Project Purpose

This project uses the data that was previously cleaned in the Jupyter Notebook "Data Cleaning". The objective of this analysis is to use the majority of the data set provided in the course to determine how customer churn might be predicted using logistic regression. I will be performing calculations that will reduce the available features to those variables which will be significant in answering the research question "Which factors can help predict whether a customer will end their contract (churn)?". I hope to obtain a strong predictive model to assist the telecommunications company with decisions on how to avoid the loss of customers.

Steps to Prepare the Data

The following steps will be taken to prepare the data for analysis:

1. Import data to pandas dataframe
2. Determine variable types and those which may require further investigation
3. Convert binary variables to yes = 1 and no = 0
4. Investigate potential categorical variables using bar charts, then convert categorical values to dummy variables
5. Drop columns that will not be used in logistic regression

Step 1

In [1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import cross_val_predict, train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings('ignore') # Ignore warning messages for readability
```

In [106]:
```
# Read in dataset and view head
df = pd.read_csv('churn_clean.csv')
pd.options.display.max_columns = None
df.head()
```

Out[106]:

	CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	I
0	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	e885b299883d4f9fb18e39c75155d990	Point Baker	AK	Prince of Wales-Hyder	99927	56.25100	-133.37571	
1	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	f2de8bef964785f41a2959829830fb8a	West Branch	MI	Ogemaw	48661	44.32893	-84.24080	
2	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	f1784cfa9f6d92ae816197eb175d3c71	Yamhill	OR	Yamhill	97148	45.35589	-123.24657	
3	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	dc8a365077241bb5cd5ccd305136b05e	Del Mar	CA	San Diego	92014	32.96687	-117.24798	
4	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	aabb64a116e83fdc4befc1fbab1663f9	Needville	TX	Fort Bend	77461	29.38012	-95.80673	

Step 2

In [107]:
```
# View list of columns, data types, and missing values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   CaseOrder             10000 non-null  int64  
 1   Customer_id           10000 non-null  object 
 2   Interaction           10000 non-null  object 
 3   UID                   10000 non-null  object 
 4   City                  10000 non-null  object 
 5   State                 10000 non-null  object 
 6   County                10000 non-null  object 
 7   Zip                   10000 non-null  int64  
 8   Lat                   10000 non-null  float64
 9   Lng                   10000 non-null  float64
 10  Population            10000 non-null  int64  
 11  Area                  10000 non-null  object 
 12  TimeZone              10000 non-null  object 
 13  Job                   10000 non-null  object 
 14  Children              10000 non-null  int64  
 15  Age                   10000 non-null  int64  
 16  Income                10000 non-null  float64
 17  Marital               10000 non-null  object 
 18  Gender                10000 non-null  object 
 19  Churn                 10000 non-null  object 
 20  Outage_sec_perweek    10000 non-null  float64
 21  Email                 10000 non-null  int64  
 22  Contacts              10000 non-null  int64  
 23  Yearly_equip_failure  10000 non-null  int64  
 24  Techie                10000 non-null  object 
 25  Contract              10000 non-null  object 
 26  Port_modem            10000 non-null  object 
 27  Tablet                10000 non-null  object 
 28  InternetService       10000 non-null  object 
 29  Phone                 10000 non-null  object 
 30  Multiple              10000 non-null  object 
 31  OnlineSecurity        10000 non-null  object 
 32  OnlineBackup          10000 non-null  object 
 33  DeviceProtection      10000 non-null  object 
 34  TechSupport           10000 non-null  object 
 35  StreamingTV           10000 non-null  object 
 36  StreamingMovies       10000 non-null  object 
 37  PaperlessBilling      10000 non-null  object 
 38  PaymentMethod         10000 non-null  object 
 39  Tenure                10000 non-null  float64
 40  MonthlyCharge         10000 non-null  float64
 41  Bandwidth_GB_Year     10000 non-null  float64
 42  Item1                 10000 non-null  int64  
 43  Item2                 10000 non-null  int64  
 44  Item3                 10000 non-null  int64  
 45  Item4                 10000 non-null  int64  
 46  Item5                 10000 non-null  int64  
 47  Item6                 10000 non-null  int64  
 48  Item7                 10000 non-null  int64  
 49  Item8                 10000 non-null  int64  
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

Results of Step 2

- As expected per the instructor guidance, there are no missing values and the dataset is clean and ready to prepare for analysis.

Breakdown of Variables

Identification variables:

- CaseOrder, Customer_id, Interaction, UID

String variables:

- City, State, County

Numeric variables:

- Zip, Lat, Lng, Population, Children, Age, Income, Outage_sec_perweek, Email, Contacts, Yearly_equip_failure, Tenure, MonthlyCharge, Bandwidth_GB_Year, Item1, Item2, Item3, Item4, Item5, Item6, Item7, Item8

Binary variables:

- Churn, Techie, Port_modem, Tablet, Phone, Multiple, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, PaperlessBilling

Possible Categorical variables (may be string, will investigate further):

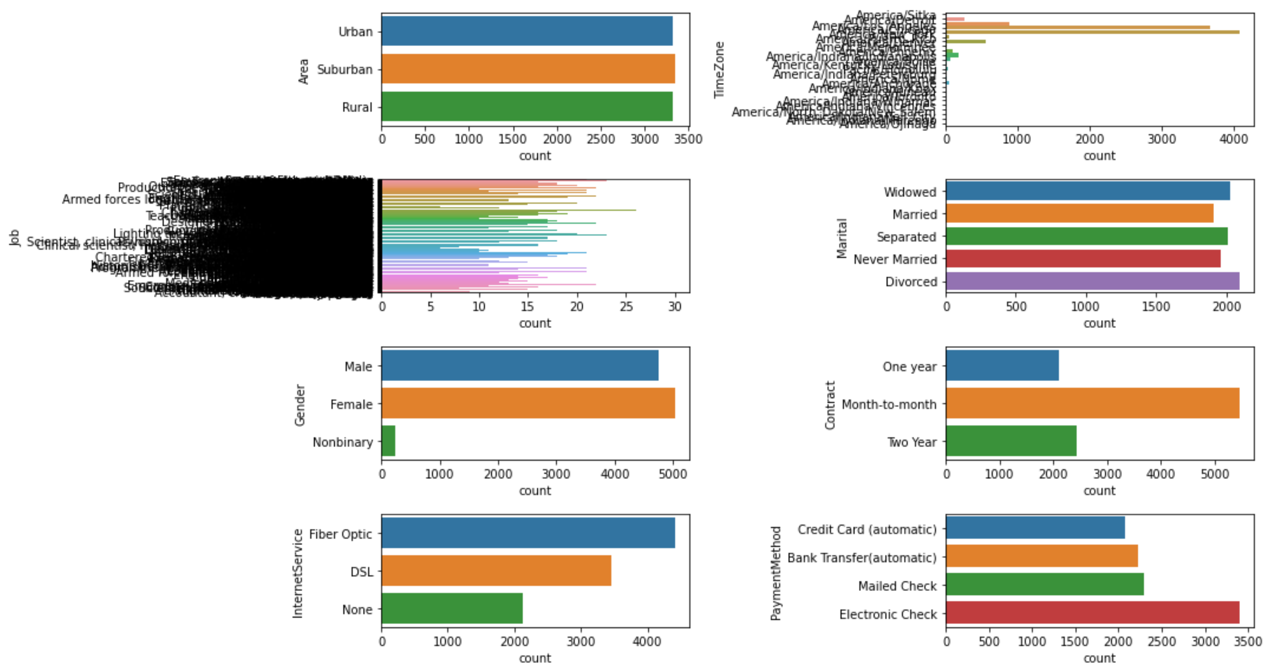
- Area, TimeZone, Job, Marital, Gender, Contract, InternetService, PaymentMethod

Step 3

In [108]:
```
# Convert binary variables into yes = 1, no = 0 (ref 1)
cols = ['Churn', 'Techie', 'Port_modem', 'Tablet', 'Phone', 'Multiple', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling']
df[cols] = df[cols].replace(to_replace = ['No', 'Yes'], value = [0, 1])
```

Step 4

In [109]:
```
# View bar charts for potential categorical variables to determine number of categories
figure, axes = plt.subplots(nrows=4, ncols=2, figsize=(15,8))
plt.subplot(4, 2, 1)
sns.countplot(data = df, y = 'Area')
plt.subplot(4, 2, 2)
sns.countplot(data = df, y = 'TimeZone')
plt.subplot(4, 2, 3)
sns.countplot(data = df, y = 'Job')
plt.subplot(4, 2, 4)
sns.countplot(data = df, y = 'Marital')
plt.subplot(4, 2, 5)
sns.countplot(data = df, y = 'Gender')
plt.subplot(4, 2, 6)
sns.countplot(data = df, y = 'Contract')
plt.subplot(4, 2, 7)
sns.countplot(data = df, y = 'InternetService')
plt.subplot(4, 2, 8)
sns.countplot(data = df, y = 'PaymentMethod')
figure.tight_layout()
plt.show();
```



- Timezone and Job seem to have too many possible categories for meaningful separation into categories. They will be dropped with the string variables. The remaining categorical variables will be converted into dummy variables.

In [110]:
```
# Create separate variables for each categorical value, with a 1 if the value is present in that row and 0 if
 not present
df = pd.get_dummies(data=df, columns=['Area', 'Marital', 'Gender', 'Contract', 'InternetService', 'PaymentMeth
od'])
```

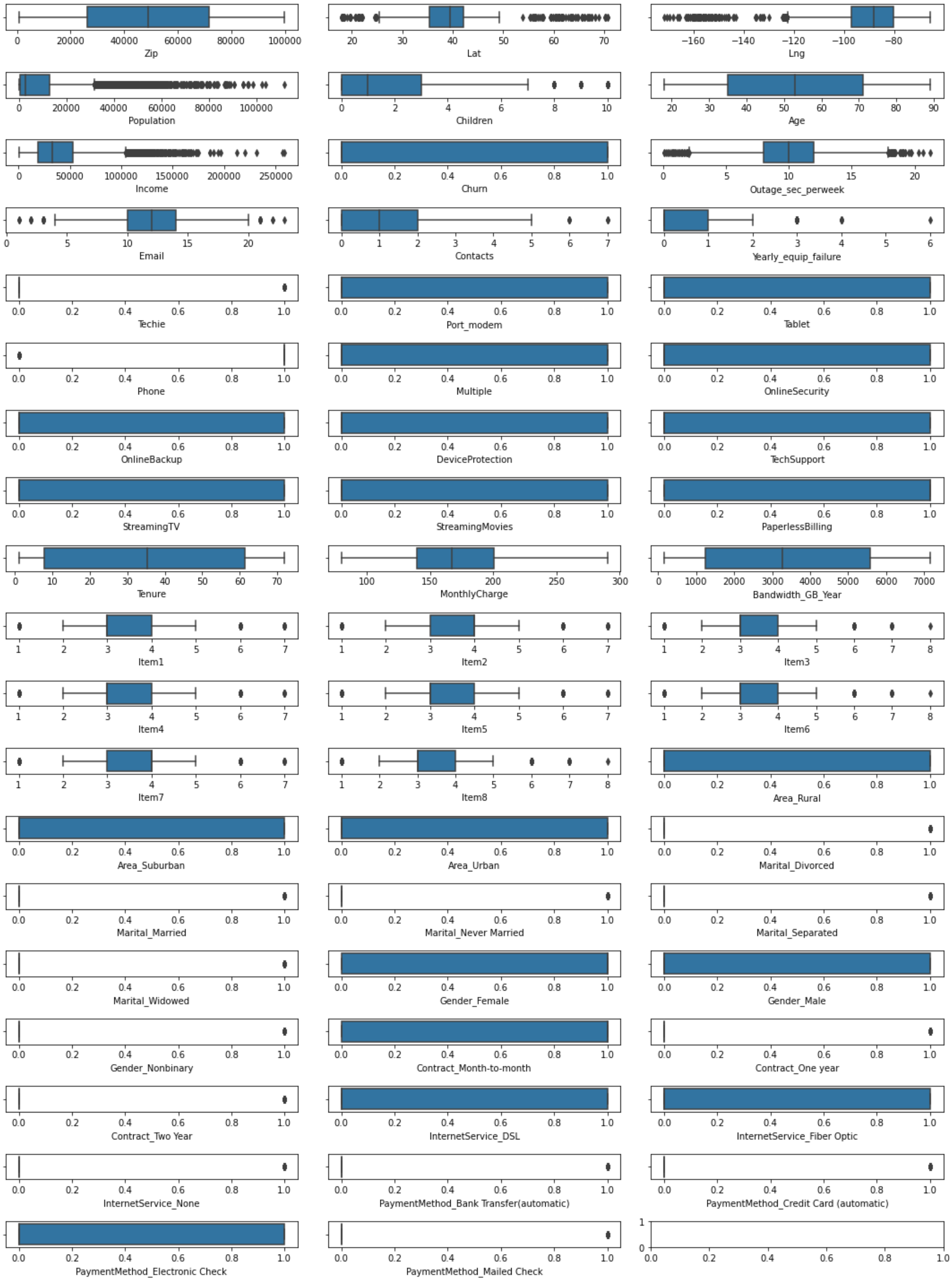
Step 5

In [111]:
```
# Drop columns not needed for analysis
drops = ['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State', 'County', 'TimeZone', 'Job']
df = df.drop(drops, axis = 1)
```

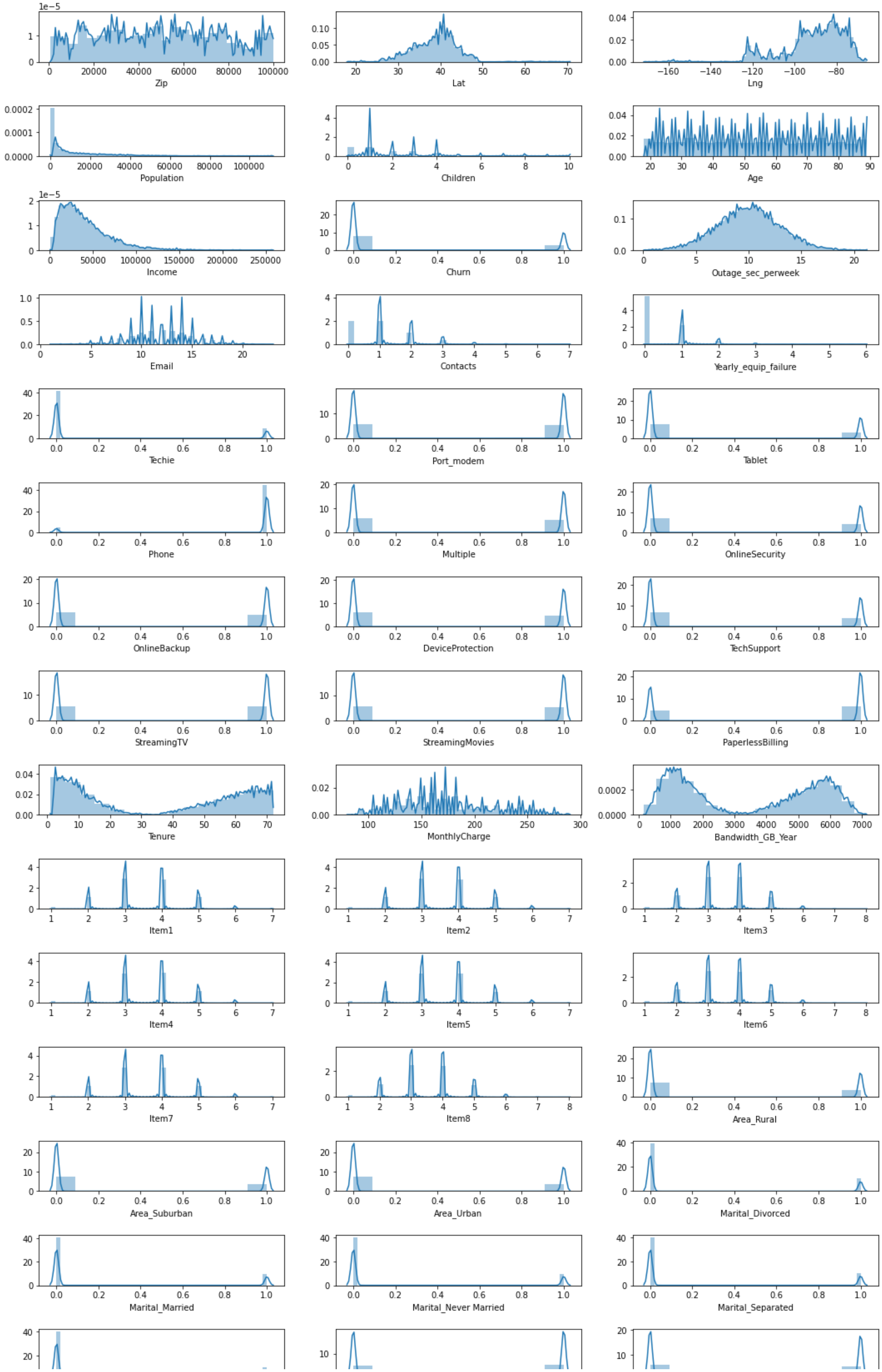
VISUALIZATIONS

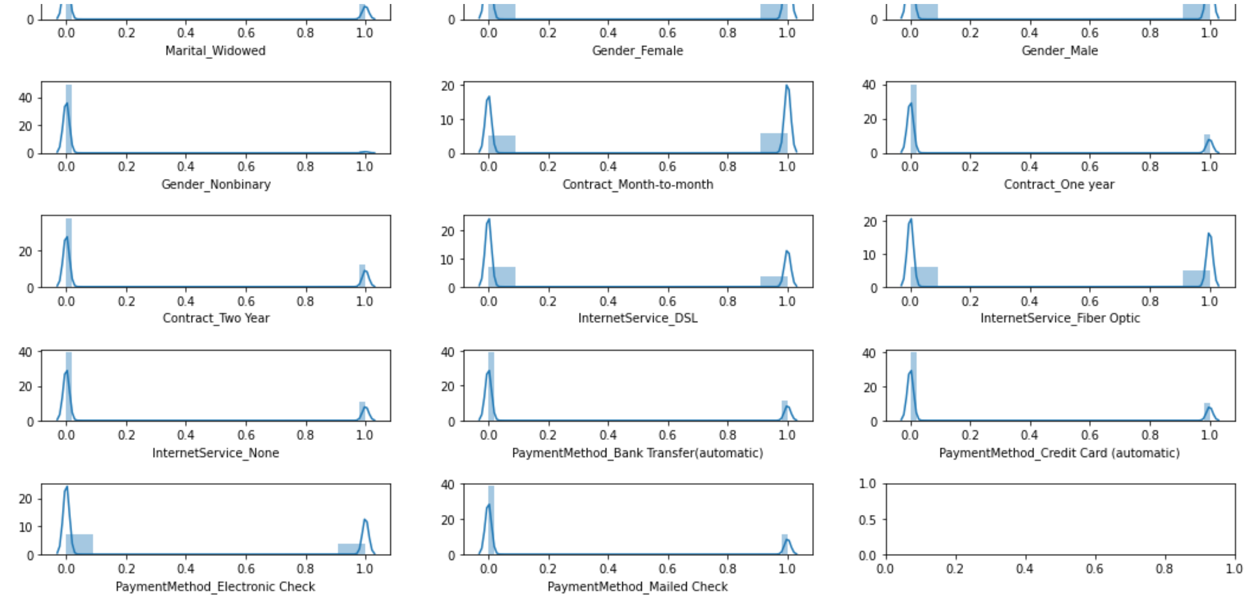
Univariate Visualizations

In [112]:
```
# Display boxplots of numeric columns (ref 2)
cols = ['Zip', 'Lat', 'Lng', 'Population', 'Children', 'Age', 'Income', 'Churn', 'Outage_sec_perweek', 'Email'
, \
        'Contacts', 'Yearly_equip_failure', 'Techie', 'Port_modem', 'Tablet', 'Phone', 'Multiple', 'OnlineSecu
rity', \
        'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBillin
g', \
        'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6',
'Item7', \
        'Item8', 'Area_Rural', 'Area_Suburban', 'Area_Urban', 'Marital_Divorced', 'Marital_Married', 'Marital_
Never Married',\
        'Marital_Separated', 'Marital_Widowed', 'Gender_Female', 'Gender_Male', 'Gender_Nonbinary', 'Contract_
Month-to-month', \
        'Contract_One year', 'Contract_Two Year', 'InternetService_DSL', 'InternetService_Fiber Optic', 'Inter
netService_None', \
        'PaymentMethod_Bank Transfer(automatic)', 'PaymentMethod_Credit Card (automatic)', 'PaymentMethod_Elec
tronic Check', \
        'PaymentMethod_Mailed Check']
f, axes = plt.subplots(round(len(cols)/3), 3, figsize=(15,20))
y = 0;
for col in cols:
    i, j = divmod(y, 3)
    sns.boxplot(x=df[col], ax=axes[i, j])
    y = y + 1

plt.tight_layout()
plt.show();
```


Zip
Lat
Lng
Population
Children
Age
Income
Churn
Outage_sec_perweek
Email
Contacts
Yearly_equip_failure
Techie
Port_modem
Tablet
Phone
Multiple
OnlineSecurity
OnlineBackup
DeviceProtection
TechSupport
StreamingTV
StreamingMovies
PaperlessBilling
Tenure
MonthlyCharge
Bandwidth_GB_Year
Item1
Item2
Item3
Item4
Item5
Item6
Item7
Item8
Area_Rural
Area_Suburban
Area_Urban
Marital_Divorced
Marital_Married
Marital_Never Married
Marital_Separated
Marital_Widowed
Gender_Female
Gender_Male
Gender_Nonbinary
Contract_Month-to-month
Contract_One year
Contract_Two Year
InternetService_DSL
InternetService_Fiber Optic
InternetService_None
PaymentMethod_Bank Transfer(automatic)
PaymentMethod_Credit Card (automatic)
PaymentMethod_Electronic Check
PaymentMethod_Mailed Check

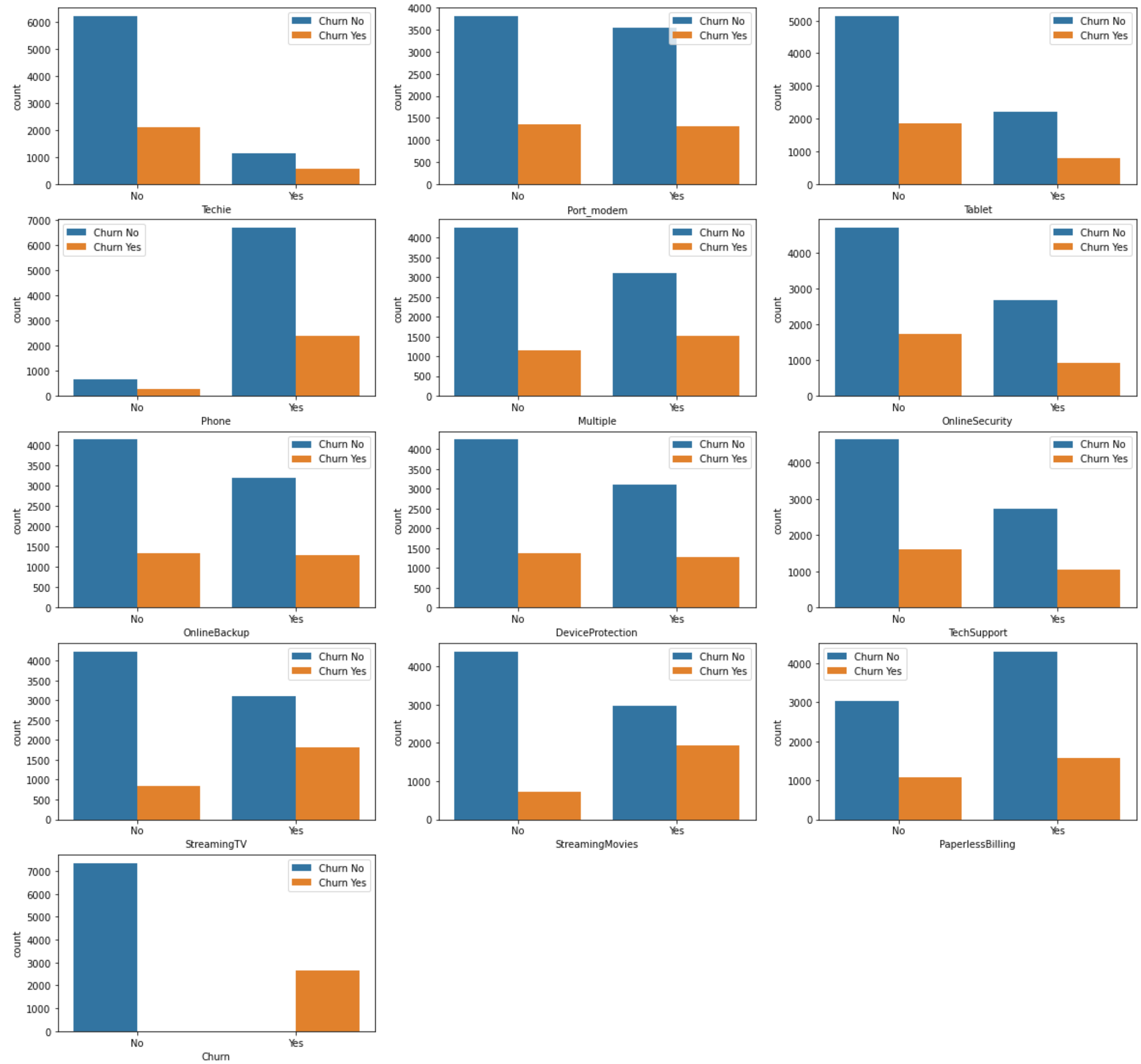
In [113]:
```
# Display histograms of numeric columns (ref 2)
f, axes = plt.subplots(round(len(cols)/3), 3, figsize=(15,30))
y = 0;
for col in cols:
    i, j = divmod(y, 3)
    sns.distplot(df[col], ax=axes[i, j], kde_kws={'bw':0.01})
    y = y + 1

plt.tight_layout()
plt.show();
```


Zip
Lat
Lng
Population
Children
Age
Income
Churn
Outage_sec_perweek
Email
Contacts
Yearly_equip_failure
Techie
Port_modem
Tablet
Phone
Multiple
OnlineSecurity
OnlineBackup
DeviceProtection
TechSupport
StreamingTV
StreamingMovies
PaperlessBilling
Tenure
MonthlyCharge
Bandwidth_GB_Year
Item1
Item2
Item3
Item4
Item5
Item6
Item7
Item8
Area_Rural
Area_Suburban
Area_Urban
Marital_Divorced
Marital_Married
Marital_Never Married
Marital_Separated



<u>Bivariate Visualizations</u>

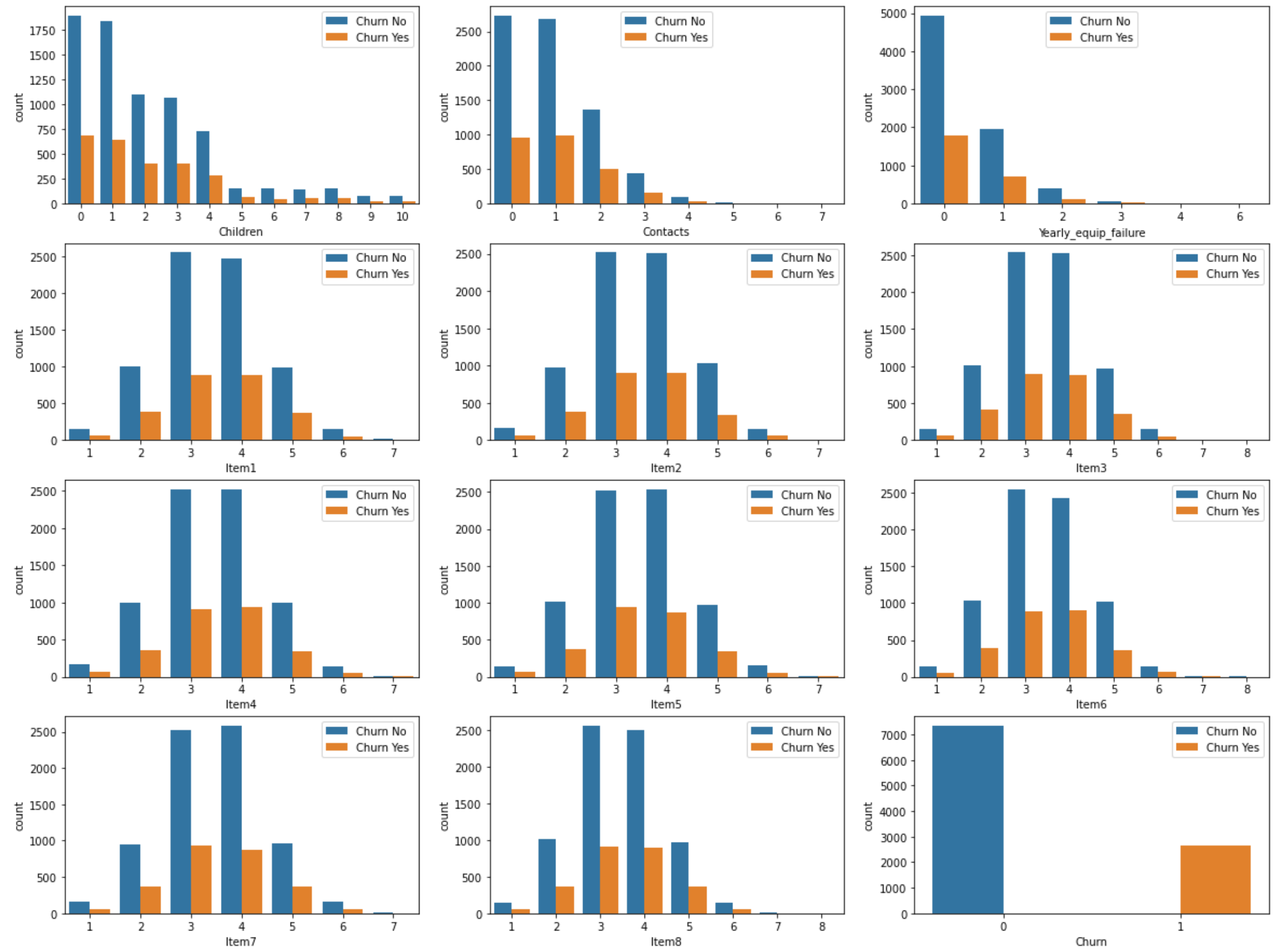
In [114]:
```
# Plot split bar charts for binary variables by Churn
df_bin = df[['Techie', 'Port_modem', 'Tablet', 'Phone', 'Multiple', 'OnlineSecurity', 'OnlineBackup', 'DeviceP
rotection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'PaperlessBilling', 'Churn']]
count=1
plt.subplots(figsize=(20, 80))
for i in df_bin.columns:
    plt.subplot(20,3,count)
    ax = sns.countplot(x = i, hue = 'Churn', data = df_bin)
    legend_labels, _= ax.get_legend_handles_labels()
    ax.legend(legend_labels, ['Churn No','Churn Yes'])
    ax.set_xticklabels(('No', 'Yes'))
    count+=1
plt.show();
```



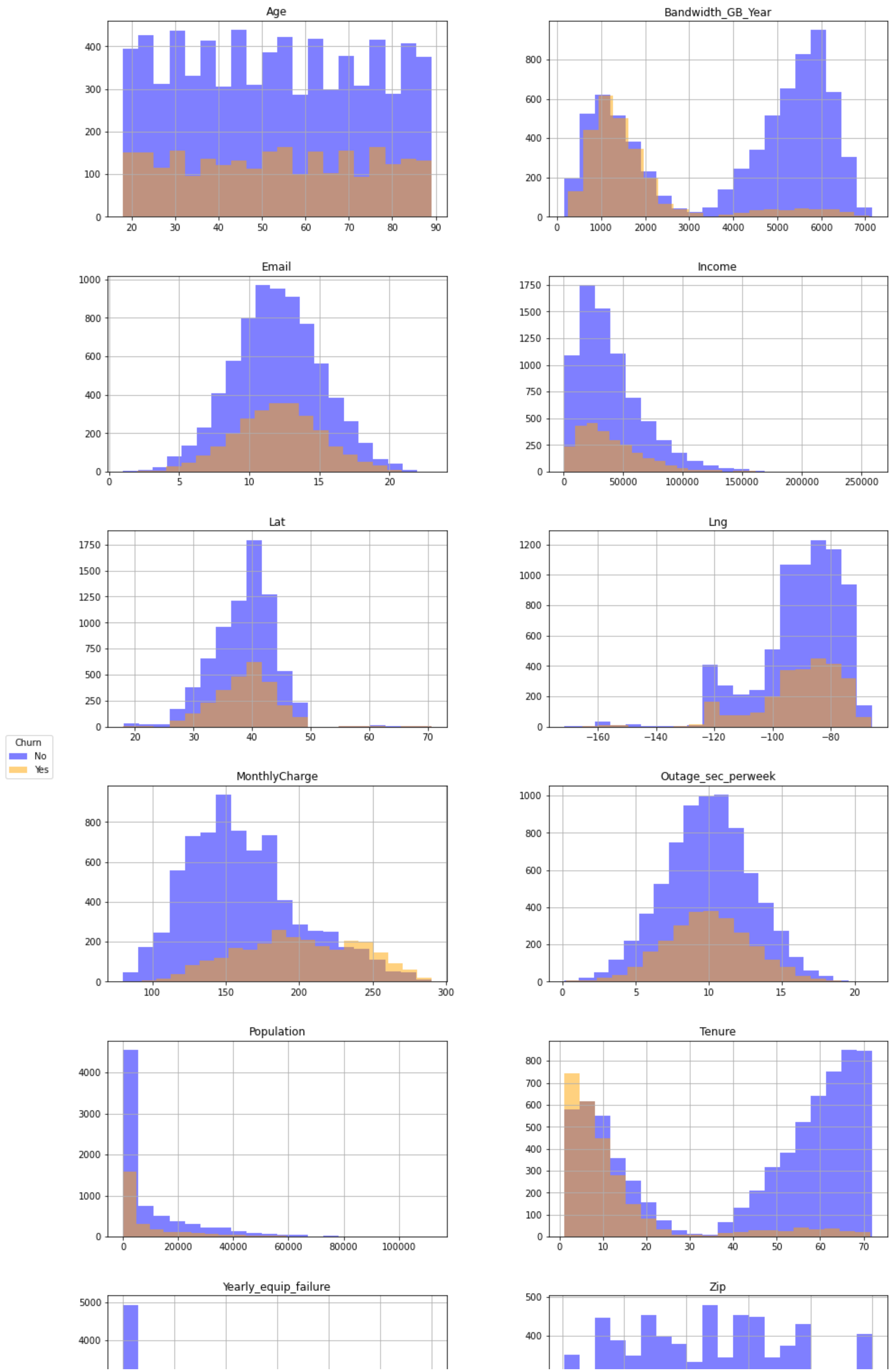
In [115]:
```
# Plot split bar charts for dummy categorical variables by Churn
df_cat = df[['Area_Suburban', 'Area_Urban', 'Marital_Divorced', 'Marital_Married', 'Marital_Never Married',
             'Marital_Separated', 'Marital_Widowed', 'Gender_Female', 'Gender_Male', 'Gender_Nonbinary',
             'Contract_Month-to-month', 'Contract_One year', 'Contract_Two Year', 'InternetService_DSL',
             'InternetService_Fiber Optic', 'InternetService_None', 'PaymentMethod_Bank Transfer(automatic)',
             'PaymentMethod_Credit Card (automatic)', 'PaymentMethod_Electronic Check', 'PaymentMethod_Mailed
 Check', 'Churn']]
count=1
plt.subplots(figsize=(20, 80))
for i in df_cat.columns:
    plt.subplot(20,3,count)
    ax = sns.countplot(x=i, hue='Churn', data = df_cat)
    legend_labels, _= ax.get_legend_handles_labels()
    ax.legend(legend_labels, ['Churn No','Churn Yes'])
    ax.set_xticklabels(('No', 'Yes'))
    count+=1
plt.show();
```

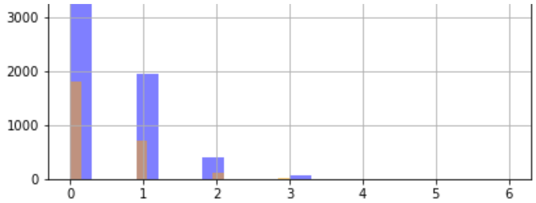

Churn No
Churn Yes
count
No
Yes
Area_Suburban
Area_Urban
Marital_Divorced
Marital_Married
Marital_Never Married
Marital_Separated
Marital_Widowed
Gender_Female
Gender_Male
Gender_Nonbinary
Contract_Month-to-month
Contract_One year
Contract_Two Year
InternetService_DSL
InternetService_Fiber Optic
InternetService_None
PaymentMethod_Bank Transfer(automatic)
PaymentMethod_Credit Card (automatic)
PaymentMethod_Electronic Check
PaymentMethod_Mailed Check
Churn

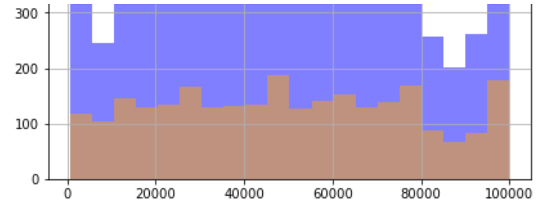
In [116]:
```
# Display plots for all remaining discrete value variables by Churn

# Plot split bar charts for binary variables by Churn
df_dis = df[['Children', 'Contacts', 'Yearly_equip_failure', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Ite
m6', 'Item7',
             'Item8', 'Churn']]
count=1
plt.subplots(figsize=(20, 80))
for i in df_dis.columns:
    plt.subplot(20,3,count)
    ax = sns.countplot(x = i, hue = 'Churn', data = df_dis)
    legend_labels, _= ax.get_legend_handles_labels()
    ax.legend(legend_labels, ['Churn No','Churn Yes'])

    count+=1
plt.show();
```



In [117]:
```
# Display histograms for continuous variables by Churn

# Array of columns to use
numerical_features = ['Zip', 'Lat', 'Lng', 'Population', 'Age', 'Income', 'Outage_sec_perweek', 'Email',
                      'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year']

# Labels to use in the legend for each line
sub_labels = ["No", "Yes"]

# Create split histogram by Churn for each column (ref 4)
fig, ax = plt.subplots(6, 2, figsize=(15, 30))
no = df[df.Churn == 0][numerical_features].hist(bins=20, color="blue", alpha=0.5, ax=ax, label='Churn No')
yes = df[df.Churn == 1][numerical_features].hist(bins=20, color="orange", alpha=0.5, ax=ax, label='Churn Yes')

# Create legend
fig.legend([no, yes], labels=sub_labels, loc="center left", borderaxespad=0.1, title="Churn")
plt.subplots_adjust(left=0.1)
plt.show();
```


Age
Bandwidth_GB_Year
Email
Income
Lat
Lng
Churn
No
Yes
MonthlyCharge
Outage_sec_perweek
Population
Tenure
Yearly_equip_failure
Zip





Prepared Data

In [32]:
```
# Save cleaned dataframe to CSV
df.to_csv('churn_clean_data_final.csv', index = False, encoding = 'utf-8')
```

Initial Model

In [118]:
```
# Set up input matrix and response variable
Xinit = df.drop('Churn', axis = 1)
y = df['Churn'].values
```

In [119]:
```
# Veiw number of independent variables
print ("There are", Xinit.shape[1], "independent variables in the initial model.")
```

```
There are 55 independent variables in the initial model.
```

In [120]:
```
# Add y-intercept, create model, and view summary
Xcinit = sm.add_constant(Xinit)
logistic_regression = sm.Logit(y,Xcinit)
fitted_model1 = logistic_regression.fit()
fitted_model1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.217085
         Iterations 9
```

Out[120]:

Logit Regression Results

Dep. Variable:	y	**No. Observations:**	10000
Model:	Logit	**Df Residuals:**	9950
Method:	MLE	**Df Model:**	49
Date:	Sat, 22 May 2021	**Pseudo R-squ.:**	0.6246
Time:	16:44:07	**Log-Likelihood:**	-2170.9
converged:	True	**LL-Null:**	-5782.2
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-4.2390	1.25e+06	-3.4e-06	1.000	-2.45e+06	2.45e+06
Zip	3.632e-07	3.35e-06	0.109	0.914	-6.2e-06	6.92e-06
Lat	0.0035	0.008	0.461	0.645	-0.011	0.018
Lng	-0.0013	0.006	-0.211	0.833	-0.013	0.011
Population	-1.473e-07	2.8e-06	-0.053	0.958	-5.64e-06	5.35e-06
Children	-0.0055	0.137	-0.040	0.968	-0.274	0.263
Age	0.0028	0.015	0.194	0.846	-0.026	0.032
Income	3.996e-07	1.38e-06	0.290	0.772	-2.3e-06	3.1e-06
Outage_sec_perweek	-0.0027	0.013	-0.207	0.836	-0.028	0.023
Email	-0.0094	0.013	-0.739	0.460	-0.034	0.016
Contacts	0.0626	0.039	1.603	0.109	-0.014	0.139
Yearly_equip_failure	-0.0349	0.061	-0.570	0.569	-0.155	0.085
Techie	1.0981	0.103	10.673	0.000	0.896	1.300
Port_modem	0.1432	0.077	1.849	0.064	-0.009	0.295
Tablet	-0.0509	0.085	-0.602	0.547	-0.217	0.115
Phone	-0.2988	0.133	-2.246	0.025	-0.560	-0.038
Multiple	0.3602	0.202	1.787	0.074	-0.035	0.755
OnlineSecurity	-0.2935	0.312	-0.942	0.346	-0.905	0.317
OnlineBackup	-0.1173	0.181	-0.649	0.516	-0.471	0.237
DeviceProtection	-0.1085	0.234	-0.465	0.642	-0.566	0.349
TechSupport	-0.2144	0.173	-1.237	0.216	-0.554	0.125
StreamingTV	1.1137	0.510	2.184	0.029	0.114	2.113
StreamingMovies	1.2743	0.363	3.506	0.000	0.562	1.987
PaperlessBilling	0.1659	0.079	2.105	0.035	0.011	0.320
Tenure	-0.1687	0.363	-0.465	0.642	-0.880	0.542
MonthlyCharge	0.0394	0.014	2.850	0.004	0.012	0.067
Bandwidth_GB_Year	0.0006	0.004	0.143	0.886	-0.008	0.009
Item1	-0.0189	0.055	-0.346	0.730	-0.126	0.088
Item2	-0.0041	0.052	-0.078	0.938	-0.106	0.098
Item3	0.0206	0.047	0.438	0.662	-0.072	0.113
Item4	-0.0301	0.042	-0.716	0.474	-0.113	0.052
Item5	-0.0362	0.044	-0.816	0.415	-0.123	0.051
Item6	-0.0193	0.045	-0.427	0.670	-0.108	0.069
Item7	-0.0025	0.043	-0.058	0.954	-0.087	0.082
Item8	-0.0140	0.040	-0.347	0.729	-0.093	0.065
Area_Rural	-0.6808	nan	nan	nan	nan	nan
Area_Suburban	-0.7289	nan	nan	nan	nan	nan
Area_Urban	-0.6294	nan	nan	nan	nan	nan
Marital_Divorced	-0.3992	4.25e+06	-9.38e-08	1.000	-8.34e+06	8.34e+06
Marital_Married	-0.2942	4.47e+06	-6.59e-08	1.000	-8.75e+06	8.75e+06

Marital_Never Married	-0.3854	4.18e+06	-9.22e-08	1.000	-8.19e+06	8.19e+06
Marital_Separated	-0.2808	4.11e+06	-6.84e-08	1.000	-8.05e+06	8.05e+06
Marital_Widowed	-0.1387	4.2e+06	-3.3e-08	1.000	-8.23e+06	8.23e+06
Gender_Female	-0.4715	nan	nan	nan	nan	nan
Gender_Male	-0.2441	nan	nan	nan	nan	nan
Gender_Nonbinary	-0.5495	nan	nan	nan	nan	nan
Contract_Month-to-month	1.5289	1.98e+05	7.71e-06	1.000	-3.89e+05	3.89e+05
Contract_One year	-1.8857	nan	nan	nan	nan	nan
Contract_Two Year	-1.9892	2.97e+05	-6.7e-06	1.000	-5.82e+05	5.82e+05
InternetService_DSL	-0.0934	nan	nan	nan	nan	nan
InternetService_Fiber Optic	-2.0076	nan	nan	nan	nan	nan
InternetService_None	-0.8181	nan	nan	nan	nan	nan
PaymentMethod_Bank Transfer(automatic)	-0.9996	3.42e+06	-2.92e-07	1.000	-6.71e+06	6.71e+06
PaymentMethod_Credit Card (automatic)	-0.7903	3.42e+06	-2.31e-07	1.000	-6.71e+06	6.71e+06
PaymentMethod_Electronic Check	-0.3688	3.42e+06	-1.08e-07	1.000	-6.71e+06	6.71e+06
PaymentMethod_Mailed Check	-0.7604	3.42e+06	-2.22e-07	1.000	-6.71e+06	6.71e+06

In [121]:
```
# View prediction
clf = LogisticRegression()
clf.fit(Xinit, y.astype(int))
y_clf = clf.predict(Xinit)
print(classification_report(y, y_clf))
```

```
              precision    recall  f1-score   support

           0       0.86      0.91      0.88      7350
           1       0.70      0.59      0.64      2650

    accuracy                           0.83     10000
   macro avg       0.78      0.75      0.76     10000
weighted avg       0.82      0.83      0.82     10000
```

Model Reduction

The model has many variables that may be causing the prediction to be inaccurate. There are several independent variables with high p-values that are insignificant to whether a customer is likely to churn and should be removed. The variables need to be investigated further to determine their relevance to the response variable. Also, the variables need to be investigated for instances of multicollinearity which could be affecting the results.

In [122]:
```
# Use recursive feature elimination to choose most important features (ref 6)
model = LogisticRegression()
rfe = RFE(model, 10)
rfe = rfe.fit(Xcinit, y)
print(rfe.support_)
print(rfe.ranking_)
f = rfe.get_support(1) # the most important features
Xfin = Xinit[Xinit.columns[f]] # final features`
```

```
[False False False False False False False False False False False False
  True False False False False False  True  True False  True  True False
 False False False False False False False False False False False False
 False False False False False False False False False False  True  True
  True  True False  True False False False False]
[16 45 29 39 46 36 41 47 31 27 34 22  1 18 15  6 35 13  1  1  5  1  1 14
 17 33 43 42 40 32 25 26 28 30 23 20 19 44 12 11 10 37 21  3 38  4  1  1
  1  1  2  1  7  8 24  9]
```

In [123]:
```python
# Look for evidence of Variance Inflation Factors (ref 7) causing multicollinearity

# VIF dataframe
vif_data = pd.DataFrame()
vif_data["feature"] = Xfin.columns

# calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(Xfin.values, i)
                          for i in range(len(Xfin.columns))]

print(vif_data)
```

```
                                feature       VIF
0                            Port_modem  1.764681
1                      DeviceProtection  1.658301
2                           TechSupport  1.505314
3                       StreamingMovies  1.781131
4                      PaperlessBilling  2.113488
5                     Contract_One year  1.310758
6                     Contract_Two Year  1.374253
7                   InternetService_DSL  1.895727
8           InternetService_Fiber Optic  2.144528
9  PaymentMethod_Bank Transfer(automatic)  1.256729
```

- All variable VIF scores are below the recommended score of 5. There is no evidence for molticollinearity.

In [124]:
```python
# Re-run the model
Xcfin = sm.add_constant(Xfin)
logistic_regression = sm.Logit(y,Xcfin)
fitted_model2 = logistic_regression.fit()
fitted_model2.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.487089
         Iterations 6
```

Out[124]:

Logit Regression Results

Dep. Variable:	y	**No. Observations:**	10000
Model:	Logit	**Df Residuals:**	9989
Method:	MLE	**Df Model:**	10
Date:	Sat, 22 May 2021	**Pseudo R-squ.:**	0.1576
Time:	16:45:22	**Log-Likelihood:**	-4870.9
converged:	True	**LL-Null:**	-5782.2
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.7484	0.084	-20.762	0.000	-1.913	-1.583
Port_modem	0.0209	0.050	0.417	0.676	-0.077	0.119
DeviceProtection	0.2876	0.050	5.729	0.000	0.189	0.386
TechSupport	0.1252	0.051	2.433	0.015	0.024	0.226
StreamingMovies	1.5229	0.053	28.820	0.000	1.419	1.626
PaperlessBilling	0.0566	0.051	1.112	0.266	-0.043	0.156
Contract_One year	-1.3906	0.071	-19.464	0.000	-1.531	-1.251
Contract_Two Year	-1.5609	0.070	-22.159	0.000	-1.699	-1.423
InternetService_DSL	0.5560	0.069	8.020	0.000	0.420	0.692
InternetService_Fiber Optic	0.0194	0.068	0.284	0.776	-0.114	0.153
PaymentMethod_Bank Transfer(automatic)	-0.0977	0.061	-1.607	0.108	-0.217	0.021

In [125]:
```
# Remove features with high p-value (above 0.5) and save dataframe for final reduced model
drops = ['Port_modem', 'PaperlessBilling', 'InternetService_Fiber Optic','PaymentMethod_Bank Transfer(automati
c)']
Xfin = Xfin.drop(drops, axis = 1)
```

In [126]:
```
# Split the data to be used in final model evaluation
X_train, X_test, y_train, y_test = train_test_split(Xfin, y.astype(float), test_size=0.33, random_state=101)
```

In [127]:
```
Xcfin = sm.add_constant(X_train)
logistic_regression = sm.Logit(y_train,Xcfin)
fitted_model2 = logistic_regression.fit()
fitted_model2.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.494973
         Iterations 6
```

Out[127]:

Logit Regression Results

Dep. Variable:	y	**No. Observations:**	6700
Model:	Logit	**Df Residuals:**	6693
Method:	MLE	**Df Model:**	6
Date:	Sat, 22 May 2021	**Pseudo R-squ.:**	0.1494
Time:	16:46:45	**Log-Likelihood:**	-3316.3
converged:	True	**LL-Null:**	-3898.9
Covariance Type:	nonrobust	**LLR p-value:**	1.745e-248

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.6457	0.069	-23.778	0.000	-1.781	-1.510
DeviceProtection	0.2516	0.061	4.142	0.000	0.133	0.371
TechSupport	0.1384	0.062	2.227	0.026	0.017	0.260
StreamingMovies	1.4628	0.064	23.031	0.000	1.338	1.587
Contract_One year	-1.3616	0.086	-15.774	0.000	-1.531	-1.192
Contract_Two Year	-1.5147	0.085	-17.850	0.000	-1.681	-1.348
InternetService_DSL	0.5254	0.062	8.411	0.000	0.403	0.648

In [128]:
```
# View prediction
clf = LogisticRegression()
clf.fit(X_train, y_train.astype(int))
y_clf = clf.predict(X_test)
print(classification_report(y_test, y_clf))
```

```
              precision    recall  f1-score   support

         0.0       0.81      0.90      0.85      2450
         1.0       0.56      0.38      0.45       850

    accuracy                           0.76      3300
   macro avg       0.68      0.64      0.65      3300
weighted avg       0.74      0.76      0.75      3300
```

In [129]:
```
print ("There are", Xfin.shape[1], "independent variables in the final model.")
```

```
There are 6 independent variables in the final model.
```

Model Comparison

I first identified the ten features which were most important to the calculation using recursive feature elimination. This method uses model accuracy to identify which combination of attributes most contribute to predicting the target variable (Brownlee, 2020). I then looked for evidence of multicollinearity using a test for Variance Inflation Factors. Finally taking a look at the reduced model, I chose to remove all remaining variables which had p-scores above the 0.05 threshold. This final step ensured that all remaining independent variables were significant. I began with 55 independent variables, which was then reduced to six remaining variables.

My initial model had better scores for the target descriptive statistics than the reduced model. The initial model would predict churn correctly 70% of the time, while the reduced model was 36%. The initial model was likely to catch actual cases of churn 59% of the time, while the reduced model was only 38% of the time. The combined F1 score was also higher on the initial model at 64% compared to 45% on the reduced model. Finally, the overall accuracy of the initial model was also slightly higher at 83% compared to the reduced model at 76%.

In [130]:
```
# View prediction (initial)
clf = LogisticRegression()
clf.fit(Xinit, y.astype(int))
y_clf = clf.predict(Xinit)
print(classification_report(y, y_clf))
```

```
              precision    recall  f1-score   support

           0       0.86      0.91      0.88      7350
           1       0.70      0.59      0.64      2650

    accuracy                           0.83     10000
   macro avg       0.78      0.75      0.76     10000
weighted avg       0.82      0.83      0.82     10000
```

In [131]:
```
# View prediction (reduced)
clf = LogisticRegression()
clf.fit(X_train, y_train.astype(int))
y_clf = clf.predict(X_test)
print(classification_report(y_test, y_clf))
```

```
              precision    recall  f1-score   support

         0.0       0.81      0.90      0.85      2450
         1.0       0.56      0.38      0.45       850

    accuracy                           0.76      3300
   macro avg       0.68      0.64      0.65      3300
weighted avg       0.74      0.76      0.75      3300
```

Results

The equation of the final regression model is: y = -1.64 + 0.25(DeviceProtection) + 0.14(TechSupport) + 1.45(StreamingMovies) - 1.35(Contract_One year) - 1.5(Contract_Two Year) + 0.52(InternetService_DSL). In this equation, -1.64 is the intercept value where Churn would reside on the y-axis when all of the predictor variables are zero. Each coefficient of the predictor variables describes how much the target variable is estimated to change if all other variables remain constant. For example, if DeviceProtection increased by one unit, the mean value of Churn would increase by 0.25. The Contract_Two Year seems to have the largest effect on churn negatively by 1.5, but most variables seem to have a similar effect.

The model does not give me a lot of confidence in its ability to accurately predict churn. Precision and recall seem to be much higher when trying to predict if a customer will not churn. I think the high scores for those metrics are inflating the accuracy score of the model, making it look like 77% of the time it is correct. Practically with low coefficients, nothing really stands out as one or two areas a company can focus on in order to reduce customer churn, as all of the coefficients of the predictor variables have a range of 0.5. This model is definitely limited by my current skills. I am still trying to understand what is necessary to produce optimal results, and I imagine that by the end of my course work with WGU, I would be able to produce an optimal model.

In [132]:
```
# Create dataframe of coefficients for regression equation (ref H8)
clf = LogisticRegression()
clf.fit(X_train, y_train.astype(int))
coeff_df = pd.DataFrame(zip(X_train.columns, np.transpose(clf.coef_.round(decimals = 2))), columns=['features'
, 'coef'])

# Print regression equation
equation = 'y = '
y1 = clf.intercept_.round(decimals = 2)
equation += str(y1)
for index, row in coeff_df.iterrows():
    if row['coef'] < 0:
        c = str(row.coef)
        equation += c + '(' + row['features'] + ')'
    else:
        c = str(row.coef)
        equation += '+' + c + '(' + row['features'] + ')'
# Remove brackets around coefficients
brackets = ['[', ']']
for i in brackets :
    equation = equation.replace(i, '')

print(equation)
```

```
y = -1.64+0.25(DeviceProtection)+0.14(TechSupport)+1.45(StreamingMovies)-1.35(Contract_One year)-1.5(Contract
_Two Year)+0.52(InternetService_DSL)
```

Sources

- Brownlee, J. (2020, August 27). Feature Selection For Machine Learning in Python. Retrieved January 12, 2021, from https://machinelearningmastery.com/feature-selection-machine-learning-python/ (https://machinelearningmastery.com/feature-selection-machine-learning-python/)
- Gahn, K. (2020). D208 Gahn Task 1 [Unpublished]. Western Governors University.
- Massaron, L., & Boschetti, A. (2016). Regression analysis with Python. Packt Publishing. ISBN: 9781785286315
- Zach. (2020, October 13). The 6 Assumptions of Logistic Regression (With Examples). Retrieved January 12, 2021, from https://www.statology.org/assumptions-of-logistic-regression (https://www.statology.org/assumptions-of-logistic-regression)

Helpful Sites Used in Coding Project

1. https://stackoverflow.com/questions/51672709/converting-no-and-yes-into-0-and-1-in-pandas-dataframe/51672855 (https://stackoverflow.com/questions/51672709/converting-no-and-yes-into-0-and-1-in-pandas-dataframe/51672855)
2. https://stackoverflow.com/questions/61526812/boxplot-for-all-data-in-dataframe-error-numpy-ndarray-object-has-no-attribut (https://stackoverflow.com/questions/61526812/boxplot-for-all-data-in-dataframe-error-numpy-ndarray-object-has-no-attribut)
3. https://datascience.stackexchange.com/questions/84840/how-to-create-multiple-subplots-scatterplot-in-for-loop (https://datascience.stackexchange.com/questions/84840/how-to-create-multiple-subplots-scatterplot-in-for-loop)
4. https://randerson112358.medium.com/predict-customer-churn-using-python-machine-learning-b92f39685f4c (https://randerson112358.medium.com/predict-customer-churn-using-python-machine-learning-b92f39685f4c)
5. https://riptutorial.com/matplotlib/example/10473/single-legend-shared-across-multiple-subplots (https://riptutorial.com/matplotlib/example/10473/single-legend-shared-across-multiple-subplots)
6. https://stackoverflow.com/questions/47730328/extract-optimal-features-from-recursive-feature-elimination-rfe (https://stackoverflow.com/questions/47730328/extract-optimal-features-from-recursive-feature-elimination-rfe).
7. https://analyticsvidhya.com/blog/2020/03/what-is-multicollinearity/ (https://analyticsvidhya.com/blog/2020/03/what-is-multicollinearity/)
8. https://stackoverflow.com/questions/57924484/finding-coefficients-for-logistic-regression-in-python (https://stackoverflow.com/questions/57924484/finding-coefficients-for-logistic-regression-in-python)