ordered                     True
Name: ordered, dtype: bool
```

In [20]:
```
training.corr()['ordered'] > 0.02
```

Out[20]:
```
basket_icon_click           True
basket_add_list             True
basket_add_detail           True
sort_by                     True
image_picker                True
account_page_click          True
promo_banner_click          True
detail_wishlist_add         True
list_size_dropdown          True
closed_minibasket_click     True
checked_delivery_detail     True
checked_returns_detail      True
sign_in                     True
saw_checkout                True
saw_sizecharts             False
```

```
saw_delivery                True
saw_account_upgrade         True
saw_homepage                True
device_mobile              False
device_computer             True
device_tablet              False
returning_user              True
loc_uk                      True
ordered                     True
Name: ordered, dtype: bool
```

Feature Selection and Separating Predictors from Target Variable

Methods

For the feature selection I would like to try 2 different methods. 1st I would like to take the variables over 0.15 correlation with ordered which would be 8 features. 2nd I would like to take the variables that are over 0.02, which would be 20 features. This will change and influence the number of features that are being used in the model building portion. This could help us limit the total number of features used or it could prove that the more features the better the result.

In [21]:
```
predictors15 = training[['basket_icon_click', 'basket_add_list', 'basket_add_det
                         'checked_delivery_detail', 'sign_in', 'saw_checkout', 's
```

In [22]:
```
predictors02 = training.drop(['saw_sizecharts', 'device_mobile', 'device_tablet'
```

In [23]:
```
predictors15.head()
```

Out[23]:

	basket_icon_click	basket_add_list	basket_add_detail	list_size_dropdown	checked_delivery_d
0	0	0	0	0	
1	0	0	0	0	
2	0	0	0	0	
3	0	0	0	0	
4	0	1	0	1	

In [24]:
```
predictors02.head()
```

Out[24]:

	basket_icon_click	basket_add_list	basket_add_detail	sort_by	image_picker	account_page_c
0	0	0	0	0	0	
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	
4	0	1	0	1	0	

In [25]:
```
target = training['ordered']
```

In [26]:
```
X_train15, X_test15, y_train15, y_test15 = train_test_split(predictors15, target

print( "Predictor - Training : ", X_train15.shape, "Predictor - Testing : ", X_t
```

```
Predictor - Training :  (341550, 8) Predictor - Testing :  (113851, 8)
```

In [27]:
```
X_train02, X_test02, y_train02, y_test02 = train_test_split(predictors02, target

print( "Predictor - Training : ", X_train02.shape, "Predictor - Testing : ", X_t
```

```
Predictor - Training :  (341550, 20) Predictor - Testing :  (113851, 20)
```

Building a Predictions Model

In [28]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

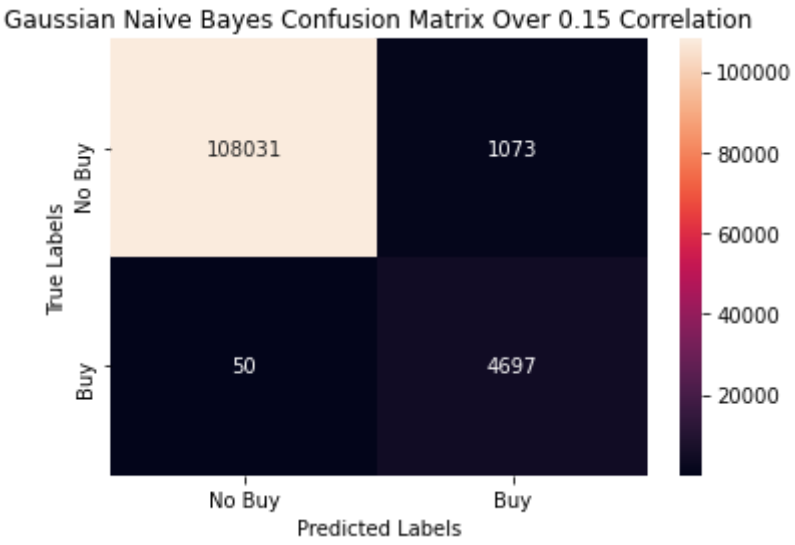
In [29]:
```
classifier = GaussianNB()
classifier = classifier.fit(X_train15, y_train15)
```

In [30]:
```
predictions15 = classifier.predict(X_test15)
```

In [31]:
```
confusion_matrix(y_test15, predictions15)
```

Out[31]:
```
array([[108031,   1073],
       [    50,   4697]])
```

In [32]:
```
cm=confusion_matrix(y_test15, predictions15)

ax = plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax)

ax.set_xlabel('Predicted Labels');ax.set_ylabel('True Labels');
ax.set_title('Gaussian Naive Bayes Confusion Matrix Over 0.15 Correlation');
ax.xaxis.set_ticklabels(['No Buy', 'Buy']);ax.yaxis.set_ticklabels(['No Buy', 'B

plt.show()
```



In [33]:
```
accuracy_score(y_test15, predictions15)
```

Out[33]:
```
0.9901362306874775
```

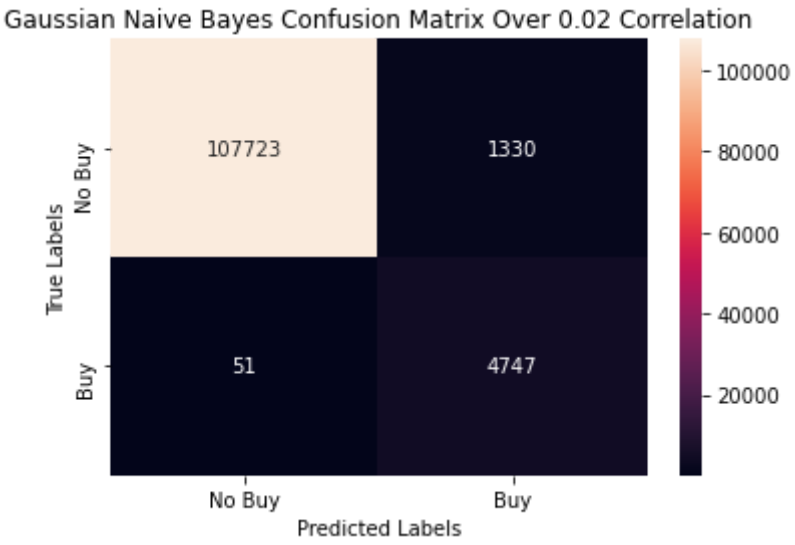
In [34]:
```
classifier = GaussianNB()
classifier = classifier.fit(X_train02, y_train02)
```

In [35]:
```
predictions02 = classifier.predict(X_test02)
```

In [36]:
```
confusion_matrix(y_test02, predictions02)
```

Out[36]:
```
array([[107723,   1330],
       [    51,   4747]])
```

In [37]:
```
cm=confusion_matrix(y_test02, predictions02)

ax = plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax)

ax.set_xlabel('Predicted Labels');ax.set_ylabel('True Labels');
ax.set_title('Gaussian Naive Bayes Confusion Matrix Over 0.02 Correlation');
ax.xaxis.set_ticklabels(['No Buy', 'Buy']);ax.yaxis.set_ticklabels(['No Buy', 'B

plt.show()
```



In [38]:
```
accuracy_score(y_test02, predictions02)
```

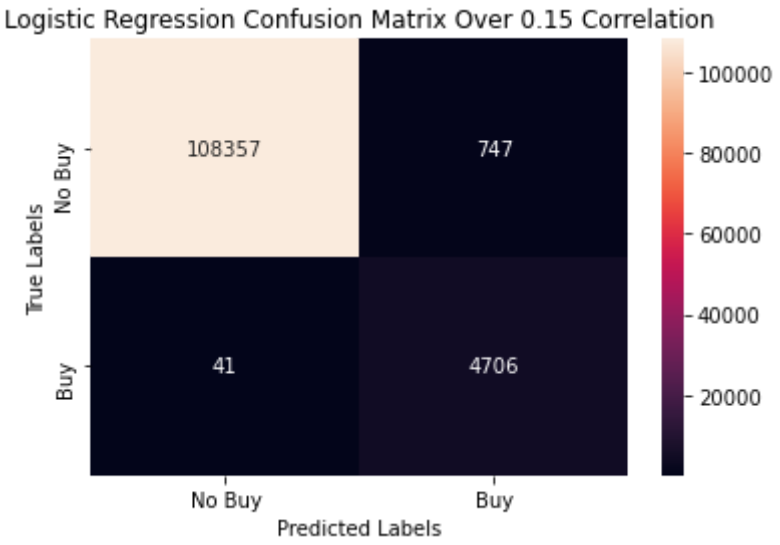
Out[38]: 0.987870110934467

In [39]:
```
log = LogisticRegression()
```

In [40]:
```
log = log.fit(X_train15, y_train15)
```

In [41]:
```
logPredict15 = log.predict(X_test15)
```

In [50]:
```
cm=confusion_matrix(y_test15, logPredict15)

ax = plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax)

ax.set_xlabel('Predicted Labels');ax.set_ylabel('True Labels');
ax.set_title('Logistic Regression Confusion Matrix Over 0.15 Correlation');
ax.xaxis.set_ticklabels(['No Buy', 'Buy']);ax.yaxis.set_ticklabels(['No Buy', 'B

plt.show()
```



In [43]:
```
confusion_matrix(y_test15, logPredict15)
```

Out[43]:
```
array([[108357,    747],
       [    41,   4706]])
```

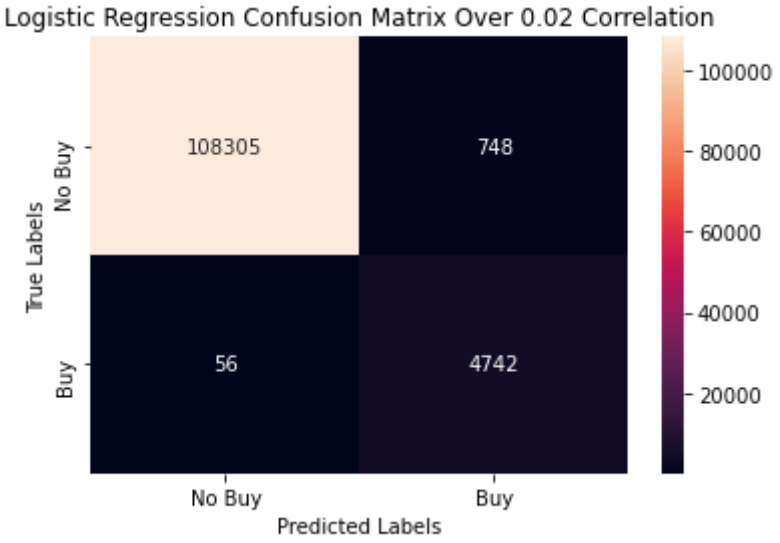
In [44]:
```
accuracy_score(y_test15, logPredict15)
```

Out[44]:
```
0.993078673002433
```

In [45]:
```
log = LogisticRegression()
log = log.fit(X_test02, y_test02)
```

In [46]:
```
logPredict02 = log.predict(X_test02)
```

In [47]:
```
cm=confusion_matrix(y_test02, logPredict02)

ax = plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax)

ax.set_xlabel('Predicted Labels');ax.set_ylabel('True Labels');
ax.set_title('Logistic Regression Confusion Matrix Over 0.02 Correlation');
ax.xaxis.set_ticklabels(['No Buy', 'Buy']);ax.yaxis.set_ticklabels(['No Buy', 'B

plt.show()
```



In [48]:
```
confusion_matrix(y_test02, logPredict02)
```

Out[48]:
```
array([[108305,    748],
       [    56,   4742]])
```

In [49]:
```
accuracy_score(y_test02, logPredict02)
```

Out[49]:
```
0.9929381384441067
```

In [ ]:
```

```