In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
sns.set(color_codes = True)
sns.set(style="whitegrid")
sns.set(rc={'figure.figsize':(7,4)})
sns.set_palette("Set3")
import warnings
warnings.filterwarnings('ignore')
warnings.filterwarnings(action='ignore', category=DeprecationWarning)
pd.set_option('display.max_columns', None)
pd.set_option('display.float_format', lambda x: '%.2f' % x)
```

In [2]:
```
df = pd.read_csv("high_churn_list_model.csv") # data = pd.read_csv("census.csv")
```

In [3]:
```
df.describe(include = ['O']).transpose()[1:]
```

Out[3]:

	count	unique	top	freq
City	2095	876	Los Angeles	93
Offer	2095	6	None	1152
MultipleLines	2095	3	No	1005
InternetType	2095	4	Fiber Optic	933
OnlineSecurity	2095	3	No	1065
OnlineBackup	2095	3	No	920
DeviceProtectionPlan	2095	3	No	956
PremiumTechSupport	2095	3	No	1064
StreamingTV	2095	3	No	865
StreamingMovies	2095	3	No	842
StreamingMusic	2095	3	No	929
UnlimitedData	2095	3	Yes	1406
Contract	2095	3	Month-to-Month	1096
PaymentMethod	2095	3	Bank Withdrawal	1171
ChurnCategory	2095	6	Unknown	1537
ChurnReason	2095	21	Unknown	1537

In [4]:
```
df.describe(exclude = ['O']).transpose()[:-3]
```

Out[4]:

	count	mean	std	min	25%	50%	75
Gender	2095.00	0.50	0.50	0.00	0.00	1.00	1.
Age	2095.00	46.68	16.80	19.00	32.00	46.00	60.
Married	2095.00	0.48	0.50	0.00	0.00	0.00	1.
NumberofDependents	2095.00	0.46	0.98	0.00	0.00	0.00	0.
ZipCode	2095.00	93521.26	1860.72	90001.00	92109.00	93550.00	95359.
Population	2095.00	21501.67	20232.50	11.00	2347.50	16717.00	35109.
NumberofReferrals	2095.00	1.95	3.00	0.00	0.00	0.00	3.
TenureinMonths	2095.00	31.95	24.33	1.00	8.00	28.00	55.
PhoneService	2095.00	0.91	0.29	0.00	1.00	1.00	1.
AvgMonthlyLongDistanceCharges	2095.00	22.78	15.45	0.00	9.27	22.30	36.
InternetService	2095.00	0.79	0.41	0.00	1.00	1.00	1.
AvgMonthlyGBDownload	2095.00	20.72	20.59	0.00	4.00	17.00	27.
PaperlessBilling	2095.00	0.58	0.49	0.00	0.00	1.00	1.
MonthlyCharge	2095.00	63.68	30.97	-10.00	33.52	70.35	89.
TotalCharges	2095.00	2242.08	2223.44	18.80	406.27	1387.45	3704.
TotalRefunds	2095.00	1.82	7.54	0.00	0.00	0.00	0.
TotalExtraDataCharges	2095.00	7.26	25.99	0.00	0.00	0.00	0.
TotalLongDistanceCharges	2095.00	734.42	833.37	0.00	74.54	378.00	1187.
TotalRevenue	2095.00	2981.95	2813.69	21.36	589.41	2047.16	4733.

Determining how many "churned"

In [5]:
```
df['Churn'].value_counts()
```

Out[5]:
```
0    1537
1     558
Name: Churn, dtype: int64
```

In [6]:
```
df.groupby(['Married']).agg({'Churn': 'mean'}).reset_index().sort_values(by='Churn'
```

Out[6]:

	Married	Churn
0	0	0.33
1	1	0.20

In [7]:
```
df.groupby(['Gender']).agg({'Churn': 'mean'}).reset_index().sort_values(by='Churn',
```

Out[7]:

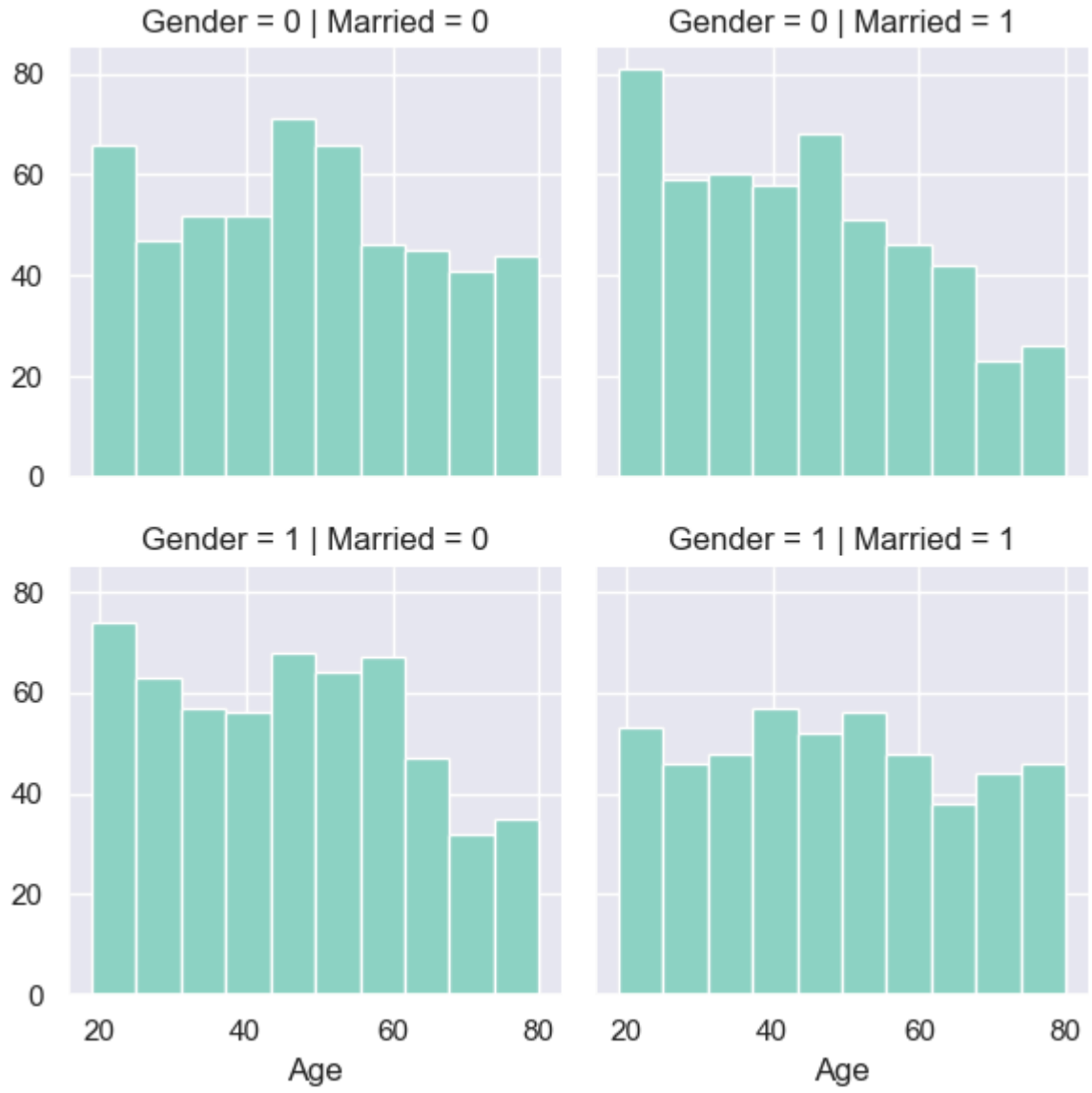
	Gender	Churn
0	0	0.27
1	1	0.26

In [8]:
```
df.groupby(['Gender','Married']).agg({'Churn': 'mean'}).reset_index().sort_values(b
```

Out[8]:

	Gender	Married	Churn
0	0	0	0.35
2	1	0	0.31
3	1	1	0.20
1	0	1	0.19

In [9]:
```
g = sns.FacetGrid(df, col = "Married", row = 'Gender')
g = g.map(plt.hist, "Age")
```



In [10]:
```
fig, [ax0, ax1, ax2] = plt.subplots(1,3, figsize = (14,4))

ax0.hist(df['TotalRevenue'])
ax0.set_xlabel('Total Revenue Distribution')
ax0.axvline(df['TotalRevenue'].mean(), color = "black")

ax1.hist(df.TotalCharges)
ax1.set_xlabel('Total Charges Distribution')
ax1.axvline(df["TotalCharges"].mean(), color = "black");

ax2.hist(df.TenureinMonths)
```

```
ax2.set_xlabel('Tenure in Month distribution')
ax2.axvline(df['TenureinMonths'].mean(), color = "black");

print("Black lines are means")
```

```
Black lines are means
```

In [11]:
```
fig, [ax0, ax1, ax2] = plt.subplots(1,3, figsize = (14,4))

ax0.hist(df['Age'])
ax0.set_xlabel('Age Distribution')
ax0.axvline(df['Age'].mean(), color = "black")

ax1.hist(df['TotalLongDistanceCharges'])
ax1.set_xlabel('Total Long Distance Charges Distribution')
ax1.axvline(df["TotalLongDistanceCharges"].mean(), color = "black");

ax2.hist(df['Population'])
ax2.set_xlabel('Population Distribution')
ax2.axvline(df['Population'].mean(), color = "black");

print("Black lines are means")
```

```
Black lines are means
```

In [12]:
```
sns.countplot(y = df['TotalExtraDataCharges'])
```

Out[12]:
```
<AxesSubplot:xlabel='count', ylabel='TotalExtraDataCharges'>
```

In [13]:
```
sns.violinplot(x=df['TotalRefunds'])
```

Out[13]:
```
<AxesSubplot:xlabel='TotalRefunds', ylabel='TotalExtraDataCharges'>
```

In [14]:
```
sns.boxplot(x=df['MonthlyCharge'])
```

Out[14]:
```
<AxesSubplot:xlabel='MonthlyCharge', ylabel='TotalExtraDataCharges'>
```

In [15]:
```
sns.countplot(df['PaymentMethod'])
```

Out[15]:
```
<AxesSubplot:xlabel='PaymentMethod', ylabel='count'>
```

In [16]:
```
sns.countplot(y=df['ChurnCategory'],order = df['ChurnCategory'].value_counts().inde
```

Out[16]:
```
<AxesSubplot:xlabel='count', ylabel='ChurnCategory'>
```

In [17]:
```
plt.figure(figsize=(10,10))
sns.countplot(y=df['ChurnReason'],order = df['ChurnReason'].value_counts().index)
```

Out[17]:
```
<AxesSubplot:xlabel='count', ylabel='ChurnReason'>
```

In [18]:
```
sns.countplot(df['PhoneService'])
```

Out[18]:
```
<AxesSubplot:xlabel='PhoneService', ylabel='count'>
```

In [19]:
```
sns.countplot(df['PaperlessBilling'],hue=df['Churn'])
```

Out[19]:
```
<AxesSubplot:xlabel='PaperlessBilling', ylabel='count'>
```

In [20]:
```
sns.countplot(df['Contract'],hue = df['Churn'],order = df['Contract'].value_counts(
```

Out[20]:
```
<AxesSubplot:xlabel='Contract', ylabel='count'>
```

In [21]:
```
sns.countplot(df['Offer'],hue = df['Churn'],order = df['Offer'].value_counts().inde
```

Out[21]:
```
<AxesSubplot:xlabel='Offer', ylabel='count'>
```

In [22]:
```
sns.countplot(df['InternetType'], order = df['InternetType'].value_counts().index)
```

Out[22]:
```
<AxesSubplot:xlabel='InternetType', ylabel='count'>
```

Exploratory data analysis

In [23]:
```
quant_df1 = df[[ #nominal
 'Churn',
 'Age',
 'NumberofDependents',
 'Population',
 'NumberofReferrals',
 'TenureinMonths',
 'AvgMonthlyLongDistanceCharges',
'AvgMonthlyGBDownload',
 'MonthlyCharge',
 'TotalCharges',
 'TotalRefunds',
 'TotalExtraDataCharges',
 'TotalLongDistanceCharges',
 'TotalRevenue'
]].copy()

# categorical columns

quant_df2 = df[[ #binary
#'Churn',
'Gender',
'Married',
'PhoneService',
'InternetService',
'PaperlessBilling'
]].copy()

quant_df3 = df[[ #categories (one-hot-encode)
 #'Churn',
 'Offer',
 #'MultipleLines',
 #'InternetType',
 #'OnlineSecurity',
 'OnlineBackup',
 #'DeviceProtectionPlan',
 'PremiumTechSupport',
 'StreamingTV',
 'StreamingMovies',
 'StreamingMusic',
 #'UnlimitedData',
 'Contract',
 #'PaymentMethod',
]].copy()
```

In [24]:
```
quant_df1.corr()
```

Out[24]:

	Churn	Age	NumberofDependents	Population	NumberofReferr
Churn	1.00	0.13	-0.21	0.08	-0
Age	0.13	1.00	-0.09	-0.02	-0
NumberofDependents	-0.21	-0.09	1.00	-0.02	0
Population	0.08	-0.02	-0.02	1.00	-0
NumberofReferrals	-0.30	-0.04	0.28	-0.04	1
TenureinMonths	-0.37	-0.01	0.14	-0.05	0
AvgMonthlyLongDistanceCharges	0.02	-0.00	-0.01	-0.04	0
AvgMonthlyGBDownload	0.02	-0.36	0.17	0.01	0
MonthlyCharge	0.16	0.14	-0.10	-0.00	0
TotalCharges	-0.22	0.05	0.04	-0.04	0
TotalRefunds	-0.02	0.00	-0.01	0.05	0
TotalExtraDataCharges	0.02	0.05	-0.01	0.01	-0
TotalLongDistanceCharges	-0.22	0.01	0.09	-0.05	0
TotalRevenue	-0.24	0.04	0.06	-0.05	0

In [25]:
```
plt.figure(figsize=(10,10))
cmap = sns.diverging_palette(250, 10, as_cmap=True)
sns.heatmap(quant_df1.corr(), cmap = cmap, annot = True);
```

In [26]:
```
quant_df1.corr()['Churn'][1:]
```

Out[26]:
```
Age                               0.13
NumberofDependents               -0.21
Population                        0.08
NumberofReferrals                -0.30
TenureinMonths                   -0.37
AvgMonthlyLongDistanceCharges     0.02
AvgMonthlyGBDownload              0.02
MonthlyCharge                     0.16
TotalCharges                     -0.22
TotalRefunds                     -0.02
TotalExtraDataCharges             0.02
TotalLongDistanceCharges         -0.22
TotalRevenue                     -0.24
Name: Churn, dtype: float64
```

In [27]:
```
print(quant_df1.columns)
```

```
Index(['Churn', 'Age', 'NumberofDependents', 'Population', 'NumberofReferrals',
       'TenureinMonths', 'AvgMonthlyLongDistanceCharges',
       'AvgMonthlyGBDownload', 'MonthlyCharge', 'TotalCharges', 'TotalRefunds',
       'TotalExtraDataCharges', 'TotalLongDistanceCharges', 'TotalRevenue'],
      dtype='object')
```

Let's check our confidense about this statment with logistic regression model:

In [28]:
```
quant_df1['intercept'] = 1
log_mod = sm.Logit(quant_df1['Churn'], quant_df1[['intercept','Age', 'NumberofDeper
       'TenureinMonths', 'AvgMonthlyLongDistanceCharges','AvgMonthlyGBDownload', 'M
```

```
       'TotalExtraDataCharges','TotalRevenue']]).fit()
log_mod.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.419326
         Iterations 7
```

Out[28]:

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	2095
Model:	Logit	**Df Residuals:**	2082
Method:	MLE	**Df Model:**	12
Date:	Sat, 17 Aug 2024	**Pseudo R-squ.:**	0.2765
Time:	07:55:13	**Log-Likelihood:**	-878.49
converged:	True	**LL-Null:**	-1214.2
Covariance Type:	nonrobust	**LLR p-value:**	5.482e-136

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-1.5502	0.304	-5.092	0.000	-2.147	-0.954
Age	0.0152	0.004	3.779	0.000	0.007	0.023
NumberofDependents	-0.4782	0.103	-4.657	0.000	-0.679	-0.277
Population	7.153e-06	2.87e-06	2.495	0.013	1.53e-06	1.28e-05
NumberofReferrals	-0.2881	0.038	-7.628	0.000	-0.362	-0.214
TenureinMonths	-0.0677	0.010	-6.472	0.000	-0.088	-0.047
AvgMonthlyLongDistanceCharges	-0.0113	0.006	-1.995	0.046	-0.022	-0.000
AvgMonthlyGBDownload	0.0055	0.004	1.487	0.137	-0.002	0.013
MonthlyCharge	0.0218	0.003	6.772	0.000	0.015	0.028
TotalCharges	-0.0005	0.000	-1.876	0.061	-0.001	2.03e-05
TotalRefunds	-0.0061	0.008	-0.757	0.449	-0.022	0.010
TotalExtraDataCharges	0.0006	0.002	0.280	0.779	-0.004	0.005
TotalRevenue	0.0006	0.000	2.849	0.004	0.000	0.001

In [29]:
```
# p values
log_mod.pvalues[:].plot.bar()
plt.axhline(y = 0.05);
```

In [30]:
```
#coefficient
log_mod.params[:].plot.bar()
plt.axhline(y = 0.05);
```

In [31]:
```
quant_df_main = {}
for i in log_mod.params[:].to_dict().keys():
    if log_mod.pvalues[i] < 0.05:
        quant_df_main[i] = log_mod.params[i]
    else:
        continue
quant_df_main
sorted(quant_df_main.items(), key=lambda x: x[1]) #sorting by highest to lowest val
```

Out[31]:
```
[('intercept', -1.5501574689605269),
 ('NumberofDependents', -0.4781870547717154),
 ('NumberofReferrals', -0.2880830533840693),
 ('TenureinMonths', -0.06765171718218492),
 ('AvgMonthlyLongDistanceCharges', -0.01125946374212831),
 ('Population', 7.153223010203917e-06),
 ('TotalRevenue', 0.0005601071073698724),
 ('Age', 0.01521105759961699),
 ('MonthlyCharge', 0.021755369254524072)]
```

In [ ]:
```
```

Compute the odds

In [32]:
```
quant_df_main_odds = {k : np.exp(v) for k, v in quant_df_main.items()}
sorted(quant_df_main_odds.items(), key=lambda x: x[1]) #sorting by highest to lowes
```

Out[32]:
```
[('intercept', 0.21221455399029313),
 ('NumberofDependents', 0.6199062297179322),
 ('NumberofReferrals', 0.749699324587366),
 ('TenureinMonths', 0.9345859171174311),
 ('AvgMonthlyLongDistanceCharges', 0.9888036867838459),
 ('Population', 1.0000071532485946),
 ('TotalRevenue', 1.000560263996646),
 ('Age', 1.0153273345533307),
 ('MonthlyCharge', 1.0219937427965726)]
```

In [33]:
```
quant_df1['intercept'] = 1
log_mod2 = sm.Logit(quant_df1['Churn'], quant_df1[['intercept','Age','Population',
log_mod2.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.475116
         Iterations 6
```

Out[33]:

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	2095
Model:	Logit	**Df Residuals:**	2090
Method:	MLE	**Df Model:**	4
Date:	Sat, 17 Aug 2024	**Pseudo R-squ.:**	0.1803
Time:	07:55:13	**Log-Likelihood:**	-995.37
converged:	True	**LL-Null:**	-1214.2
Covariance Type:	nonrobust	**LLR p-value:**	1.925e-93

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-2.9004	0.216	-13.451	0.000	-3.323	-2.478
Age	0.0140	0.003	4.211	0.000	0.007	0.020
Population	6.696e-06	2.69e-06	2.491	0.013	1.43e-06	1.2e-05
MonthlyCharge	0.0342	0.002	14.282	0.000	0.029	0.039
TotalRevenue	-0.0005	2.88e-05	-16.004	0.000	-0.001	-0.000

Log Mod 2 (Binary)

In [34]:
```
quant_df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2095 entries, 0 to 2094
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Gender            2095 non-null   int64
 1   Married           2095 non-null   int64
 2   PhoneService      2095 non-null   int64
 3   InternetService   2095 non-null   int64
 4   PaperlessBilling  2095 non-null   int64
dtypes: int64(5)
memory usage: 82.0 KB
```

In [35]:
```
quant_df1['intercept'] = 1
log_mod2 = sm.Logit(quant_df1['Churn'], quant_df2).fit()
log_mod2.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.562800
         Iterations 5
```

Out[35]:

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	2095
Model:	Logit	**Df Residuals:**	2090
Method:	MLE	**Df Model:**	4
Date:	Sat, 17 Aug 2024	**Pseudo R-squ.:**	0.02897
Time:	07:55:13	**Log-Likelihood:**	-1179.1
converged:	True	**LL-Null:**	-1214.2
Covariance Type:	nonrobust	**LLR p-value:**	1.902e-14

	coef	std err	z	P>|z|	[0.025	0.975]
Gender	-0.3192	0.098	-3.271	0.001	-0.510	-0.128
Married	-0.9272	0.101	-9.148	0.000	-1.126	-0.729
PhoneService	-0.9664	0.106	-9.117	0.000	-1.174	-0.759
InternetService	0.2870	0.115	2.490	0.013	0.061	0.513
PaperlessBilling	0.4793	0.110	4.355	0.000	0.264	0.695

In [36]:
```
quant_df_main2 = {}
for i in log_mod2.params[:].to_dict().keys():
    if log_mod2.pvalues[i] < 0.05:
        quant_df_main2[i] = log_mod2.params[i].round(4)
    else:
        continue
quant_df_main2
sorted(quant_df_main2.items(), key=lambda x: x[1]) #sorting by highe
```

Out[36]:
```
[('PhoneService', -0.9664),
 ('Married', -0.9272),
 ('Gender', -0.3192),
 ('InternetService', 0.287),
 ('PaperlessBilling', 0.4793)]
```

In [37]:
```
quant_df_main_odds2 = {k : np.exp(v) for k, v in quant_df_main2.items()}
sorted(quant_df_main_odds2.items(), key=lambda x: x[1])
```

Out[37]:
```
[('PhoneService', 0.3804501964490625),
 ('Married', 0.3956600088557378),
 ('Gender', 0.7267301887330189),
 ('InternetService', 1.3324242134756759),
 ('PaperlessBilling', 1.6149435459572175)]
```

Log Mod 3 (Category)

In [38]:
```
quant_df3 = df[[
 'Churn',
 'Offer',
 'InternetType',
 'Contract',
 'PaymentMethod',
]].copy()
```

In [39]:
```
quant_df3.head()
```

Out[39]:

	Churn	Offer	InternetType	Contract	PaymentMethod
0	1	Offer E	Fiber Optic	Month-to-Month	Bank Withdrawal
1	1	None	Fiber Optic	Month-to-Month	Bank Withdrawal
2	1	None	Fiber Optic	Month-to-Month	Bank Withdrawal
3	1	None	DSL	Month-to-Month	Bank Withdrawal
4	1	Offer D	Fiber Optic	Month-to-Month	Credit Card

In [40]:
```
quant_df3 = pd.get_dummies(quant_df3)
```

In [41]:
```
quant_df3.describe().transpose()
```

Out[41]:

	count	mean	std	min	25%	50%	75%	max
Churn	2095.00	0.27	0.44	0.00	0.00	0.00	1.00	1.00
Offer_None	2095.00	0.55	0.50	0.00	0.00	1.00	1.00	1.00
Offer_Offer A	2095.00	0.07	0.25	0.00	0.00	0.00	0.00	1.00
Offer_Offer B	2095.00	0.12	0.32	0.00	0.00	0.00	0.00	1.00
Offer_Offer C	2095.00	0.06	0.23	0.00	0.00	0.00	0.00	1.00
Offer_Offer D	2095.00	0.09	0.28	0.00	0.00	0.00	0.00	1.00
Offer_Offer E	2095.00	0.12	0.32	0.00	0.00	0.00	0.00	1.00
InternetType_Cable	2095.00	0.12	0.32	0.00	0.00	0.00	0.00	1.00
InternetType_DSL	2095.00	0.22	0.42	0.00	0.00	0.00	0.00	1.00
InternetType_Fiber Optic	2095.00	0.45	0.50	0.00	0.00	0.00	1.00	1.00
InternetType_Unknown	2095.00	0.21	0.41	0.00	0.00	0.00	0.00	1.00
Contract_Month-to-Month	2095.00	0.52	0.50	0.00	0.00	1.00	1.00	1.00
Contract_One Year	2095.00	0.21	0.41	0.00	0.00	0.00	0.00	1.00
Contract_Two Year	2095.00	0.27	0.44	0.00	0.00	0.00	1.00	1.00
PaymentMethod_Bank Withdrawal	2095.00	0.56	0.50	0.00	0.00	1.00	1.00	1.00
PaymentMethod_Credit Card	2095.00	0.40	0.49	0.00	0.00	0.00	1.00	1.00
PaymentMethod_Mailed Check	2095.00	0.05	0.21	0.00	0.00	0.00	0.00	1.00

In [42]:
```
quant_df3.shape
```

Out[42]:
```
(2095, 17)
```

In [43]:
```
log_mod3 = sm.Logit(quant_df3['Churn'], quant_df3[['Offer_None','Offer_Offer A','Of
                                                   'Offer_Offer C','Offer_Offer E',

log_mod3.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.542199
         Iterations 7
```

Out[43]:

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	2095
Model:	Logit	**Df Residuals:**	2089
Method:	MLE	**Df Model:**	5
Date:	Sat, 17 Aug 2024	**Pseudo R-squ.:**	0.06452
Time:	07:55:14	**Log-Likelihood:**	-1135.9
converged:	True	**LL-Null:**	-1214.2
Covariance Type:	nonrobust	**LLR p-value:**	5.044e-32

	coef	std err	z	P>|z|	[0.025	0.975]
Offer_None	-0.9729	0.066	-14.733	0.000	-1.102	-0.843
Offer_Offer A	-2.9886	0.387	-7.715	0.000	-3.748	-2.229
Offer_Offer B	-2.0384	0.201	-10.146	0.000	-2.432	-1.645
Offer_Offer C	-1.2040	0.219	-5.487	0.000	-1.634	-0.774
Offer_Offer E	0.0877	0.126	0.694	0.488	-0.160	0.335
Offer_Offer D	-1.0282	0.166	-6.177	0.000	-1.354	-0.702

In [44]:
```
log_mod3 = sm.Logit(quant_df3['Churn'], quant_df3[['Offer_None','Offer_Offer A','Of
                                                   'Offer_Offer C','Offer_Offer E',
log_mod3.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.542199
         Iterations 7
```

Out[44]:

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	2095
Model:	Logit	**Df Residuals:**	2089
Method:	MLE	**Df Model:**	5
Date:	Sat, 17 Aug 2024	**Pseudo R-squ.:**	0.06452
Time:	07:55:14	**Log-Likelihood:**	-1135.9
converged:	True	**LL-Null:**	-1214.2
Covariance Type:	nonrobust	**LLR p-value:**	5.044e-32

	coef	std err	z	P>|z|	[0.025	0.975]
Offer_None	-0.9729	0.066	-14.733	0.000	-1.102	-0.843
Offer_Offer A	-2.9886	0.387	-7.715	0.000	-3.748	-2.229
Offer_Offer B	-2.0384	0.201	-10.146	0.000	-2.432	-1.645
Offer_Offer C	-1.2040	0.219	-5.487	0.000	-1.634	-0.774
Offer_Offer E	0.0877	0.126	0.694	0.488	-0.160	0.335
Offer_Offer D	-1.0282	0.166	-6.177	0.000	-1.354	-0.702

In [45]:
```
quant_df_main = {}
for i in log_mod3.params[:].to_dict().keys():
```

```
    if log_mod3.pvalues[i] < 0.05:
        quant_df_main[i] = log_mod3.params[i].round(4)
    else:
        continue
quant_df_main
sorted(quant_df_main.items(), key=lambda x: x[1]) #sorting by highest to lowest val
```

Out[45]:
```
[('Offer_Offer A', -2.9886),
 ('Offer_Offer B', -2.0384),
 ('Offer_Offer C', -1.204),
 ('Offer_Offer D', -1.0282),
 ('Offer_None', -0.9729)]
```

In [46]:
```
quant_df3['intercept'] = 1
log_mod3 = sm.Logit(quant_df3['Churn'], quant_df3[['Contract_Month-to-Month',
                                                   'Contract_One Year','Contract_Tw
log_mod3.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.465441
         Iterations 8
```

Out[46]:

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	2095
Model:	Logit	**Df Residuals:**	2092
Method:	MLE	**Df Model:**	2
Date:	Sat, 17 Aug 2024	**Pseudo R-squ.:**	0.1970
Time:	07:55:14	**Log-Likelihood:**	-975.10
converged:	True	**LL-Null:**	-1214.2
Covariance Type:	nonrobust	**LLR p-value:**	1.377e-104

	coef	std err	z	P>|z|	[0.025	0.975]
Contract_Month-to-Month	-0.2088	0.061	-3.437	0.001	-0.328	-0.090
Contract_One Year	-1.9148	0.143	-13.377	0.000	-2.195	-1.634
Contract_Two Year	-3.9157	0.305	-12.859	0.000	-4.512	-3.319

In [47]:
```
quant_df3['intercept'] = 1
log_mod3 = sm.Logit(quant_df3['Churn'], quant_df3[['PaymentMethod_Bank Withdrawal',
                                                   'PaymentMethod_Mailed Check']]).f
log_mod3.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.554132
         Iterations 6
```

Out[47]:

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	2095
Model:	Logit	**Df Residuals:**	2092
Method:	MLE	**Df Model:**	2
Date:	Sat, 17 Aug 2024	**Pseudo R-squ.:**	0.04393
Time:	07:55:14	**Log-Likelihood:**	-1160.9
converged:	True	**LL-Null:**	-1214.2
Covariance Type:	nonrobust	**LLR p-value:**	6.826e-24

	coef	std err	z	P>|z|	[0.025	0.975]
PaymentMethod_Bank Withdrawal	-0.6791	0.062	-10.980	0.000	-0.800	-0.558
PaymentMethod_Credit Card	-1.7474	0.098	-17.885	0.000	-1.939	-1.556
PaymentMethod_Mailed Check	-0.2754	0.207	-1.330	0.184	-0.681	0.131

In [ ]:

In [48]:
```
quant_df3['intercept'] = 1
log_mod3 = sm.Logit(quant_df1['Churn'], quant_df3[['InternetType_Cable','InternetTy
                                                   'InternetType_Fiber Optic','Inter
log_mod3.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.536571
         Iterations 6
```

Out[48]:

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	2095
Model:	Logit	**Df Residuals:**	2091
Method:	MLE	**Df Model:**	3
Date:	Sat, 17 Aug 2024	**Pseudo R-squ.:**	0.07423
Time:	07:55:14	**Log-Likelihood:**	-1124.1
converged:	True	**LL-Null:**	-1214.2
Covariance Type:	nonrobust	**LLR p-value:**	7.736e-39

	coef	std err	z	P>|z|	[0.025	0.975]
InternetType_Cable	-1.2397	0.153	-8.096	0.000	-1.540	-0.940
InternetType_DSL	-1.4655	0.118	-12.391	0.000	-1.697	-1.234
InternetType_Fiber Optic	-0.4064	0.067	-6.080	0.000	-0.537	-0.275
InternetType_Unknown	-2.2687	0.162	-13.997	0.000	-2.586	-1.951

In [ ]:

In [ ]: