CUSTOMER CART ABANDONMENT

Shilpi Gupta A001

Chelsi Fagniya A008

INTRODUCTION:

Customer cart Abandonment can be referred as an e-commerce term to describe a visitor who leaves the web page before completing the purchase. Shopping cart abandonment is one of the most crucial problems for online businesses to overcome. Examples of abandonment include shopping cart abandonment, referring to visitors who add items to their online shopping cart, but exit without completing the purchase.

DATASET:

We have used Python as our primary software for the analysis part; here our first step is to import the dataset. After importing the dataset we have explored features of our dataset.

```
dataset.shape
(4284, 13)
```

The dataset used contains 4284 rows and 13 columns.
The dataset consists of following 13 attributes:

S.No.	Name	Description
1	ID	The session id of the customer
2	Is_Product_Details_Viewed	Whether the customer is viewing the product details or not
3	Session_Activity_Count	How many times a customer is going to the different pages.
4	No_Items_Added_InCart	Number of items in cart
5	No_Items_Removed_FromCart	Number of items removed from the cart
6	No_Cart_Viewed	How many times the customer is going to the cart page.
7	No_Checkout_Confirmed	How many times the checkout has been confirmed successfully by the customer.
8	No_Checkout_Initiated	How many times the checkout(successful as well as unsuccess) is being done by the user
9	No_Cart_Items_Viewed	How many times a user is viewing the product from cart
10	No_Customer_Login	Number of times the customer had did log in
11	No_Page_Viewed	Number of pages viewed by the customer
12	Customer_Segment_Type	The customer falls under which category,i.e, 0 for Target Customer, 1 for Loyal Customer, and 2 for Untargeted customer
13	Cart_Abandoned	Whether the customer is doing cart abandonment or not. This is the target variable that we need to predict

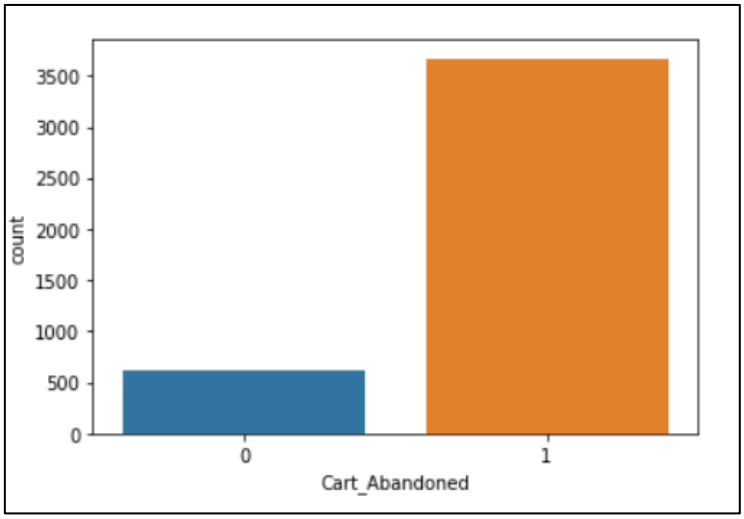
EXPLORATORY DATA ANALYSIS

Here we have performed both univariate as well as bivariate analysis to infer sights from our dataset. We have divided all numerical as well as categorical variables.

```
num=dataset.select_dtypes(include=["float64","int64"])
cat=dataset.select_dtypes(include=["object","category"]).drop(["ID"],axis=1)
```

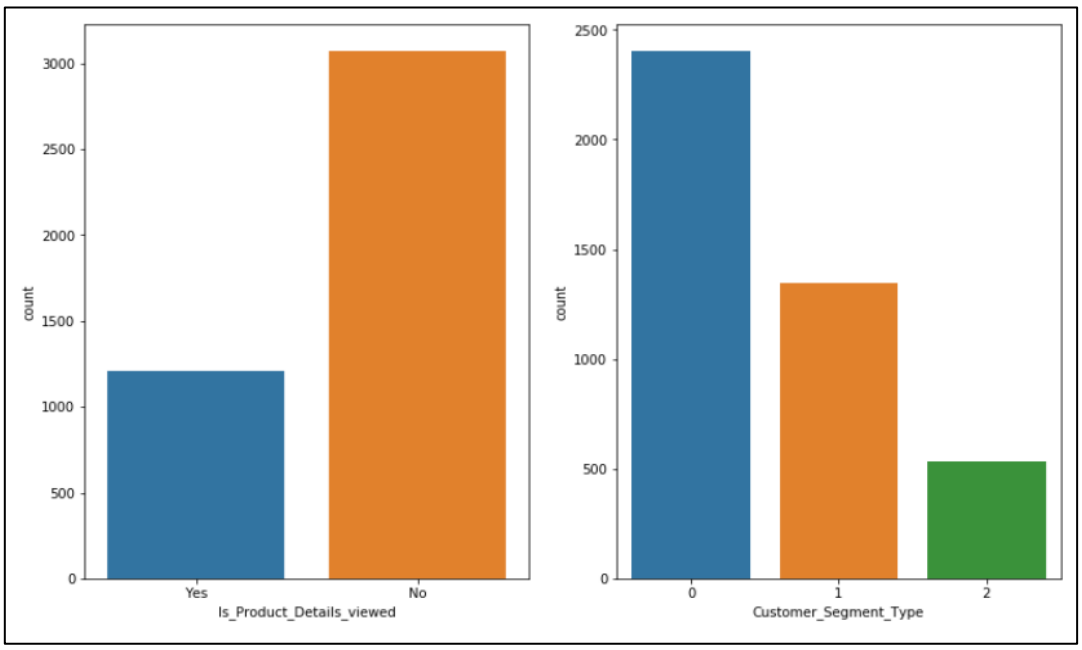
UNIVARIATE ANALYSIS

- Target variable – Cart Abandoned



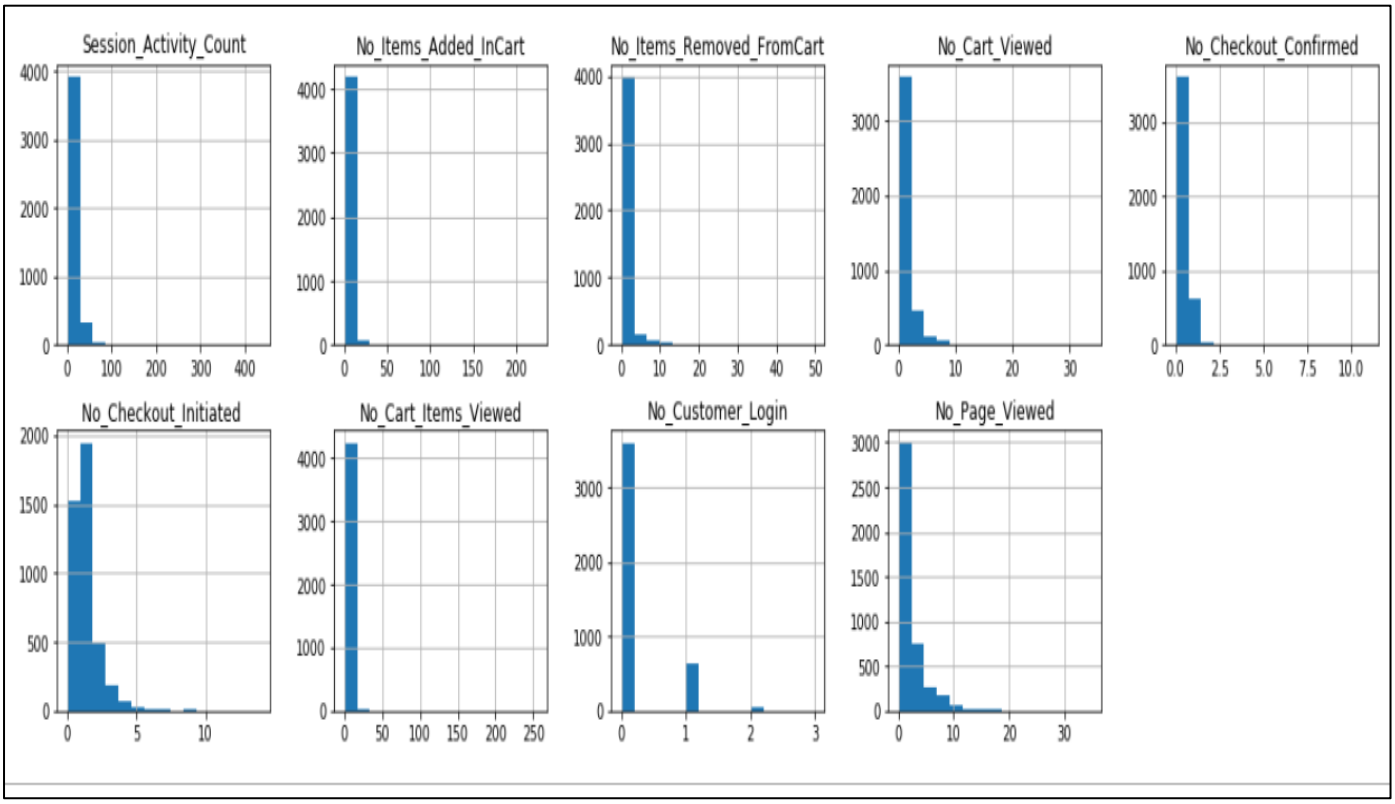
It can be clearly seen that the data is an example of imbalanced dataset.

- Count plot for categorical attributes – Is_Product_Details_viewed & Customer_Segment_Type



Here we can observe that there are very few sessions in which the product details have been viewed by the user. And we can also observe that our targeted customers (0) are very high compared to untargeted (2) customers.

- Histogram for numerical attributes -

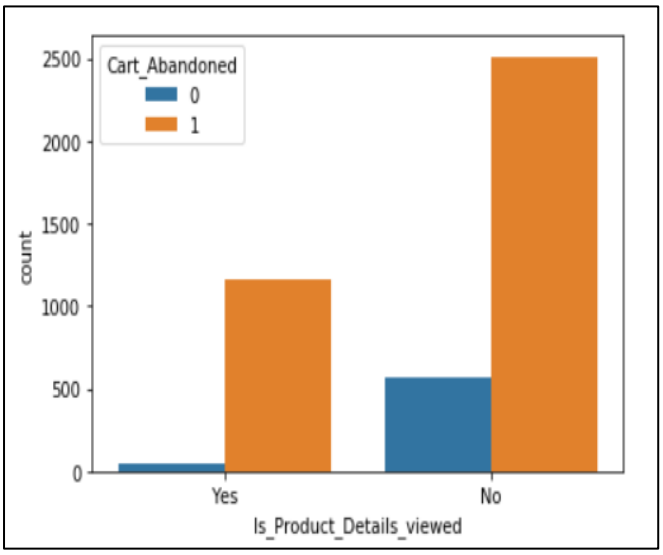


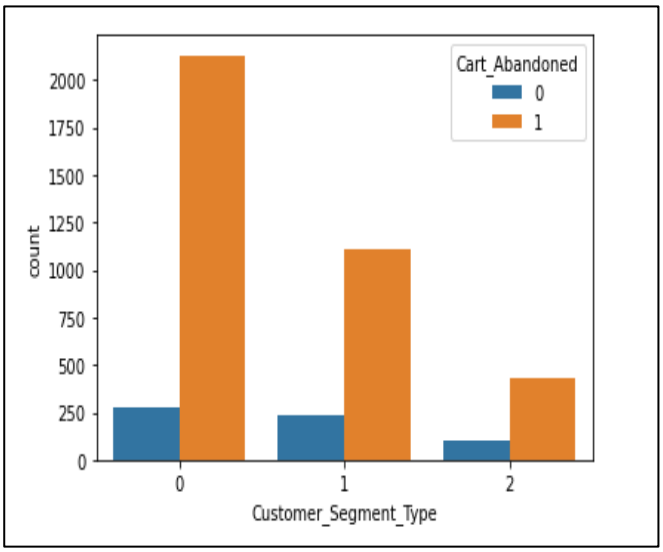
Here we can see that our data unevenly spread.

BIVARIATE ANALYSIS

- FOR CATEGORICAL
  1. Is_Product_Details_viewed Vs. Cart_Abandoned
  2. Customer_Segment_Type Vs. Cart_Abandoned

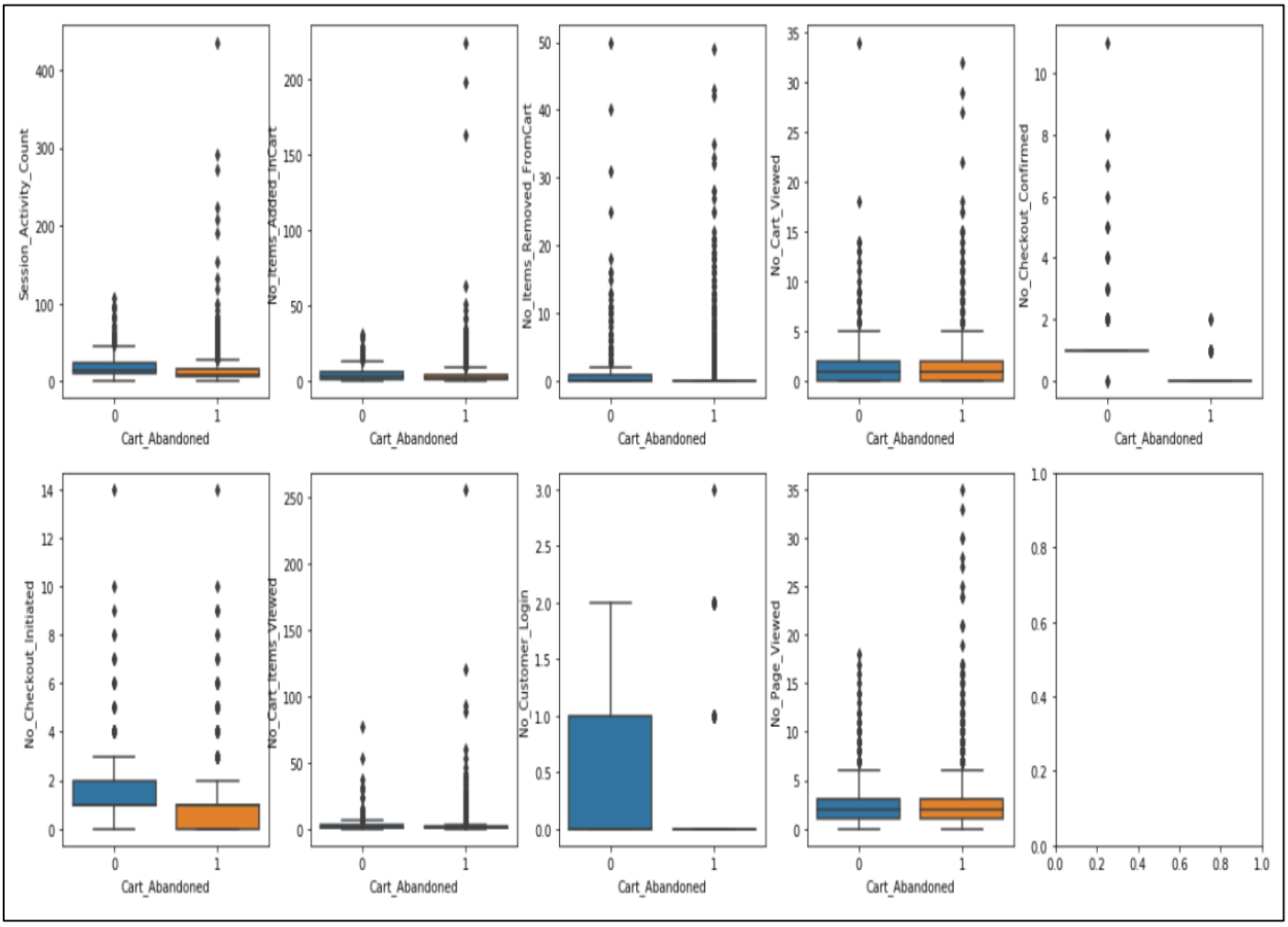
If a customer is viewing the product details then what is the chance that he is doing cart abandonment? How does customer segment type (0 for target customer, 1 for loyal customer, and 2 for untargeted customer) affect cart abandonment?





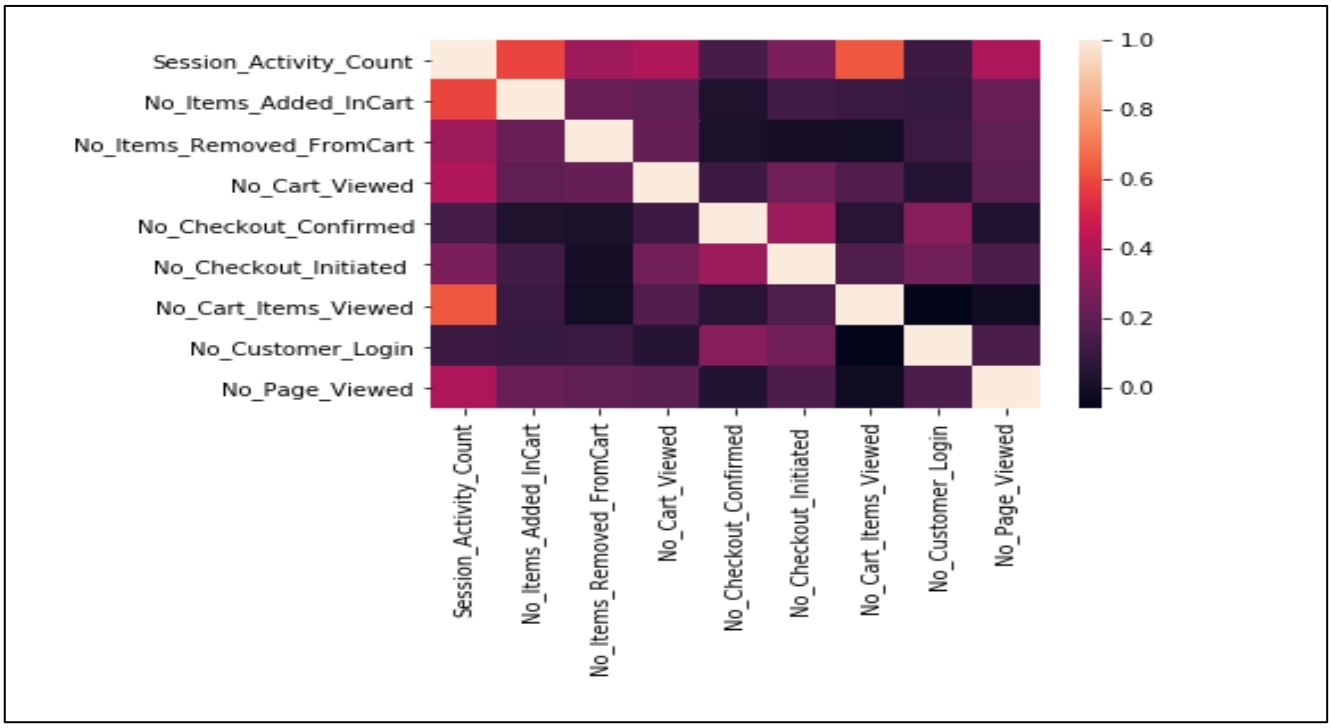
Here we can see that cart abandonment is seen more when the product details are not viewed.

- FOR NUMERICAL
  1. Numerical Attributes Vs. Cart_Abandoned: By Boxplot



Here we can see that we have many outliers

  2. Correlation plot of independent attributes



Here we can observe that our attributes are mostly independent of each other.

DATA PRE-PROCESSING

- Checking for missing values

```
data = dataset.copy()
data.isna().sum()
```

```
ID                              0
Is_Product_Details_viewed       0
Session_Activity_Count          0
No_Items_Added_InCart           9
No_Items_Removed_FromCart       0
No_Cart_Viewed                  9
No_Checkout_Confirmed           0
No_Checkout_Initiated           0
No_Cart_Items_Viewed            0
No_Customer_Login               0
No_Page_Viewed                  0
Customer_Segment_Type           0
Cart_Abandoned                  0
dtype: int64
```

- Imputing missing values

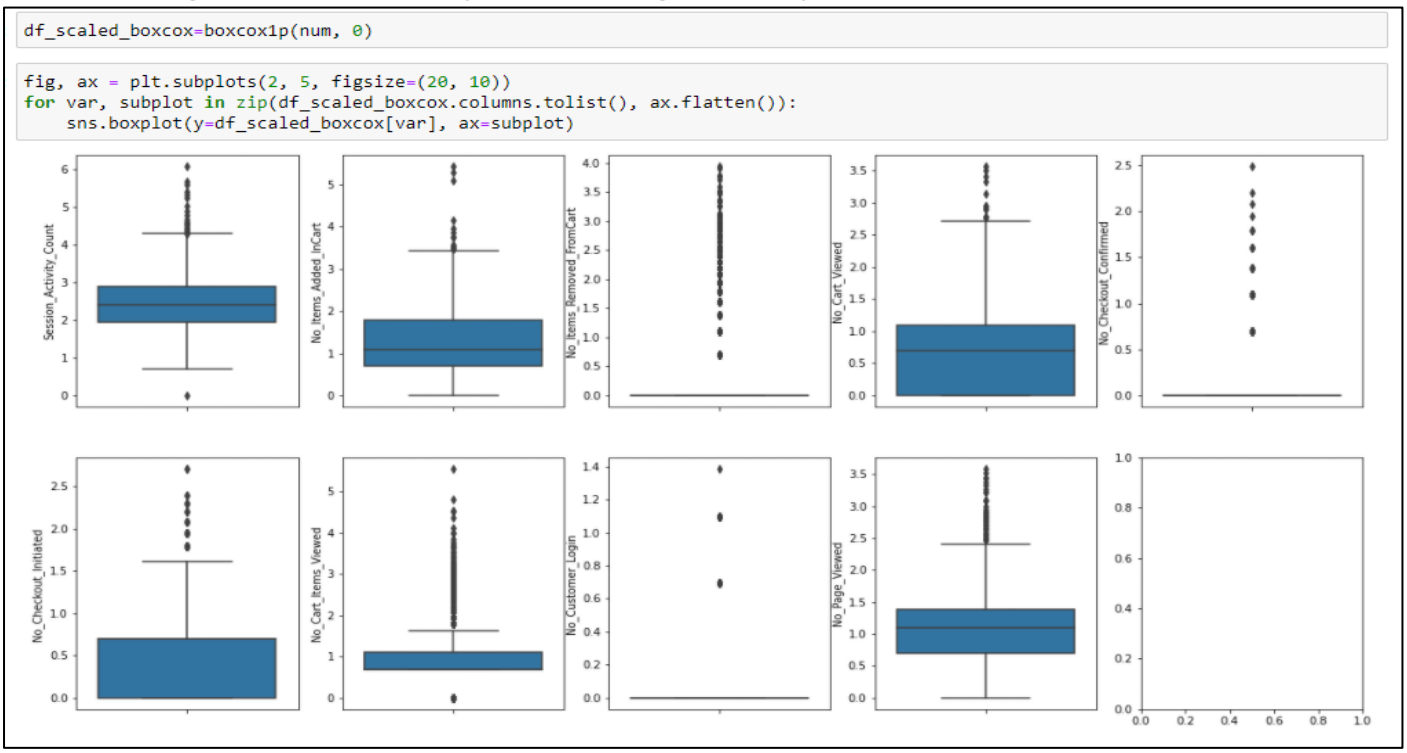
```
null_col = data.columns[data.isna().any()].tolist()
null_col
data['No_Cart_Viewed'].mean()
data['No_Items_Added_InCart'].mean()

data['No_Items_Added_InCart'].fillna(3.48,inplace=True)
data['No_Cart_Viewed'].fillna(1.44,inplace=True)
data[null_col] = data[null_col].astype("int64")
data.isna().sum()
```

```
ID                              0
Is_Product_Details_viewed       0
Session_Activity_Count          0
No_Items_Added_InCart           0
No_Items_Removed_FromCart       0
No_Cart_Viewed                  0
No_Checkout_Confirmed           0
No_Checkout_Initiated           0
No_Cart_Items_Viewed            0
No_Customer_Login               0
No_Page_Viewed                  0
Customer_Segment_Type           0
Cart_Abandoned                  0
dtype: int64
```

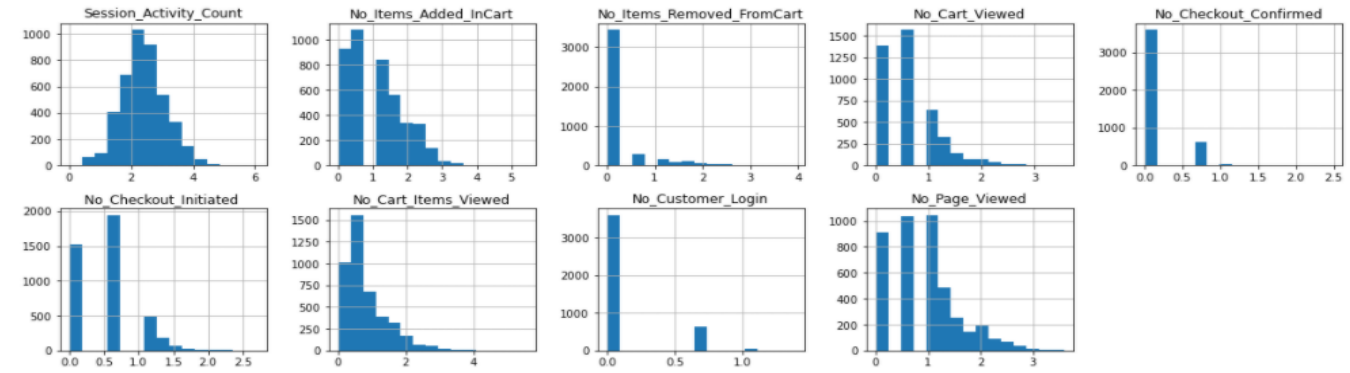
- Taking care of outliers by normalizing data - By BoxCox Normalization

```
df_scaled_boxcox=boxcox1p(num, 0)
```

```
fig, ax = plt.subplots(2, 5, figsize=(20, 10))
for var, subplot in zip(df_scaled_boxcox.columns.tolist(), ax.flatten()):
    sns.boxplot(y=df_scaled_boxcox[var], ax=subplot)
```



```
df_scaled_boxcox.hist(bins=15, figsize=(20, 6), layout=(2, 5));
```




```
df_scaled_boxcox["ID"]=data.ID
df_scaled_boxcox.set_index('ID',inplace=True)
df_scaled_boxcox.reset_index(inplace=True)
df_final=df_scaled_boxcox.join(cat)
df_final.Is_Product_Details_viewed.replace({"Yes":1,"No":0},inplace=True)
df_final.Is_Product_Details_viewed=pd.Categorical(df_final.Is_Product_Details_viewed)
```

Here we can see that our data is almost normally distributed. Now we are ready to go to the next step i.e., feature selection.

FEATURE SELECTION

Here we have taken two models through which we will get feature importance, by comparing the feature importance's of both models we will select our best feature.

1. By RFE(Recursive Feature Elimination), the features selected are –

```
feature_selected = [X[X.columns[l[i]]].name for i,x in enumerate(l)]
feature_selected
```

```
['No_Checkout_Confirmed',
 'No_Checkout_Initiated ',
 'No_Customer_Login',
 'No_Page_Viewed',
 'Is_Product_Details_viewed']
```

2. By Random forest classifier, the features selected are-

```
sorted_feature_weightage_dict = sorted(feature_weightage_dict.items(), key=lambda kv: kv[1], reverse = True)
sorted_feature_weightage_dict
```

```
[('No_Checkout_Confirmed', 0.7341868777064404),
 ('No_Customer_Login', 0.05604221107820115),
 ('Session_Activity_Count', 0.05052857005529642),
 ('No_Checkout_Initiated ', 0.0441808380652188),
 ('No_Page_Viewed', 0.027120211503535017),
 ('No_Items_Added_InCart', 0.025547984898027527),
 ('No_Cart_Items_Viewed', 0.018703955749515146),
 ('No_Cart_Viewed', 0.017810335296951977),
 ('No_Items_Removed_FromCart', 0.010172300692196705),
 ('Is_Product_Details_viewed', 0.009764913736069736),
 ('Customer_Segment_Type', 0.0059418012185472185)]
```

Hence on selecting common features from both models, we get the following features to build our model and predict the outcome.

```
Index(['No_Checkout_Confirmed', 'No_Checkout_Initiated ', 'No_Customer_Login',
       'No_Page_Viewed', 'No_Items_Added_InCart'],
      dtype='object')
```

OVER-SAMPLING USING SMOTE

Since our dataset wasn't balanced so we decided to use SMOTE technique (Amongst the 3 techniques namely under sampling, over Sampling and SMOTE) to balance it first.

```
sm = SMOTE(random_state=2,k_neighbors=5)
X_train, y_train =sm.fit_resample(X_train,y_train)
```

Train-Validation Split after SMOTE

```
X_train_new, X_test_new, y_train_new, y_test_new = train_test_split(X_train, y_train, test_size=0.40, random_state=0)
```

MODEL BUILDING AND PREDICTION

```
lr1 = LogisticRegression()
lr1.fit(X_train_new,y_train_new)
```

```
LogisticRegression()
```

```
y_pred_new = lr1.predict(X_test_new)  #### For SMOTE validation samples
y_pred=lr1.predict(X_test)##### For actual validation samples
```

MODEL EVALUATION

```
print(" accuracy is %2.3f" % accuracy_score(y_test_new, y_pred_new))
print(" Kappa is %f" %cohen_kappa_score(y_test_new, y_pred_new))
```

```
 accuracy is 0.988
 Kappa is 0.976122
```

```
print(" accuracy is %2.3f" % accuracy_score(y_test, y_pred))
print(" Kappa is %f" %cohen_kappa_score(y_test, y_pred))
```

```
 accuracy is 0.984
 Kappa is 0.936154
```

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
print(confusion_matrix( y_test_new ,y_pred_new ))
print(accuracy_score( y_test_new ,y_pred_new ))
print(classification_report( y_test_new ,y_pred_new ))
```

```
[[880   1]
 [ 20 858]]
0.9880613985218875
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       881
           1       1.00      0.98      0.99       878

    accuracy                           0.99      1759
   macro avg       0.99      0.99      0.99      1759
weighted avg       0.99      0.99      0.99      1759
```

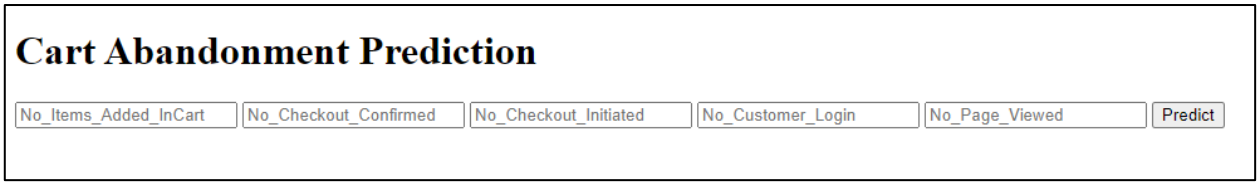
We can see clearly that our model has predicted the outcomes with an accuracy of 98.4%. As our kappa score is also high we can conclude that our model will give the same accuracy with other data.

DEPLOYMENT

We have built an app for our model whose link is given below. Using this link one can predict whether the cart will be abandoned or not.

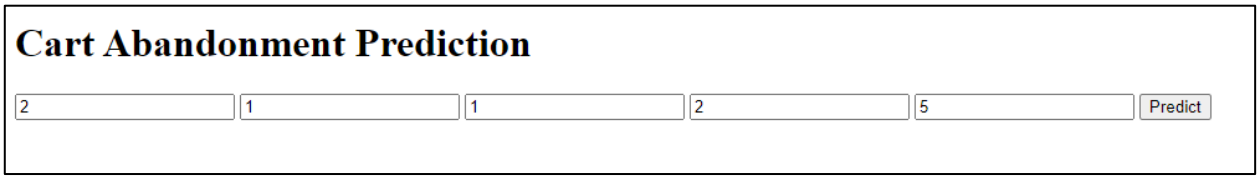
https://cartabandonment.herokuapp.com/

Step 1 – Open the link on browser.



Step 2 – Assign values of your choice to each feature. Refer to the table below.
The possible values of the features affecting cart abandonment can be –

Features	Description	Possible Values
No_Items_Added_InCart	Number of items in cart	Any whole number
No_Checkout_Confirmed	How many times the checkout has been confirmed successfully by the customer.	
No_Checkout_Initiated	How many times the checkout(successful as well as unsuccessful) is being done by the user	
No_Customer_Login	Number of times the customer had did log in	
No_Page_Viewed	Number of pages viewed by the customer	



Step 3 – Click the predict tab.
After assigning the values of choice to each feature the app will automatically predict whether cart abandonment has happened or not using the above model.
If the cart abandonment has happened then it will predict Cart_Abandonment 1 otherwise Cart_Abandonment 0.

CONCLUSION

By using this analysis we can predict whether a customer will do abandonment or not in the future. So that customers can be given some offers to complete the checkout successfully and avoid cart abandonment.