WGU D208 TASK 1 REV 8 - MATTINSON

Multiple Regression Using Churn Data

Mike Mattinson

Master of Science, Data Analytics, WGU.edu

D208: Predictive Modeling

Dr. Keiona Middleton

September 21, 2021

Abstract. This paper provides the results of a multiple regression analysis conducted on a customer dataset in partial fulfillment of WGU's D208 Predictive Analysis class requirements. The dataset represents 10,000 rows of customer data for a typical services company. There are fifty (50) attributes for each customer. The provided dataset was mostly clean and ready to use, however, some few additional data cleaning steps were completed prior to running the predictive analysis. The predictive analysis includes both an initial model using all the predictor variables and a final model using a reduced set of predictor variables. The final model includes both numerical and categorical predictor variables. P-values and multi-collinearity were used to select the features used in the final model. The principal research question "how to predict customer monthly charge with high confidence using as few predictor variables as possible" was determined (R-squared = 94.8%) using fifteen (15) of the original attributes. The analysis was conducted in a Python environment using a Jupyter notebook. The Jupyter notebook includes both code and discussion of the analysis. Key words: Churn. Regression. Linear Regression. Multiple Regression. Primary data set: clean_churn.csv, the initial set has 10,000 records with 50 attributes.

Custom Styles. In order for custom styles to be applied to this notebook, a file called "d208.css" is created within the styles subfolder of the Python project. I am including the contents of that file here for reference, it will be visible in the .ipynb file as well as the .pdf file:

```
<style>

body {
    counter-reset: part-counter 0;
}

h1 {
    margin: 0 0 0 0;
    font-family: 'Times New Roman', serif;
    font-size: 40px;
    padding: 5px;
    text-transform: uppercase;
    letter-spacing: 5px;
    color: #0000ff;
}

.part {
    margin: 0 0 0 0;
    font-family: 'Impact';
    font-size: 20px;
    padding: 5px;
    text-transform: uppercase;
    letter-spacing: 5px;
    color: #000000;
    background: inherit;
}

.part:before {
    counter-increment: part-counter;
    content: "Part " counter(part-counter, upper-roman)": ";
}

h2 {
    margin: 0 0 0 0;
    font-family: 'Times New Roman', serif;
    font-size: 36px;
    padding: 5px;
    text-transform: uppercase;
    letter-spacing: 5px;
    color: #0000ff;
    background: inherit;

}

h2:before {
    content: attr(data-nbr)". ";
}


h3 {
    background: #E6EFFA;
```

```
    font-family: 'Impact';
    font-size: 100%;
    text-align: center;
    text-transform: uppercase;
    padding: 5px 0;

}

.title {
    text-align: center;
    line-height: 48px;
    font-size: 20px;
}

.quote {
    padding: 10px;
    border: 1px groove gray;
    background-color: rgb(202, 197, 198);
}

.impact {
    font-family: 'Courier New';
    font-size: 18px;
    line-height: 24px;
}

.impact:before {
    content: attr(data-hdr)". ";
    font-family: 'Impact';
}

p {
    font-family: 'Courier New';
    font-size: 18px;
    color: #000000;
    line-height: 24px;
}

ul.a {
    list-style-position: outside;
    list-style-type: upper-alpha;
    line-height: 2.0;
  }

.apa {
    padding-left: 4em;
    text-indent: -4em;
    background: powderblue;
    font-size: 18px;
    line-height: 2.0;
}

.apa:before {
    content: attr(data-author) " (" attr(data-date) "). ";
}

.apa:after {
    content: ". Retrieved from: " attr(data-url);
}
```

```
</style>
```

In [39]:
```
# Styling notebook with custom css
import os
s = os.path.join('styles','d208.css')
print('custom styles are found in {}'.format(s))
from IPython.core.display import HTML
HTML(open(s, "r").read())
```

```
custom styles are found in styles\d208.css
```

Out[39]:

PART I: RESEARCH QUESTION

A1. RESEARCH QUESTION

```
<div class="impact" data-hdr="Primary Research Question">
    A typical services company’s revenue is maximized based on the total number of customers and how much each of those customers pay for those services. If the company charges too much, then the customer may stop the service, this is known as churn. If the company charges too little, then it will not maximize its revenue. This analysis will attempt to predict a customer’s monthly payment (dependent variable is ‘MonthlyCharge’) using multiple regression with high degree of accuracy (R-squared >= 95%) based on a minimum set of predictor variables. The final set of predictor variables should include both numeric (e.g., Tenure, Child, and Income, etc.) and categorical data (e.g., Techie, Gender, and Internet Service type, etc.).
    </div>
```

A2. OBJECTIVES AND GOALS

Data Preparation. Data Preparation objectives are addressed in Part III below and include the following:

A. Convert categorical data.

B. Mitigate missing data.

C. Select data required for the analysis.

D. Remove data deemed unneccesary.

E. Explore data.

F. Visualize data.

G. Provide copy of final data.

Model Analysis. Model Analysis objectives are addressed in Part IV below and include the following:

A. Eliminate predictor variables with high p-values.

B. Eliminate predictor variables with high degree of multicollinearity.

C. Create initial model using all the data.

D. Refine model using a reduced set of the data.

E. Summarize results.

F. Ensure independent and dependent variables are linear.

G. Ensure independent variables are not highly collinear

H. Ensure final model residuals are normally distributed.

PART II: METHOD JUSTIFICATION

B. Describe multiple regression methods by doing the following:

B1. ASSUMPTIONS

1. Summarize the assumptions of a multiple regression model.

Assumptions. The multiple regression analysis is based on the following assumptions:

A. **Linear Relationship**. Linear relationship between dependent and independent variables.

B. **Multivariate Normality**. Residuals are normally distributed.

C. **Multicollinearity**. The independent or predictor variables are not highly correlated with each other.

B2. BENEFITS OF PYTHON

2. Describe the benefits of using the tool(s) you have chosen (i.e., Python, R, or both) in support of various phases of the analysis.

Python-Jupyter. The analysis is completed by executing Python code inside of a Jupyter notebook. Python is installed within VS

Code IDE. VS Code is used to manage the overall environment and Jupyter notebook is used to execute Python code and discuss and highlight the analysis process. Within the Python project, a virtual environment is set up to include the packages and configurations unique to this assignment.

Here are the versions of Python and VS Code:

A. Python 3.9.6

B. VS Code 1.60.0

Benefits. Using Python has the following benefits:

A. Ease of use

B. Availability of required statistical and modeling packages

C. User friendly Jupyter notebook allows segmented code execution and ability to include additional markup with the code.

D. Able to use custom .css styles to format html output

In [2]:
```
# show python environment
import sys
print(sys.version)
print(sys.executable)
```

```
3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)]
p:\code_wgu\5\v\scripts\python.exe
```

Note. It is possible to list all of the currently installed Python packages using [>pip list] from the terminal in the VS Code IDE. Here is an example of that output:

Package Version

======================

backcall 0.2.0

colorama 0.4.4

cycler 0.10.0

debugpy 1.4.3

decorator 5.1.0

entrypoints 0.3

ipykernel 6.4.1

ipython 7.27.0

ipython-genutils 0.2.0

jedi 0.18.0

joblib 1.0.1

jupyter-client 7.0.2

```
jupyter-core 4.7.1
kiwisolver 1.3.2
matplotlib 3.4.3
matplotlib-inline 0.1.3
...
```

B3. WHY MULTIPLE REGRESSION IS APPROPRIATE

> **3.** Explain why multiple regression is an appropriate technique to analyze the research question summarized in Part I.

Why. Multiple regression is “the most common” tool for conducting regression analysis. According to Petchko (2018), multiple regression allows the researcher to asses the strength of the relationship between the dependent and several predictor variables as well as to determine the importance of each predictor to the relationship.

PART III: EXPLORATORY DATA ANALYSIS

C1. DESCRIBE DATA ANALYSIS PREP AND EXPLORE

> **1.** Describe your data preparation goals and the data manipulations that will be used to achieve the goals.

Select Data. From the original data, determine which attributes fit the best for the primary research question. Load the data from the provided .csv file as a pandas dataframe.

Mitigate Missing Data. Look through data for missing rows or columns. Also, look for Null or NaN values. If found, decide how best to mitigate the issue.

Remove Data. Once data is determined not to be of value to the analysis, use the pandas .drop() method to remove the data.

Convert Categorical Data. In order to use categorical data in the regression model, each variable must be converted into numeric dummy data. I will use pandas .get_dummies() method. This will generate new numeric variables based on the unique values and this will also remove the original attribute.

Explore Data. Explore customer data by calculating traditional statistics. Look for patterns and relationships between attributes. If possible, create visualizations to add in the exploratory process.

Visualize Data. Continue to explore data and their relationships using histogram, countplots, barplots and scatter plot diagrams. Use matplotlib and sns packages to generate these univariate and bivariate diagrams.

C2-C4. PREPARE AND EXPLORE DATA

2-4. 2. Discuss the summary statistics, including the target variable and all predictor variables that you will need to gather from the data set to answer the research question. 3. Explain the steps used to prepare the data for the analysis, including the annotated code. 4. Generate univariate and bivariate visualizations of the distributions of variables in the cleaned data set. Include the target variable in your bivariate visualizations.

Imports. Before going further, let's import Python libraries that will be used throughout the notebook. In addition to standard plotting and modeling packages, I have created a few custom helper functions which are imported towards the end.

In [3]:
```
# imports
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.decomposition import PCA
import statsmodels.api as sm
import statsmodels.formula.api as smf
from IPython.core.display import HTML
from IPython.display import display
```

Helper Functions. Here are helper functions that will be used thoughout the notebook.

In [4]:
```
def get_redundant_pairs(df):
    '''Get diagonal and lower triangular pairs of correlation matrix'''
    pairs_to_drop = set()
    cols = df.columns
    for i in range(0, df.shape[1]):
        for j in range(0, i+1):
            pairs_to_drop.add((cols[i], cols[j]))
    return pairs_to_drop

def get_top_abs_correlations(df, n=5):
    au_corr = df.corr().abs().unstack()
    labels_to_drop = get_redundant_pairs(df)
    au_corr = au_corr.drop(labels=labels_to_drop).sort_values(ascending=False)
    return au_corr[0:n]

def custom_corr_matrix(df, title):
    fig = plt.figure(figsize=(30, 30))
    sns.set(font_scale=1.0)
    sns.heatmap(data=df.corr().round(1), annot=True,annot_kws={'size':30})
    print(get_top_abs_correlations(df))
    plt.savefig('output/' + COURSE + '/fig_corr_matrix_' + title + '.png', facecolor='w')
```

Constants. Here are a couple of global variables that will be reused thoughout the notebook.

In [5]:
```
# constants
COURSE = 'd208' # name of course to be added to filename of generated figures and tables.
target = 'MonthlyCharge' # this is the column name of the primary research column
```

Select Data. The customer dataset as a .csv file is loaded into Python as a Pandas dataframe using the .read_csv() method. After the dataframe is created, I use the df.shape function to show number of rows and columns. To begin the analysis, I have selected to load all of the data from the .csv file.

In [6]:
```
# read csv file
import os
df = pd.read_csv(os.path.join('data','churn_clean.csv'), header=0)
df.shape
```

Out[6]:
```
(10000, 50)
```

There are 10,000 customer records with fifty (50) attributes for each customer.

Mitigate Missing Data. Use .info() and .isna().any() methods to view a summary of possible missing data. I do not expect to find any missing data as the dataset provided has already been cleaned.

In [7]:
```
# explore missing data
missing = df[df.columns[df.isna().any()]].columns
df_missing = df[missing]
print(df_missing.info())
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Empty DataFrameNone
<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
None
```

Analysis of the raw data shows no missing data, each attribute has 10,000 non-null values.

Duplicate Data. Look for duplicate data in rows and columns. This dataset had been provided to this assignment in a very clean, ready state, so I don't expect to find anything here.

In [8]:
```
# look for duplicate data - looking for zero rows
df[df.duplicated()]
```

Out[8]:

CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	...	MonthlyCharge	Bandwidth

0 rows × 50 columns

In [9]:
```
# check if any cols are duplicated - Looking for False
df.columns.duplicated().any()
```

Out[9]: False

In [10]:
```
# check if any rows are duplicated - looing for False
df.duplicated().any()
```

Out[10]: False

Remove Data. Identify columns that are not needed for the analysis and then use the .drop() methode to remove the data. Looking at the data, I select some of the demographic data, customer identification data and the survey data to be removed.

In [11]:
```python
# drop unwanted data
cols_to_be_removed = ['City','County','Zip','Job','TimeZone', 'State', 'Churn',
            'Lat', 'Lng', 'UID', 'Customer_id','Interaction', 'CaseOrder',
            'Item1','Item2','Item3','Item4','Item5','Item6','Item7','Item8']

# print list of dropped data
print('data to be removed: {}'.format(cols_to_be_removed))

# loop through list, if in current df, drop col
for c in cols_to_be_removed:
    if c in df.columns:
        df.drop(columns = c, inplace=True)
        print('Data named [{}] has been removed.'.format(c))
```

```
data to be removed: ['City', 'County', 'Zip', 'Job', 'TimeZone', 'State', 'Churn', 'Lat',
'Lng', 'UID', 'Customer_id', 'Interaction', 'CaseOrder', 'Item1', 'Item2', 'Item3', 'Item
4', 'Item5', 'Item6', 'Item7', 'Item8']
Data named [City] has been removed.
Data named [County] has been removed.
Data named [Zip] has been removed.
Data named [Job] has been removed.
Data named [TimeZone] has been removed.
Data named [State] has been removed.
Data named [Churn] has been removed.
Data named [Lat] has been removed.
Data named [Lng] has been removed.
Data named [UID] has been removed.
Data named [Customer_id] has been removed.
Data named [Interaction] has been removed.
Data named [CaseOrder] has been removed.
Data named [Item1] has been removed.
Data named [Item2] has been removed.
Data named [Item3] has been removed.
Data named [Item4] has been removed.
Data named [Item5] has been removed.
Data named [Item6] has been removed.
Data named [Item7] has been removed.
Data named [Item8] has been removed.
```

Explore Data - Independent Variables. Excluding target data, here is the final list of predictor variables. For quick reference, numerical data includes brief traditional statistic data, and categorical data includes a list of unique values:

Note. Independent variables are sometimes called by different names, they are synonymous, they can be referred to as independent variables, predictor variables, input variables and sometimes, as features.

In [12]:
```python
# print out input variables
for c in df.loc[:, df.columns != target]:
    if df.dtypes[c] == "object":
        print('\n{} is categorical: {}.'.format(c,df[c].unique()))
    else:
        print('\n{} is numerical:'.format(c ))
        print('\trange = {} - {}'.format(df[c].min(),df[c].max()))
        print('\tmean = {:.2f} +/- {:.2f}'.format(df[c].mean(), df[c].std()))
```

```
Population is numerical:
	range = 0 - 111850
	mean = 9756.56 +/- 14432.70

Area is categorical: ['Urban' 'Suburban' 'Rural'].

Children is numerical:
	range = 0 - 10
	mean = 2.09 +/- 2.15

Age is numerical:
	range = 18 - 89
	mean = 53.08 +/- 20.70

Income is numerical:
	range = 348.67 - 258900.7
	mean = 39806.93 +/- 28199.92

Marital is categorical: ['Widowed' 'Married' 'Separated' 'Never Married' 'Divorced'].

Gender is categorical: ['Male' 'Female' 'Nonbinary'].

Outage_sec_perweek is numerical:
	range = 0.09974694 - 21.20723
	mean = 10.00 +/- 2.98

Email is numerical:
	range = 1 - 23
	mean = 12.02 +/- 3.03

Contacts is numerical:
	range = 0 - 7
	mean = 0.99 +/- 0.99

Yearly_equip_failure is numerical:
	range = 0 - 6
	mean = 0.40 +/- 0.64

Techie is categorical: ['No' 'Yes'].

Contract is categorical: ['One year' 'Month-to-month' 'Two Year'].

Port_modem is categorical: ['Yes' 'No'].

Tablet is categorical: ['Yes' 'No'].

InternetService is categorical: ['Fiber Optic' 'DSL' 'None'].
```

```
Phone is categorical: ['Yes' 'No'].

Multiple is categorical: ['No' 'Yes'].

OnlineSecurity is categorical: ['Yes' 'No'].

OnlineBackup is categorical: ['Yes' 'No'].

DeviceProtection is categorical: ['No' 'Yes'].

TechSupport is categorical: ['No' 'Yes'].

StreamingTV is categorical: ['No' 'Yes'].

StreamingMovies is categorical: ['Yes' 'No'].

PaperlessBilling is categorical: ['Yes' 'No'].

PaymentMethod is categorical: ['Credit Card (automatic)' 'Bank Transfer(automatic)' 'Mail
ed Check'
 'Electronic Check'].

Tenure is numerical:
        range = 1.00025934 - 71.99928
        mean = 34.53 +/- 26.44

Bandwidth_GB_Year is numerical:
        range = 155.5067148 - 7158.98153
        mean = 3392.34 +/- 2185.29
```

Numeric vs Cateogrical Data. The analysis will use the following variables to separate the numeric and categorical data.

In [13]:
```
# variable for numeric data
num_cols = df.select_dtypes(include="number").columns
print(num_cols)
```

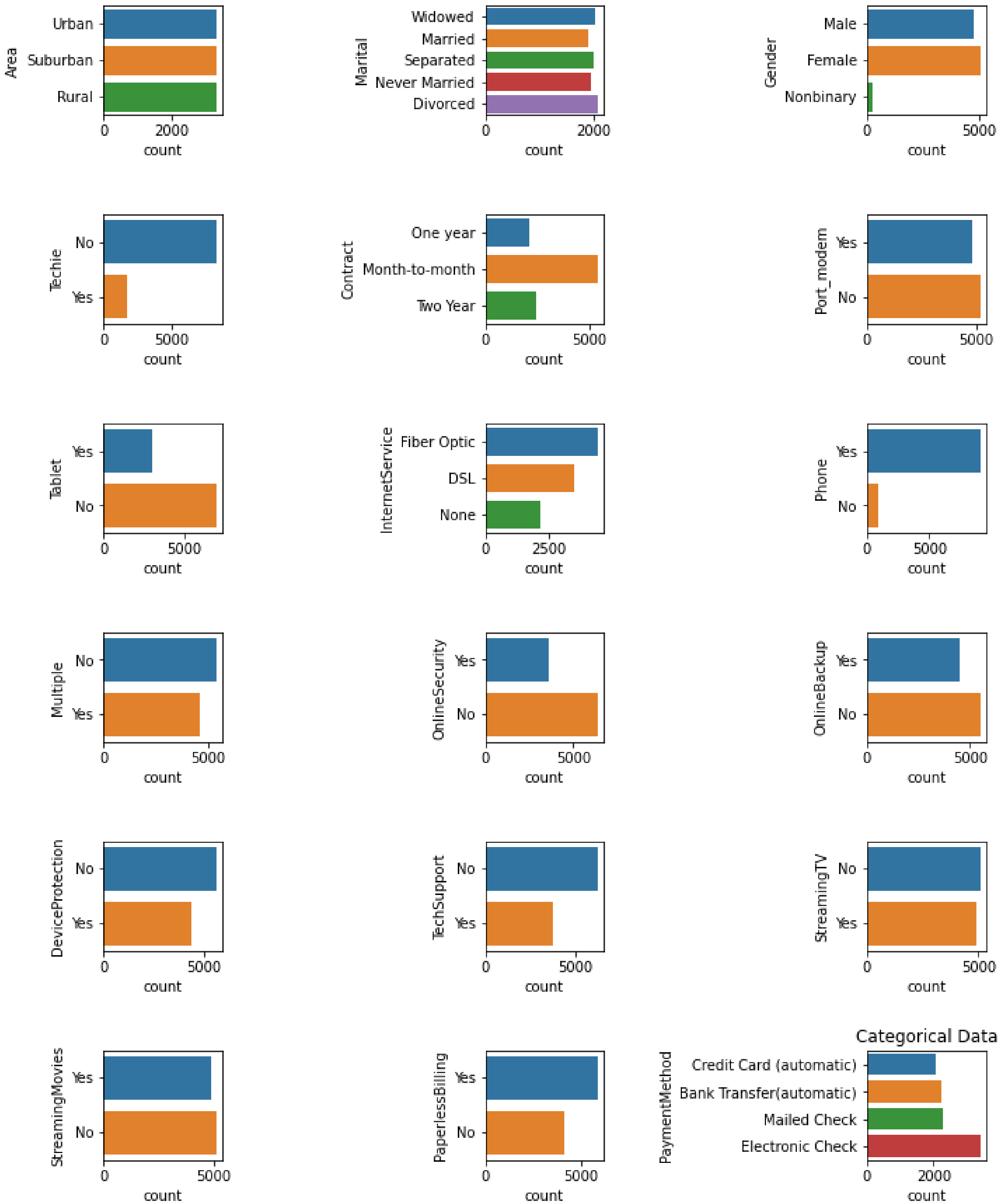
```
Index(['Population', 'Children', 'Age', 'Income', 'Outage_sec_perweek',
       'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
       'Bandwidth_GB_Year'],
      dtype='object')
```

In [14]:
```
# variable for categorical data
cat_cols = df.select_dtypes(include="object").columns
print(cat_cols)
```

```
Index(['Area', 'Marital', 'Gender', 'Techie', 'Contract', 'Port_modem',
       'Tablet', 'InternetService', 'Phone', 'Multiple', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'PaperlessBilling', 'PaymentMethod'],
      dtype='object')
```

Explore Categorical Data. Prior to converting the categorical data for use in the model, as part of exploratory data analysis, I will visualize the original categorical data using a countplot. In a moment, the categorical data will be converted to dummy data and I will lose the original data.

In [15]:
```
# plot categorical data - before it gets converted
fig = plt.figure(figsize=(10, 20))

for i, col in enumerate(cat_cols):
    plt.subplot(10, 3, i+1)
    ax = sns.countplot(y=col, data=df)
    fig.tight_layout(h_pad=4, w_pad=4)

plt.title('Categorical Data')
plt.savefig('output/' + COURSE + '/fig_countplot_categorical.png')
plt.show()
```



Convert Categorical Data. The regression model requires all of the independent variables to be numeric. Because there are many categorical data, each will have to be converted into numeric data. The data is converted into dummy numeric data using the pandas .get_dummies() method. After the conversion, the original data is removed.

Note. The method uses the option 'drop first=True'. Most of the categorical data has two or more unique values. When using this option, the .get_dummies() method will remove the first unique value, which is good, because of the multi-collinear nature of this operation. It can be a problem, however, if the data that is removed is data that is necessary. For the purpose of this analysis, I am using the 'drop_first' option, but future analysis may decide to use the other data. I am creating a variable called 'contract' with a snapshot of the contract data before the conversion in order to present an example of this potential problem.

In [16]:
```
# in a moment, I will generate a scatter plot of this data
contract = df[['MonthlyCharge','Contract']] # keep this for a future calculation
```

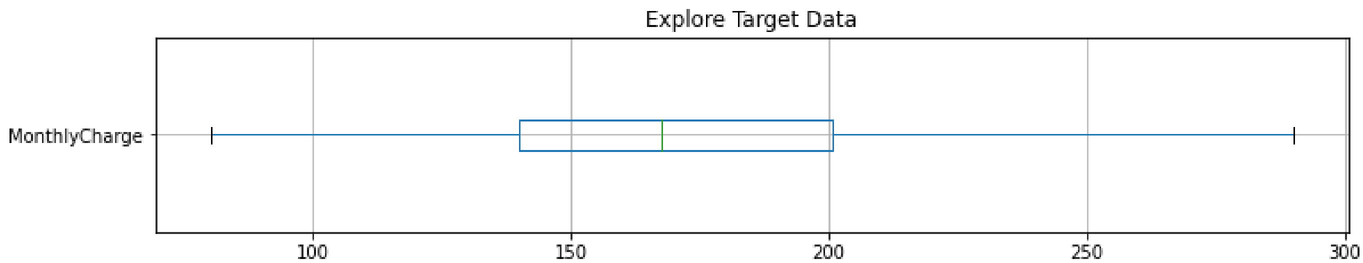
In [17]:
```
# convert categorical data
for c in cat_cols:
    if c in df.columns:
        df = pd.get_dummies(df, columns=cat_cols, drop_first=True)
        print(df.select_dtypes(include="uint8").columns)
```

```
Index(['Area_Suburban', 'Area_Urban', 'Marital_Married',
       'Marital_Never Married', 'Marital_Separated', 'Marital_Widowed',
       'Gender_Male', 'Gender_Nonbinary', 'Techie_Yes', 'Contract_One year',
       'Contract_Two Year', 'Port_modem_Yes', 'Tablet_Yes',
       'InternetService_Fiber Optic', 'InternetService_None', 'Phone_Yes',
       'Multiple_Yes', 'OnlineSecurity_Yes', 'OnlineBackup_Yes',
       'DeviceProtection_Yes', 'TechSupport_Yes', 'StreamingTV_Yes',
       'StreamingMovies_Yes', 'PaperlessBilling_Yes',
       'PaymentMethod_Credit Card (automatic)',
       'PaymentMethod_Electronic Check', 'PaymentMethod_Mailed Check'],
      dtype='object')
```

Explore Target Data. For this task, **MonthlyCharge** is the target (or dependent) variable. It is numeric data. The purpose of this regression model is to find the best set of predictor (independent) variables that can be used to predict a customer's **MonthlyCharge** to a high degree of accuracy. Target data will be described and visualized. To describe the numeric data, traditional statistics will be included with the boxplot.

A. **MonthlyCharge**. The amount charged to the customer monthly. This value reflects an average per customer. Here is a plot and description of the MonthlyCharge data:

In [18]:
```
# explore target data
plt.figure(figsize=(12, 2))
ax = df.boxplot([target], vert=False)
plt.title('Explore Target Data')
plt.savefig('output/' + COURSE + '/fig_boxplot_target.png', facecolor='w')
plt.show()
print(df[target].describe().round(3))
```




```
count    10000.000
mean       172.625
std         42.943
min         79.979
25%        139.979
50%        167.485
75%        200.735
max        290.160
Name: MonthlyCharge, dtype: float64
```

Explore Numeric Predictor Data. In addition to the categorical data shown above, the following numerical data will be included as independent variables in the analysis. Each numeric variable will be ploted using a boxplot. The data is standardized for the purpose of the boxplot. The data below the figure shows the traditional statistics of the non-standardized data. After the boxplots, I will also generate historam plots of the same data.

Numerical data:

A. **Children**: Number of children in customer's household as reported in sign up information
B. **Age**: Age of customer as reported in sign up information
C. **Income**: Annual income of cu stomer as reported at time of sign up
D. **Outage_sec_perweek**: Average number of seconds per week of system outage s in the customer's neighborhood
E. **Email**: Number of emails sent to the customer in the last year (marketing or correspondence)
F. **Contacts**: Number of times customer contacted technical support

G. **Population**: Population within a mile radius of customer, based on census data

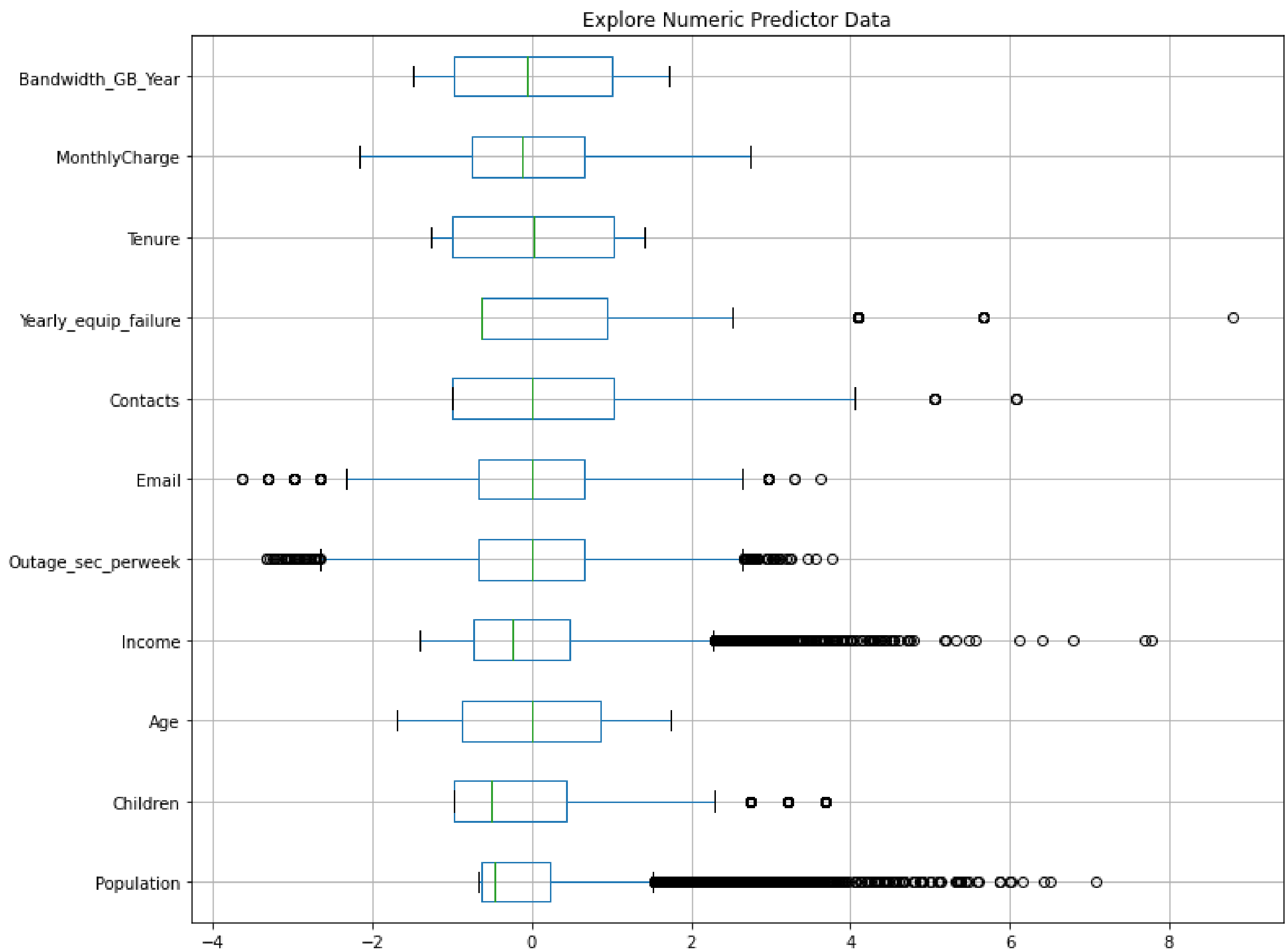
H. **Yearly_equip_failure**: The number of times customer's equipment failed and had to be reset/replaced in the past year

I. **Tenure**: Number of months the customer has stayed with the provider

J. **Bandwidth_GB_Year**: The average amount of data used, in GB, in a year by the customer

Univariate Boxplot of Numeric Predictor Data. Here are the box plots.

In [19]:
```
# explore numeric predictor data
plt.figure(figsize=(12, 10))
std_numeric_data = (df[num_cols] -df[num_cols].mean()) / df[num_cols].std()
ax = std_numeric_data.boxplot(vert=False)
plt.title('Explore Numeric Predictor Data')
plt.savefig('output/' + COURSE + '/fig_boxplot_numeric.png', facecolor='w')
plt.show()
#print(std_numeric_data.describe(percentiles=None).round(3).T)
print(df[num_cols].describe(percentiles=None).round(3).T)
```



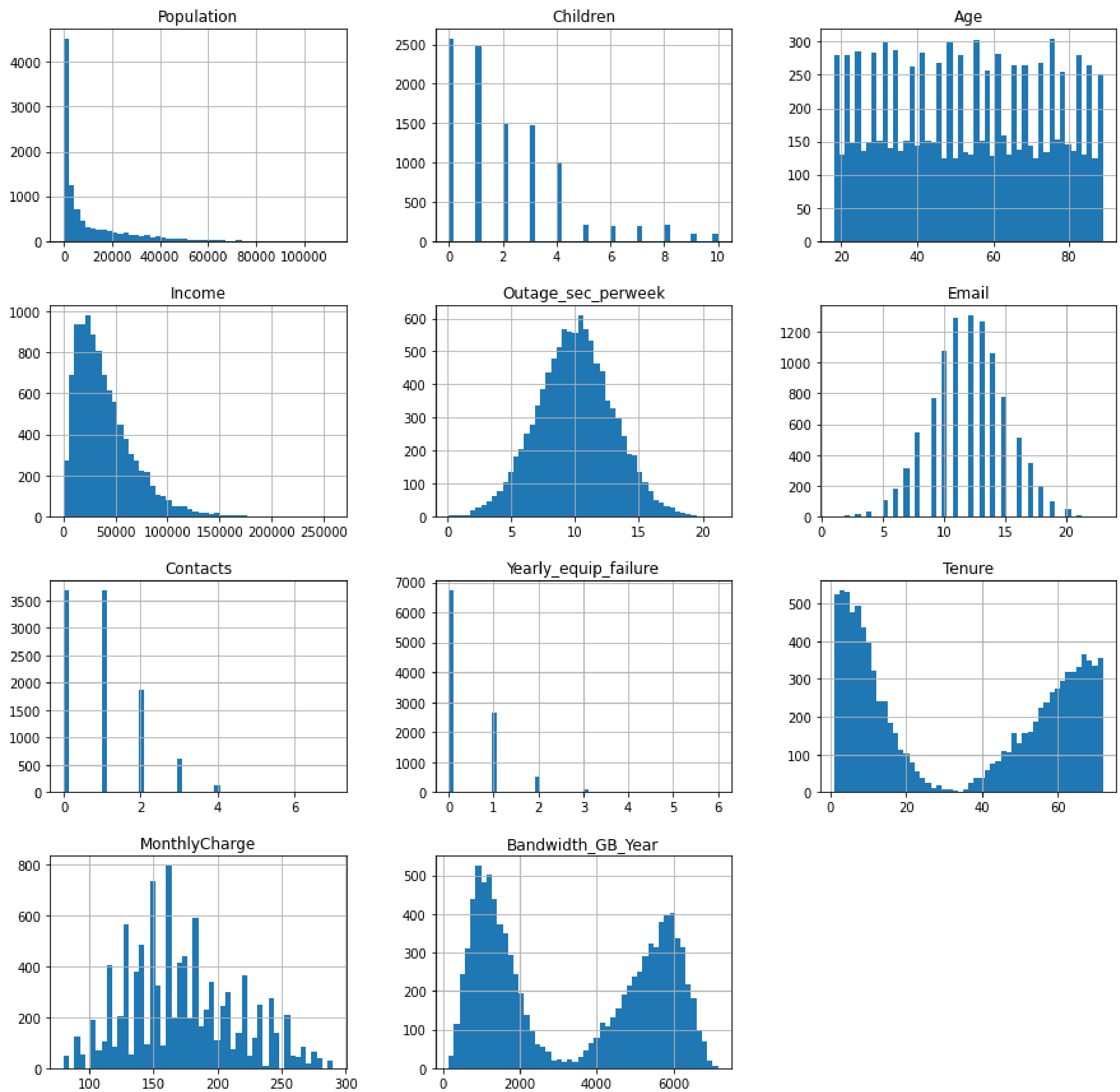

```
                        count       mean        std      min        25%  \
Population            10000.0   9756.562  14432.699    0.000    738.000
Children              10000.0      2.088      2.147    0.000      0.000
Age                   10000.0     53.078     20.699   18.000     35.000
Income                10000.0  39806.927  28199.917  348.670  19224.718
Outage_sec_perweek    10000.0     10.002      2.976    0.100      8.018
Email                 10000.0     12.016      3.026    1.000     10.000
Contacts              10000.0      0.994      0.988    0.000      0.000
Yearly_equip_failure  10000.0      0.398      0.636    0.000      0.000
Tenure                10000.0     34.526     26.443    1.000      7.918
MonthlyCharge         10000.0    172.625     42.943   79.979    139.979
Bandwidth_GB_Year     10000.0   3392.342   2185.295  155.507   1236.471

                            50%        75%         max
Population             2910.500  13168.000  111850.000
Children                  1.000      3.000      10.000
Age                      53.000     71.000      89.000
Income                33170.605  53246.170  258900.700
```

```
Outage_sec_perweek       10.019     11.969     21.207
Email                    12.000     14.000     23.000
Contacts                  1.000      2.000      7.000
Yearly_equip_failure      0.000      1.000      6.000
Tenure                   35.431     61.480     71.999
MonthlyCharge           167.485    200.735    290.160
Bandwidth_GB_Year      3279.537   5586.141   7158.982
```

Univariate Histogram Plot of Numeric Predictor Data. Here are the histogram plots for numeric data.

In [20]:
```python
# histogram plot numeric data
fig = plt.figure(figsize=(10, 20))
ax = df[num_cols].hist(bins = 50, figsize=(15,15))
plt.title('Numeric Data')
fig.tight_layout(h_pad=5, w_pad=5)
plt.savefig('output/' + COURSE + '/fig_hist_numeric.png', facecolor='w')
plt.show()
```

```
<Figure size 720x1440 with 0 Axes>
```



Survey data, not to be used for this analysis:

A. **Item1**: Timely response

B. **Item2**: Timely fixes

C. **Item3**: Timely replacements

D. **Item4**: Reliability

E. **Item5**: Options

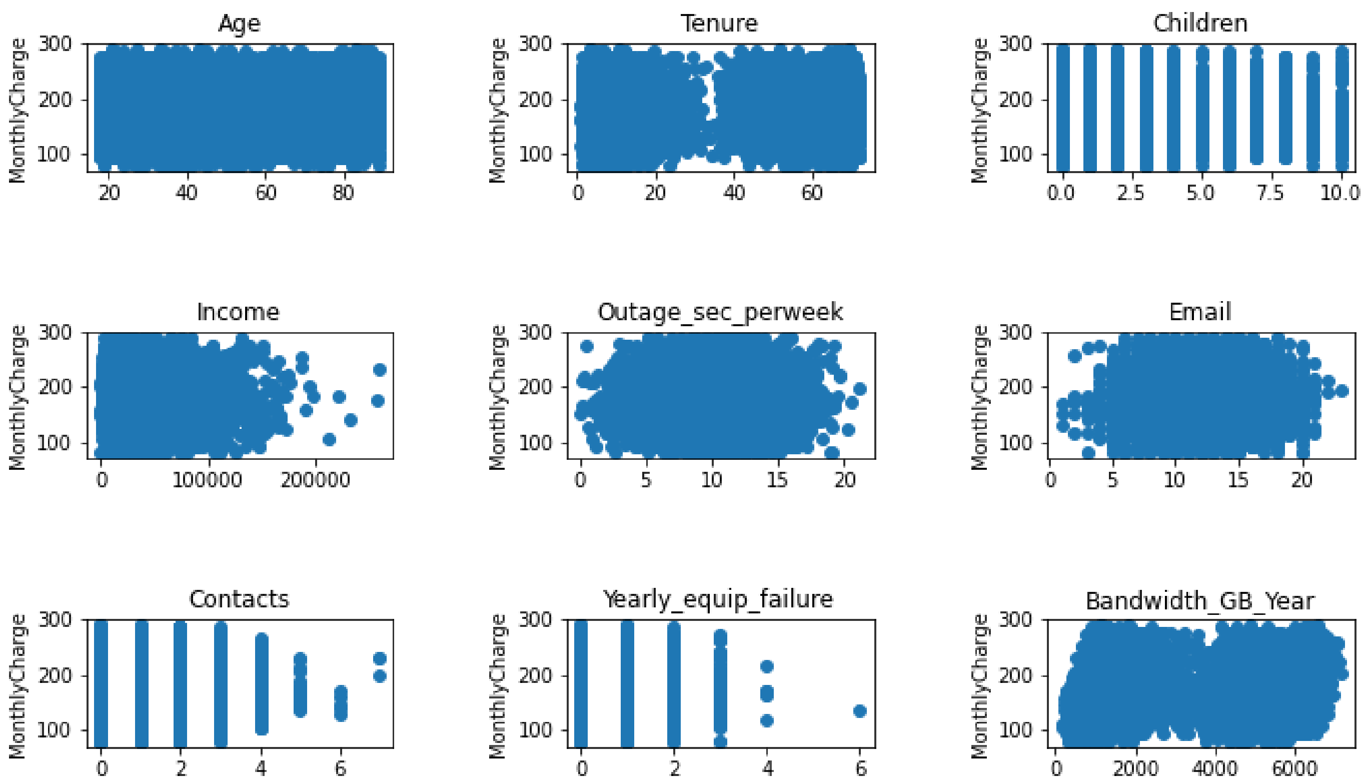
F. **Item6**: Respectful response

G. **Item7**: Courteous exchange

H. **Item8**: Evidence of active listening

Bivariate Scatter Plot of Numeric Predictor Data. Here are the scatter plots of selected numeric data vs. the target variable of 'MonthlyCharge'. One of the assumptions is that independent and dependent variables are linear, so I am looking for linear relationships here.

In [21]:
```python
# scatter plot of selected features
fig = plt.figure(figsize=(10, 20))
features = ['Age','Tenure','Children','Income',
            'Outage_sec_perweek','Email','Contacts',
            'Yearly_equip_failure','Bandwidth_GB_Year']
target = df['MonthlyCharge']

for i, col in enumerate(features):
    plt.subplot(10, 3, i+1)
    x = df[col]
    y = target
    plt.scatter(x, y, marker='o')
    plt.title(col)
    #plt.xlabel(col)
    plt.ylabel('MonthlyCharge')
    fig.tight_layout(h_pad=5, w_pad=5)

plt.savefig('output/' + COURSE + '/fig_scatterplot_selected.png', facecolor='w')
```



None of these indicate linear relationships. Use this information when selecting features to include in the model.

Bivariate Scatter Plot of Predictor Pairs. Here is a pair plot between each predictor variable and the others. Two (2) potential predictor variables should not have a linear relationship. If linear, then I will not use both together in the regression model. I am using the seaborn .pairplot() to scatter plot each pair of predictor variables.

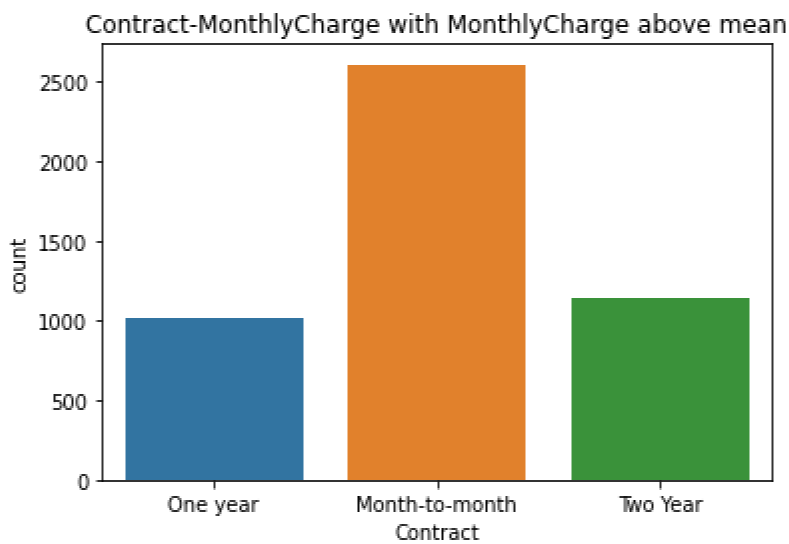
```
In [22]: # create data visualizations
ax = sns.pairplot(df[num_cols])
plt.show()
```

There is a linear relationship between **Tenure** and **Bandwidth_GB_Year** indicating multicollinearity. Ensure both variables are not selected together in the regression models. All of the other pairs look ok.

Bivariate Bar Plot - Contract Data. Here is the example where the .drop_first() method doesn't select the best unique value when converting dummy data. The plot shows the number of contracts by type of contract where the customer had above averate monthly charge.

In [23]:
```
# bar plot contract-monthlycharge
c = 'Contract'
mean = contract['MonthlyCharge'].mean()
print('Mean of MonthlyCharge: ${:.2f}'.format(mean))
temp_df = contract.query('MonthlyCharge>=172')
sns.countplot(x=c, data=temp_df)
plt.title('Contract-MonthlyCharge with MonthlyCharge above mean')
print(temp_df.groupby("Contract")["MonthlyCharge"].count())
plt.savefig('output/' + COURSE + '/fig_barplot_contracts.png', facecolor='w')
plt.show()
```

```
Mean of MonthlyCharge: $172.62
Contract
Month-to-month    2609
One year          1017
Two Year          1141
Name: MonthlyCharge, dtype: int64
```



Notice there is a higher number of customers with Month-to-month contracts where the **MonthlyCharge** is at or above the mean. This may be an area to consider in a future analysis.

C5. PROVIDE COPY OF FINAL DATA

In [24]:
```
# Provide copy of the prepared data set.
final_data = 'd208_final_data.csv'
df.to_csv(final_data, index=False, header=True)
print('File saved to: {}'.format(final_data))
print(df.columns.to_series().groupby(df.dtypes).groups)
```

```
File saved to: d208_final_data.csv
{uint8: ['Area_Suburban', 'Area_Urban', 'Marital_Married', 'Marital_Never Married', 'Mari
tal_Separated', 'Marital_Widowed', 'Gender_Male', 'Gender_Nonbinary', 'Techie_Yes', 'Cont
ract_One year', 'Contract_Two Year', 'Port_modem_Yes', 'Tablet_Yes', 'InternetService_Fib
er Optic', 'InternetService_None', 'Phone_Yes', 'Multiple_Yes', 'OnlineSecurity_Yes', 'On
lineBackup_Yes', 'DeviceProtection_Yes', 'TechSupport_Yes', 'StreamingTV_Yes', 'Streaming
Movies_Yes', 'PaperlessBilling_Yes', 'PaymentMethod_Credit Card (automatic)', 'PaymentMet
hod_Electronic Check', 'PaymentMethod_Mailed Check'], int64: ['Population', 'Children',
'Age', 'Email', 'Contacts', 'Yearly_equip_failure'], float64: ['Income', 'Outage_sec_perw
eek', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year']}
```

PART IV: MODEL COMPARISON AND ANALYSIS

D. Compare an initial and a reduced multiple regression model by doing the following:

D1. INITIAL MODEL

1. Construct an initial multiple regression model from all predictors that were identified in Part C2.

Initial Model. Here is the first model using all available predictor variables. I will use all of the available numeric and dummy categorical data for the initial model. However, based on "common sense", I believe that the models will eliminate all of the variables with the exception of the actual services involved. In other words, I don't believe **Age** for example, has anything to do with predicting **MonthlyCharge**. It makes sense that the **MonthlyCharge** would be a function of the customer's services, more than anything else. I will address this in the final analysis summary to see if my observations here bear any merit.

In [25]:
```python
# initial model
y = df.loc[ : , df.columns == 'MonthlyCharge']
X = df.loc[ : , df.columns != 'MonthlyCharge']
Xc = sm.add_constant(X)
model_1 = sm.OLS(y, Xc).fit()
print(model_1.summary2()) # using alternate summary layout
```

```
                       Results: Ordinary least squares
=========================================================================================
=
Model:                     OLS                    Adj. R-squared:           0.995
Dependent Variable:        MonthlyCharge          AIC:                      49597.878
9
Date:                      2021-09-26 21:37       BIC:                      49871.871
9
No. Observations:          10000                  Log-Likelihood:           -24761.
Df Model:                  37                     F-statistic:              5.966e+04
Df Residuals:              9962                   Prob (F-statistic):       0.00
R-squared:                 0.996                  Scale:                    8.3155
-----------------------------------------------------------------------------------------
-
                                     Coef.   Std.Err.      t     P>|t|   [0.025   0.975]
-----------------------------------------------------------------------------------------
-
const                              -88.3666   0.6488 -136.2005 0.0000 -89.6384 -87.094
8
Population                          -0.0000   0.0000   -0.9277 0.3536  -0.0000   0.000
0
Children                            -9.5210   0.0358 -266.2744 0.0000  -9.5910  -9.450
9
Age                                  1.0142   0.0038  268.2800 0.0000   1.0068   1.021
7
Income                               0.0000   0.0000    0.7061 0.4801  -0.0000   0.000
0
Outage_sec_perweek                   0.0057   0.0097    0.5916 0.5541  -0.0133   0.024
8
Email                               -0.0017   0.0095   -0.1781 0.8587  -0.0204   0.017
0
Contacts                            -0.0199   0.0292   -0.6800 0.4965  -0.0772   0.037
4
Yearly_equip_failure                -0.0078   0.0454   -0.1719 0.8635  -0.0969   0.081
2
Tenure                             -25.3563   0.0881 -287.6991 0.0000 -25.5291 -25.183
5
Bandwidth_GB_Year                    0.3095   0.0011  287.7098 0.0000   0.3074   0.311
6
Area_Suburban                       -0.1677   0.0707   -2.3731 0.0177  -0.3062  -0.029
2
Area_Urban                          -0.0841   0.0708   -1.1886 0.2346  -0.2229   0.054
6
Marital_Married                     -0.0133   0.0914   -0.1453 0.8845  -0.1924   0.165
8
Marital_Never Married               -0.0524   0.0909   -0.5763 0.5644  -0.2305   0.125
7
Marital_Separated                    0.0672   0.0901    0.7458 0.4558  -0.1094   0.243
8
Marital_Widowed                     -0.0146   0.0900   -0.1626 0.8709  -0.1911   0.161
8
```

```
Gender_Male                           -20.1183   0.0905 -222.2582 0.0000 -20.2958 -19.940
9
Gender_Nonbinary                        6.5516   0.1960   33.4241 0.0000   6.1674   6.935
9
Techie_Yes                              0.0512   0.0773    0.6625 0.5077  -0.1003   0.202
7
Contract_One year                       0.0415   0.0742    0.5594 0.5759  -0.1039   0.186
8
Contract_Two Year                       0.0638   0.0703    0.9074 0.3642  -0.0740   0.201
7
Port_modem_Yes                          0.0449   0.0578    0.7772 0.4371  -0.0683   0.158
1
Tablet_Yes                             -0.0009   0.0631   -0.0138 0.9890  -0.1246   0.122
9
InternetService_Fiber Optic           148.0308   0.4503  328.7031 0.0000 147.1480 148.913
6
InternetService_None                  115.2702   0.4524  254.7721 0.0000 114.3833 116.157
1
Phone_Yes                              -0.0183   0.0993   -0.1841 0.8539  -0.2130   0.176
4
Multiple_Yes                           10.3089   0.0967  106.5650 0.0000  10.1192  10.498
5
OnlineSecurity_Yes                    -20.7889   0.1014 -204.9362 0.0000 -20.9878 -20.590
1
OnlineBackup_Yes                       -6.5860   0.1168  -56.4004 0.0000  -6.8149  -6.357
1
DeviceProtection_Yes                  -13.7972   0.1084 -127.2990 0.0000 -14.0096 -13.584
7
TechSupport_Yes                        11.1126   0.0599  185.5809 0.0000  10.9953  11.230
0
StreamingTV_Yes                       -28.3892   0.2520 -112.6494 0.0000 -28.8832 -27.895
2
StreamingMovies_Yes                   -12.7029   0.2333  -54.4516 0.0000 -13.1602 -12.245
6
PaperlessBilling_Yes                   -0.0733   0.0587   -1.2489 0.2117  -0.1884   0.041
8
PaymentMethod_Credit Card (automatic)   0.0268   0.0880    0.3041 0.7610  -0.1457   0.199
3
PaymentMethod_Electronic Check          0.0688   0.0787    0.8733 0.3825  -0.0856   0.223
1
PaymentMethod_Mailed Check              0.1827   0.0860    2.1256 0.0336   0.0142   0.351
2
-----------------------------------------------------------------------------------------
-
Omnibus:                    48314.337            Durbin-Watson:                 2.011
Prob(Omnibus):              0.000                Jarque-Bera (JB):              1091.29
4
Skew:                       -0.026               Prob(JB):                      0.000
Kurtosis:                   1.382                Condition No.:                 1670116
=========================================================================================
=
* The condition number is large (2e+06). This might indicate             strong
multicollinearity or other numerical problems.


p:\code_wgu\5\v\lib\site-packages\statsmodels\tsa\tsatools.py:142: FutureWarning: In a fu
ture version of pandas all arguments of concat except for the argument 'objs' will be key
word-only
  x = pd.concat(x[::order], 1)
```

Notice high condition number indicating possible high multicollinearity between predictor variables. Also, many of the predictor p-values are high, above 0.05.

Coorelation Data. Here is a coorelation matrix and a list of the data-pairs with the highest coorelation.

In [26]:
```
# find predictor pairs with high coorelation
#custom_corr_matrix(X,'Model_2')
get_top_abs_correlations(X, 10)
```

Out[26]:
```
Tenure                               Bandwidth_GB_Year                 0.991495
Area_Suburban                        Area_Urban                        0.500711
InternetService_Fiber Optic          InternetService_None              0.461753
PaymentMethod_Electronic Check       PaymentMethod_Mailed Check        0.390989
PaymentMethod_Credit Card (automatic) PaymentMethod_Electronic Check   0.367992
Contract_One year                    Contract_Two Year                 0.293243
PaymentMethod_Credit Card (automatic) PaymentMethod_Mailed Check       0.279547
Marital_Separated                    Marital_Widowed                   0.253210
Marital_Never Married                Marital_Widowed                   0.248636
                                     Marital_Separated                 0.247636
dtype: float64
```

The top correlation is between **Tenure** and **Bandwidth_GB_Year**. We saw this relationship earlier in the exploratory data analysis section. Based on this high correlation, remove **Bandwidth_GB_Year** prior to the next model iteration.

High P-Values. On the high p-values side of the house, I wrote code to loop through the model summary and drop any predictor whose p-value is greater than 0.05. Here is the code:

In [27]:
```
# drop all columns from model where p-value > 0.05 (see Geeks for Geeks (2021))
equation = model_1.summary2().tables[1]
temp_drop = []
for i in equation.itertuples():
    if i[4] > 0.05:
        temp_drop.append(i[0])
        print('Drop {} with p-value of {:.3f}.'.format(i[0],i[4]))
X = pd.DataFrame(X) # reset dataframe
X.drop(temp_drop, axis = 1, inplace=True) # drop
```

```
Drop Population with p-value of 0.354.
Drop Income with p-value of 0.480.
Drop Outage_sec_perweek with p-value of 0.554.
Drop Email with p-value of 0.859.
Drop Contacts with p-value of 0.497.
Drop Yearly_equip_failure with p-value of 0.863.
Drop Area_Urban with p-value of 0.235.
Drop Marital_Married with p-value of 0.885.
Drop Marital_Never Married with p-value of 0.564.
Drop Marital_Separated with p-value of 0.456.
Drop Marital_Widowed with p-value of 0.871.
Drop Techie_Yes with p-value of 0.508.
Drop Contract_One year with p-value of 0.576.
Drop Contract_Two Year with p-value of 0.364.
Drop Port_modem_Yes with p-value of 0.437.
Drop Tablet_Yes with p-value of 0.989.
Drop Phone_Yes with p-value of 0.854.
Drop PaperlessBilling_Yes with p-value of 0.212.
Drop PaymentMethod_Credit Card (automatic) with p-value of 0.761.
Drop PaymentMethod_Electronic Check with p-value of 0.383.
```

In [28]:
```
# drop other columns with high multi-collinearity (see Geeks for Geeks (2021))
temp_drop =['Bandwidth_GB_Year', 'Tenure','Children','InternetService_None','Age']
X = pd.DataFrame(X) # reset dataframe
X.drop(temp_drop, axis = 1, inplace=True) # drop
```

In [29]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Area_Suburban                10000 non-null  uint8
 1   Gender_Male                  10000 non-null  uint8
 2   Gender_Nonbinary             10000 non-null  uint8
 3   InternetService_Fiber Optic  10000 non-null  uint8
 4   Multiple_Yes                 10000 non-null  uint8
 5   OnlineSecurity_Yes           10000 non-null  uint8
 6   OnlineBackup_Yes             10000 non-null  uint8
 7   DeviceProtection_Yes         10000 non-null  uint8
 8   TechSupport_Yes              10000 non-null  uint8
 9   StreamingTV_Yes              10000 non-null  uint8
 10  StreamingMovies_Yes          10000 non-null  uint8
 11  PaymentMethod_Mailed Check   10000 non-null  uint8
dtypes: uint8(12)
memory usage: 117.3 KB
```

Updated Model. Here is the next iteration of the regression model:

In [30]:

```python
# updated model
# y is already defined
# X is already defined and reduced
Xc = sm.add_constant(X) # reset
model_2 = sm.OLS(y, Xc).fit()
print(model_2.summary2()) # using alternate summary layout
```

```
                      Results: Ordinary least squares
============================================================================
Model:                 OLS                Adj. R-squared:       0.946
Dependent Variable:    MonthlyCharge      AIC:                  74366.7372
Date:                  2021-09-26 21:37   BIC:                  74460.4716
No. Observations:      10000              Log-Likelihood:       -37170.
Df Model:              12                 F-statistic:          1.465e+04
Df Residuals:          9987               Prob (F-statistic):   0.00
R-squared:             0.946              Scale:                99.236
----------------------------------------------------------------------------
                             Coef.  Std.Err.    t     P>|t|   [0.025  0.975]
----------------------------------------------------------------------------
const                       78.8953   0.3004 262.6253 0.0000 78.3065 79.4842
Area_Suburban                0.0860   0.2112   0.4072 0.6839 -0.3280  0.5000
Gender_Male                 -0.1950   0.2018  -0.9662 0.3340 -0.5906  0.2006
Gender_Nonbinary            -0.9131   0.6708  -1.3613 0.1735 -2.2279  0.4017
InternetService_Fiber Optic 24.7407   0.2008 123.2322 0.0000 24.3471 25.1342
Multiple_Yes                32.7987   0.1999 164.0971 0.0000 32.4069 33.1905
OnlineSecurity_Yes           2.8015   0.2080  13.4711 0.0000  2.3938  3.2091
OnlineBackup_Yes            22.5834   0.2004 112.7161 0.0000 22.1906 22.9761
DeviceProtection_Yes        12.4557   0.2009  62.0043 0.0000 12.0619 12.8494
TechSupport_Yes             12.6435   0.2059  61.3954 0.0000 12.2398 13.0472
StreamingTV_Yes             42.1841   0.1993 211.6510 0.0000 41.7934 42.5748
StreamingMovies_Yes         52.3389   0.1994 262.5063 0.0000 51.9481 52.7297
PaymentMethod_Mailed Check   0.1155   0.2372   0.4872 0.6261 -0.3493  0.5804
----------------------------------------------------------------------------
Omnibus:                 29582.971         Durbin-Watson:            2.007
Prob(Omnibus):           0.000             Jarque-Bera (JB):         699.816
Skew:                    -0.052            Prob(JB):                 0.000
Kurtosis:                1.708             Condition No.:            12
============================================================================
```

```
p:\code_wgu\5\v\lib\site-packages\statsmodels\tsa\tsatools.py:142: FutureWarning: In a fu
ture version of pandas all arguments of concat except for the argument 'objs' will be key
word-only
  x = pd.concat(x[::order], 1)
```

In [31]:
```
# equation of the regression line/plane
print('Adj. R-squared: {}'.format(model_2.summary2().tables[0][3][0]))
equation = model_2.summary2().tables[1]
print('Estimate [{}] as y = '.format(model_2.summary2().tables[0][1][1]))
for i in equation.itertuples():
    print('   {:+.2f} x ( {} ) '.format(i[1],i[0]))
```

```
Adj. R-squared: 0.946
Estimate [MonthlyCharge] as y =
   +78.90 x ( const )
   +0.09 x ( Area_Suburban )
   -0.20 x ( Gender_Male )
   -0.91 x ( Gender_Nonbinary )
   +24.74 x ( InternetService_Fiber Optic )
   +32.80 x ( Multiple_Yes )
   +2.80 x ( OnlineSecurity_Yes )
   +22.58 x ( OnlineBackup_Yes )
   +12.46 x ( DeviceProtection_Yes )
   +12.64 x ( TechSupport_Yes )
   +42.18 x ( StreamingTV_Yes )
   +52.34 x ( StreamingMovies_Yes )
   +0.12 x ( PaymentMethod_Mailed Check )
```

High Coorelation. Use coorelation matrix to find predictor pairs with high coorelation.

In [32]:
```
# find predictor pairs with high coorelation
#custom_corr_matrix(X,'Model_2')
get_top_abs_correlations(X, 10)
```

Out[32]:
```
Gender_Male                  Gender_Nonbinary             0.146092
Gender_Nonbinary             OnlineBackup_Yes             0.029316
InternetService_Fiber Optic  TechSupport_Yes              0.026211
Gender_Male                  PaymentMethod_Mailed Check   0.022103
DeviceProtection_Yes         StreamingMovies_Yes          0.019450
Gender_Male                  DeviceProtection_Yes         0.018678
Gender_Nonbinary             DeviceProtection_Yes         0.016523
Gender_Male                  OnlineSecurity_Yes           0.016105
Area_Suburban                TechSupport_Yes              0.015650
Gender_Male                  StreamingTV_Yes              0.015094
dtype: float64
```

It looks like **Gender_Male** and **Gender_Nonbinary** are slightly coorelated. I spectulate that maybe a large percent of the **Gender_Nonbinary** might actually be male. I am not going to do anything with this but, maybe this relationship could be addressed in a future study.

D3. FINAL MODEL

3. Provide a reduced multiple regression model that includes both categorical and continuous variables. Note: The output should include a screenshot of each model.

This analysis created three (3) models all together. The initial model shown above, then a second model. The second model also had predictor variables with high p-values. But, for the final model, the numerical data was left in the model. The for-loop was used to remove only the column(s) with high multicollinearity.

High P-Values. Find and remove predictors whose p-value is greater than 0.05.

In [33]:
```
# drop all columns from model where p-value > 0.05 (see Geeks for Geeks (2021))
equation = model_2.summary2().tables[1]
temp_drop = []
for i in equation.itertuples():
    if i[4] > 0.05:
        temp_drop.append(i[0])
        print('Drop {} with p-value of {:.3f}.'.format(i[0],i[4]))
X = pd.DataFrame(X) # reset dataframe
X.drop(temp_drop, axis = 1, inplace=True) # drop
```

```
Drop Area_Suburban with p-value of 0.684.
Drop Gender_Male with p-value of 0.334.
Drop Gender_Nonbinary with p-value of 0.173.
Drop PaymentMethod_Mailed Check with p-value of 0.626.
```

Run Final Model. Final model.

In [34]:
```
# final model
# y is already defined
# X is already defined and reduced
Xc = sm.add_constant(X) # reset
model_3 = sm.OLS(y, Xc).fit()
print(model_3.summary2()) # using alternate summary layout
```

```
                     Results: Ordinary least squares
============================================================================
Model:                  OLS                  Adj. R-squared:       0.946
Dependent Variable:     MonthlyCharge        AIC:                  74361.6344
Date:                   2021-09-26 21:37     BIC:                  74426.5275
No. Observations:       10000                Log-Likelihood:       -37172.
Df Model:               8                    F-statistic:          2.198e+04
Df Residuals:           9991                 Prob (F-statistic):   0.00
R-squared:              0.946                Scale:                99.225
----------------------------------------------------------------------------
                            Coef.  Std.Err.    t     P>|t|   [0.025  0.975]
----------------------------------------------------------------------------
const                      78.8429   0.2689 293.2430 0.0000 78.3159 79.3700
InternetService_Fiber Optic 24.7418   0.2007 123.2522 0.0000 24.3483 25.1353
Multiple_Yes               32.7986   0.1999 164.1060 0.0000 32.4068 33.1904
OnlineSecurity_Yes          2.8009   0.2079  13.4720 0.0000  2.3934  3.2084
OnlineBackup_Yes           22.5755   0.2002 112.7469 0.0000 22.1830 22.9680
DeviceProtection_Yes       12.4560   0.2008  62.0275 0.0000 12.0624 12.8496
TechSupport_Yes            12.6418   0.2059  61.4057 0.0000 12.2382 13.0454
StreamingTV_Yes            42.1795   0.1993 211.6736 0.0000 41.7889 42.5701
StreamingMovies_Yes        52.3391   0.1994 262.5468 0.0000 51.9484 52.7299
----------------------------------------------------------------------------
Omnibus:                  30980.596         Durbin-Watson:            2.008
Prob(Omnibus):            0.000             Jarque-Bera (JB):         702.004
Skew:                     -0.052            Prob(JB):                 0.000
Kurtosis:                 1.706             Condition No.:            5
============================================================================

p:\code_wgu\5\v\lib\site-packages\statsmodels\tsa\tsatools.py:142: FutureWarning: In a fu
ture version of pandas all arguments of concat except for the argument 'objs' will be key
word-only
  x = pd.concat(x[::order], 1)
```

E1. EXPLAIN DATA ANALYSIS

> **1.** Explain your data analysis process by comparing the initial and reduced multiple regression models, including the following elements: • the logic of the variable selection technique • the model evaluation metric • a residual plot

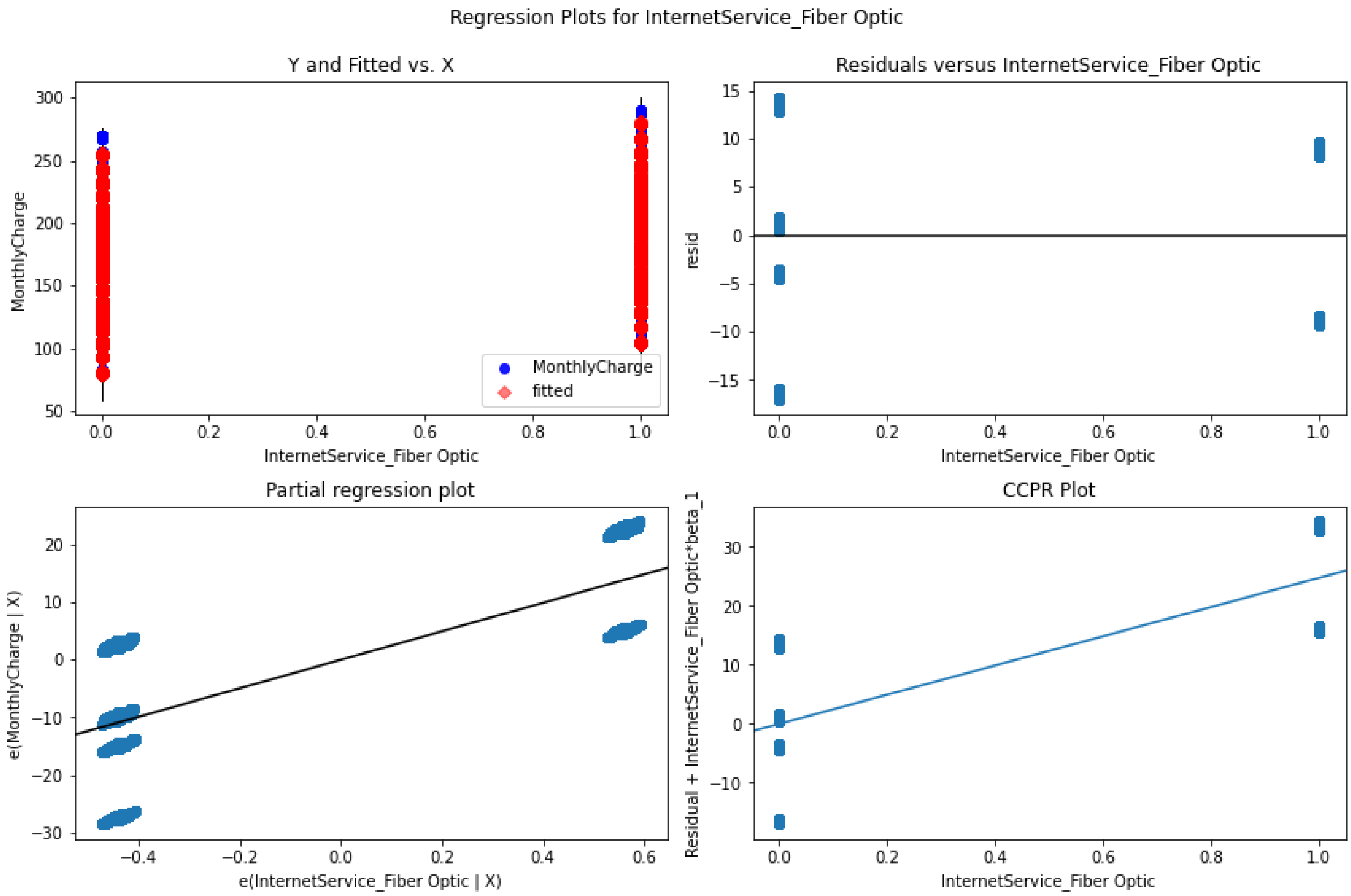
Logic of Variable Selection. The features for each model were selected based on high p-value and high multi-collinearity with other feature.

Model Evaluation Metric. Each of the models are compared to each other using the cooeefficient of determination, R-squared, the condition number and the total number of features. Here is a summary table showing the R-squared values for each of the model iterations:

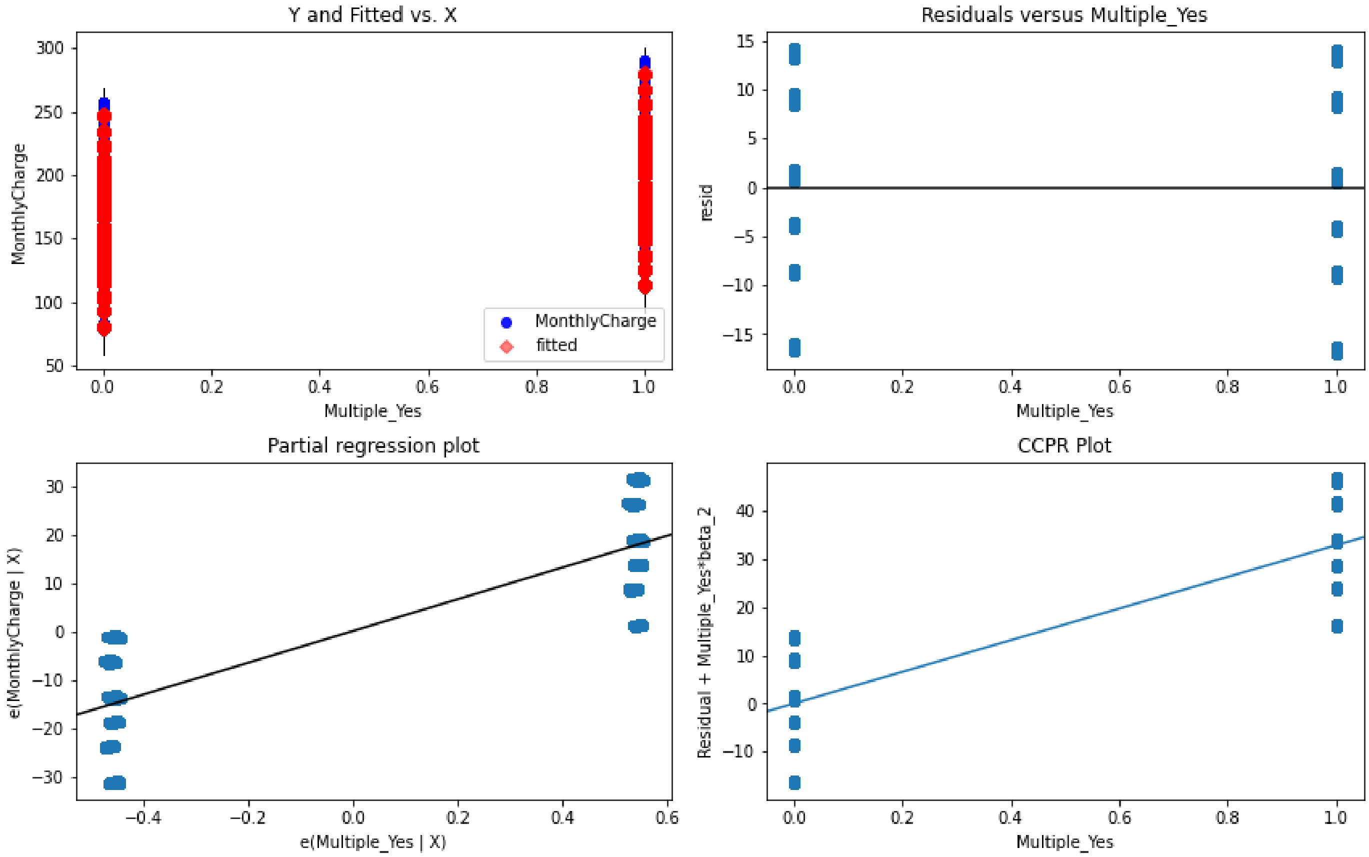
```
Model Comparison
+---------+----------+----------+----------+
|  Model  | R-squared | Cond Numb | #Feature |
+---------+----------+----------+----------+
| Model 1 |     0.995 |   1670116 |       37 |
| Model 2 |     0.946 |        12 |       12 |
| Model 3 |     0.946 |         5 |        8 |
+---------+----------+----------+----------+
Note. R-squared is the 'Adj R-square' value
```

Residual Plots. Here are the residual plots for each of the model's final predictor variables using statsmodel.api:

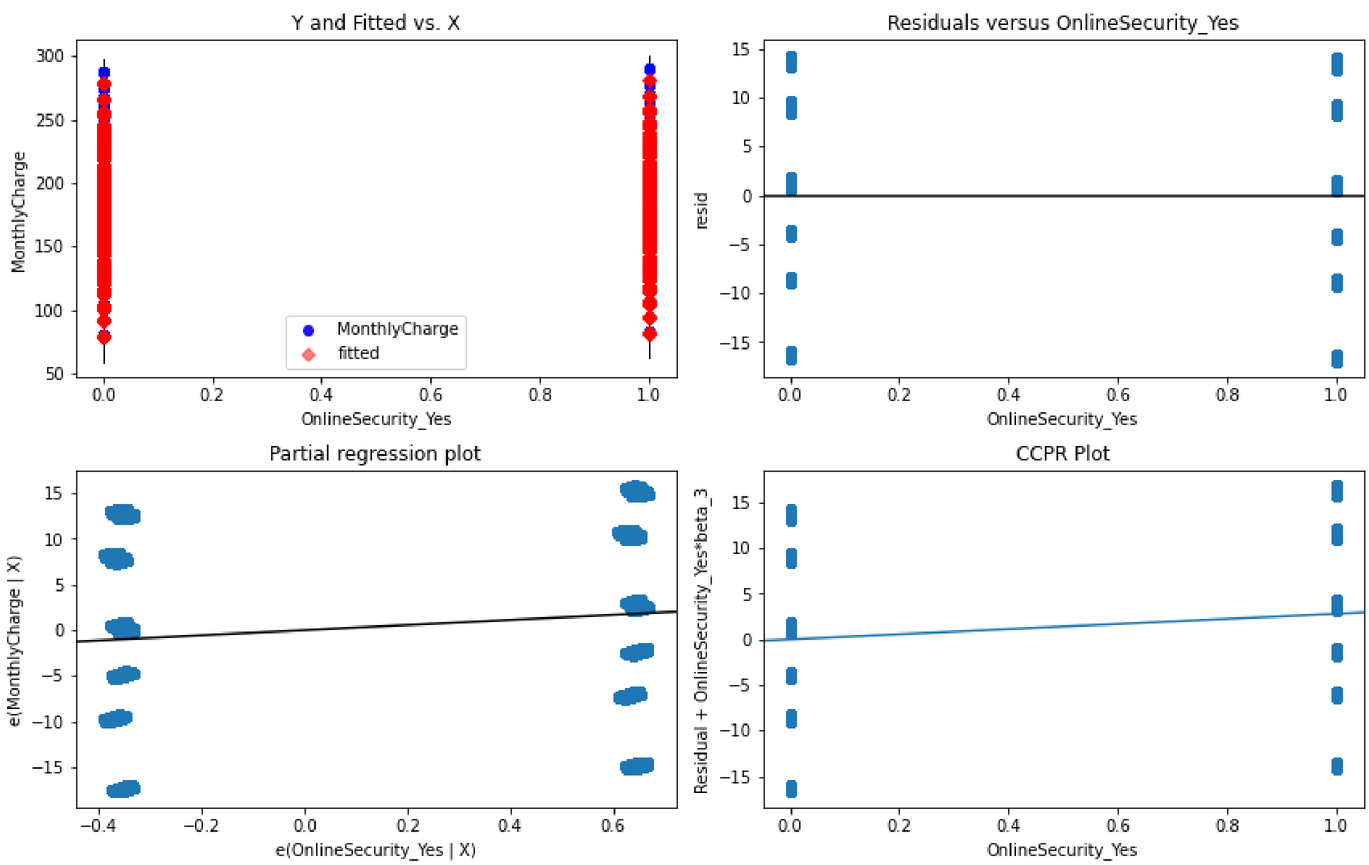
In [35]:
```
#create residual plots for all of the model's final predictor variables
for c in X.columns:
    fig = plt.figure(figsize=(12,8))
    fig = sm.graphics.plot_regress_exog(model_3, c, fig=fig)
```

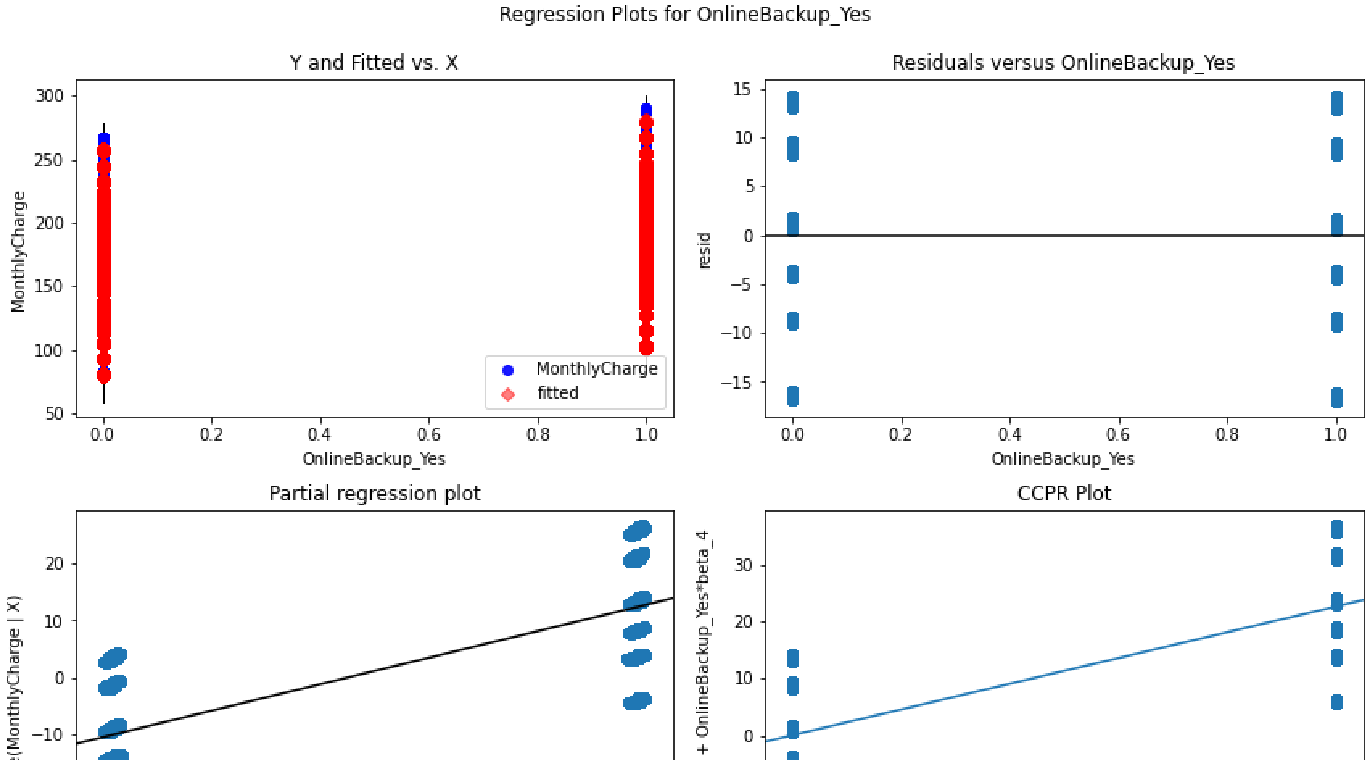


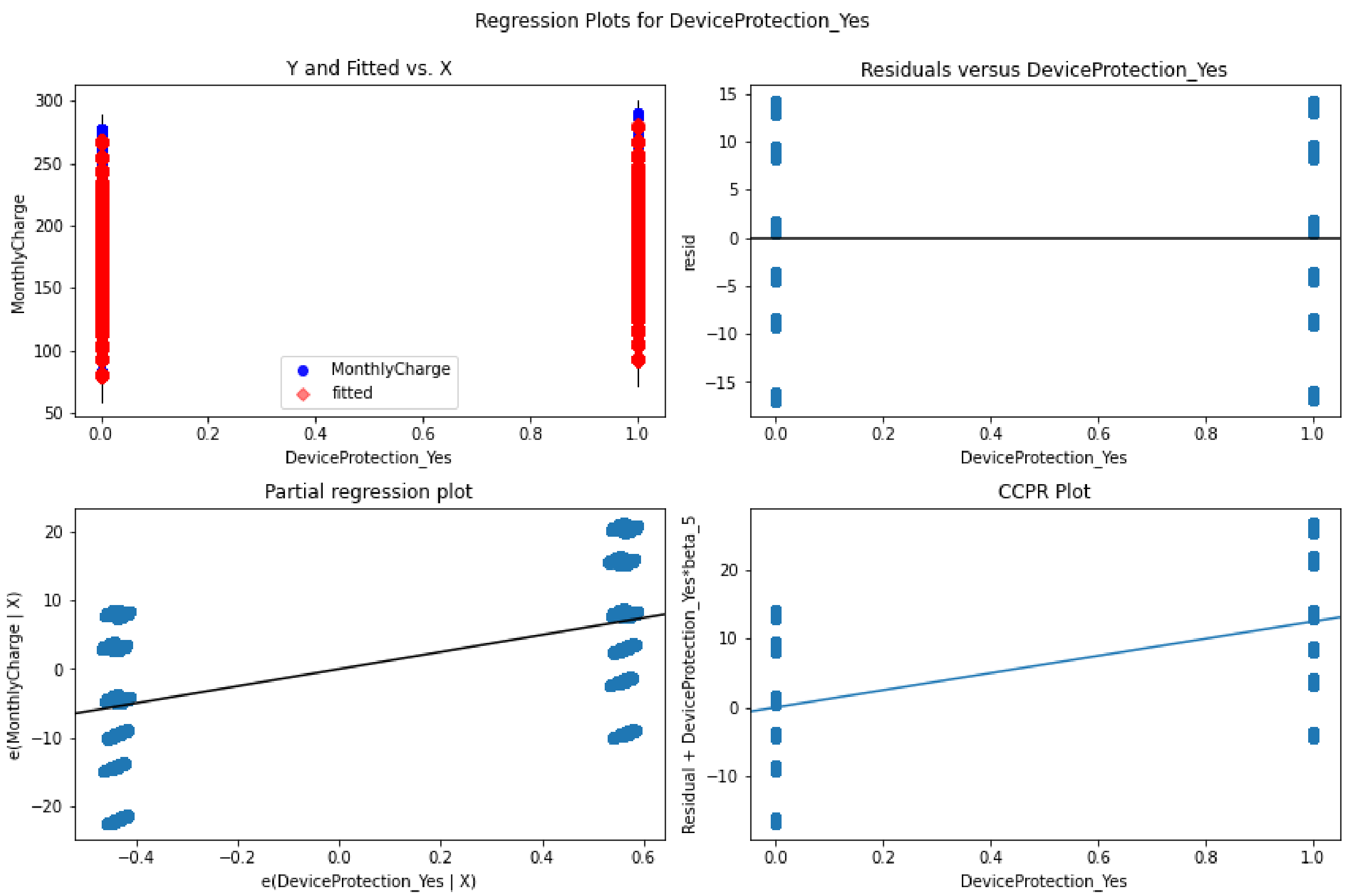
Regression Plots for Multiple_Yes



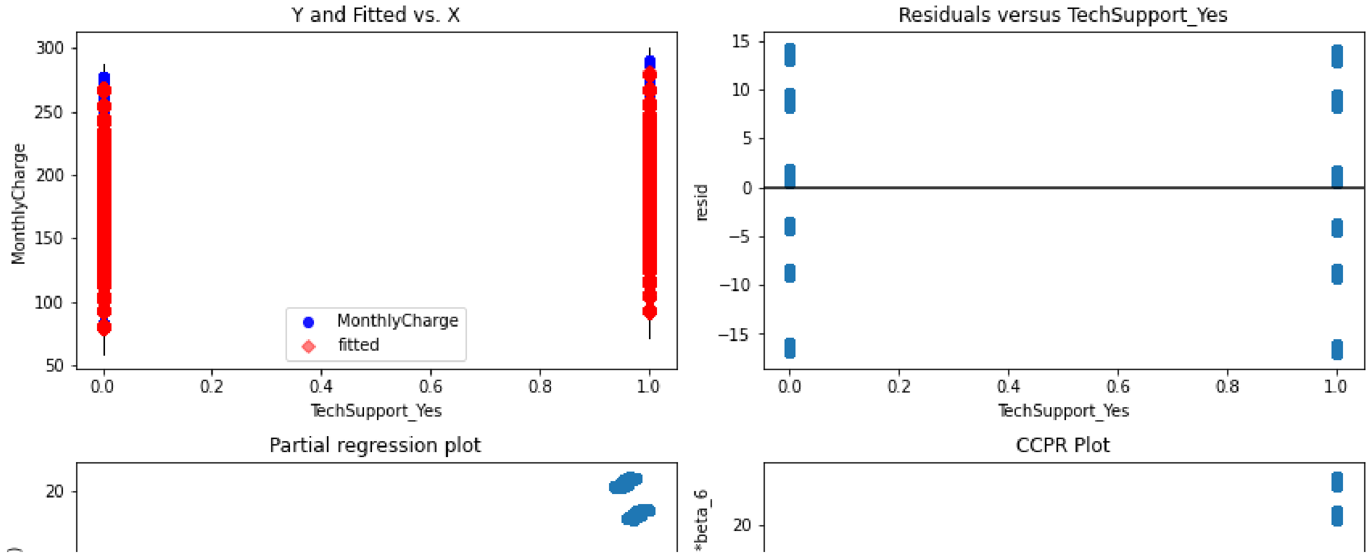
Regression Plots for OnlineSecurity_Yes



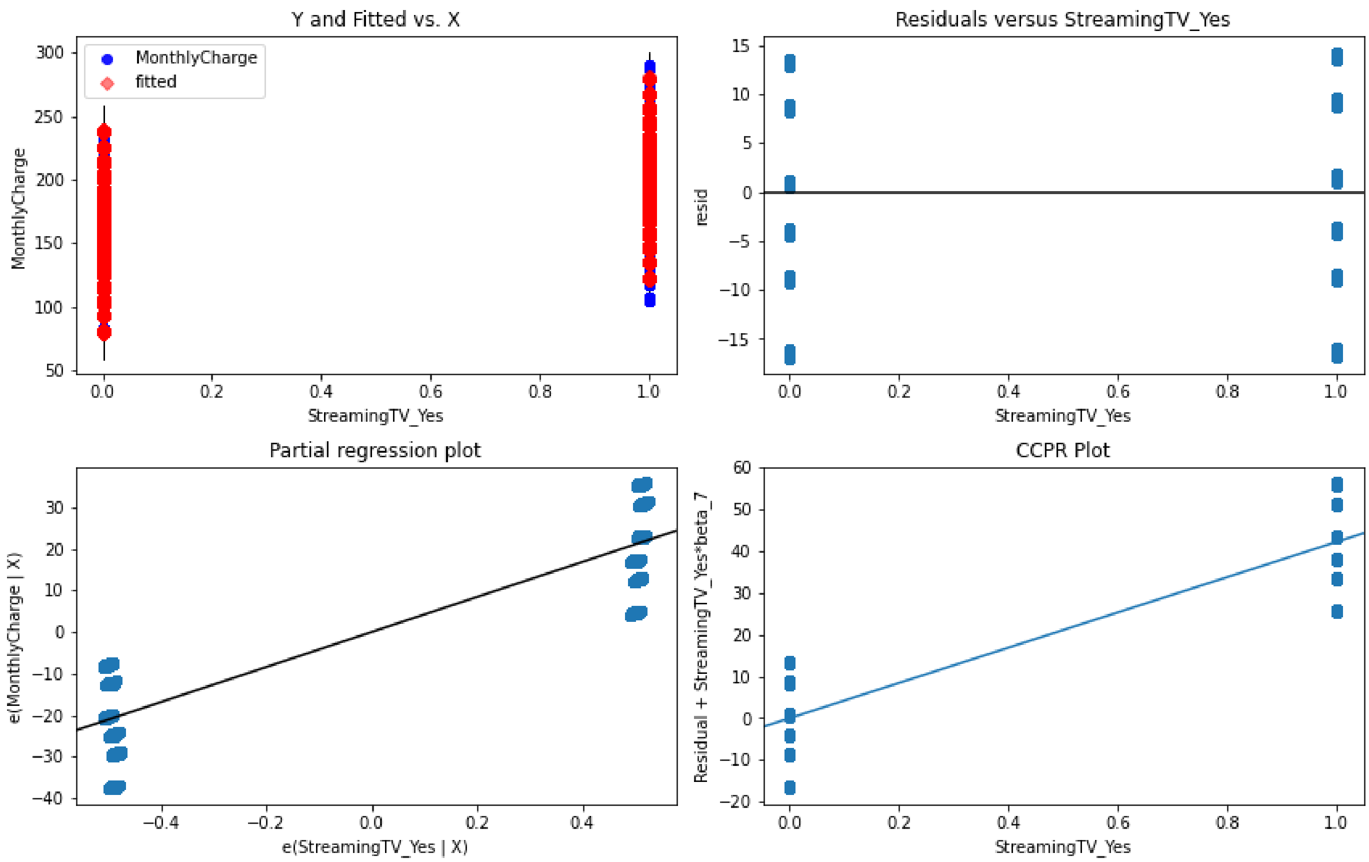

Regression Plots for OnlineBackup_Yes
Y and Fitted vs. X
MonthlyCharge
OnlineBackup_Yes
MonthlyCharge
fitted
Residuals versus OnlineBackup_Yes
resid
OnlineBackup_Yes
Partial regression plot
CCPR Plot
+ OnlineBackup_Yes*beta_4

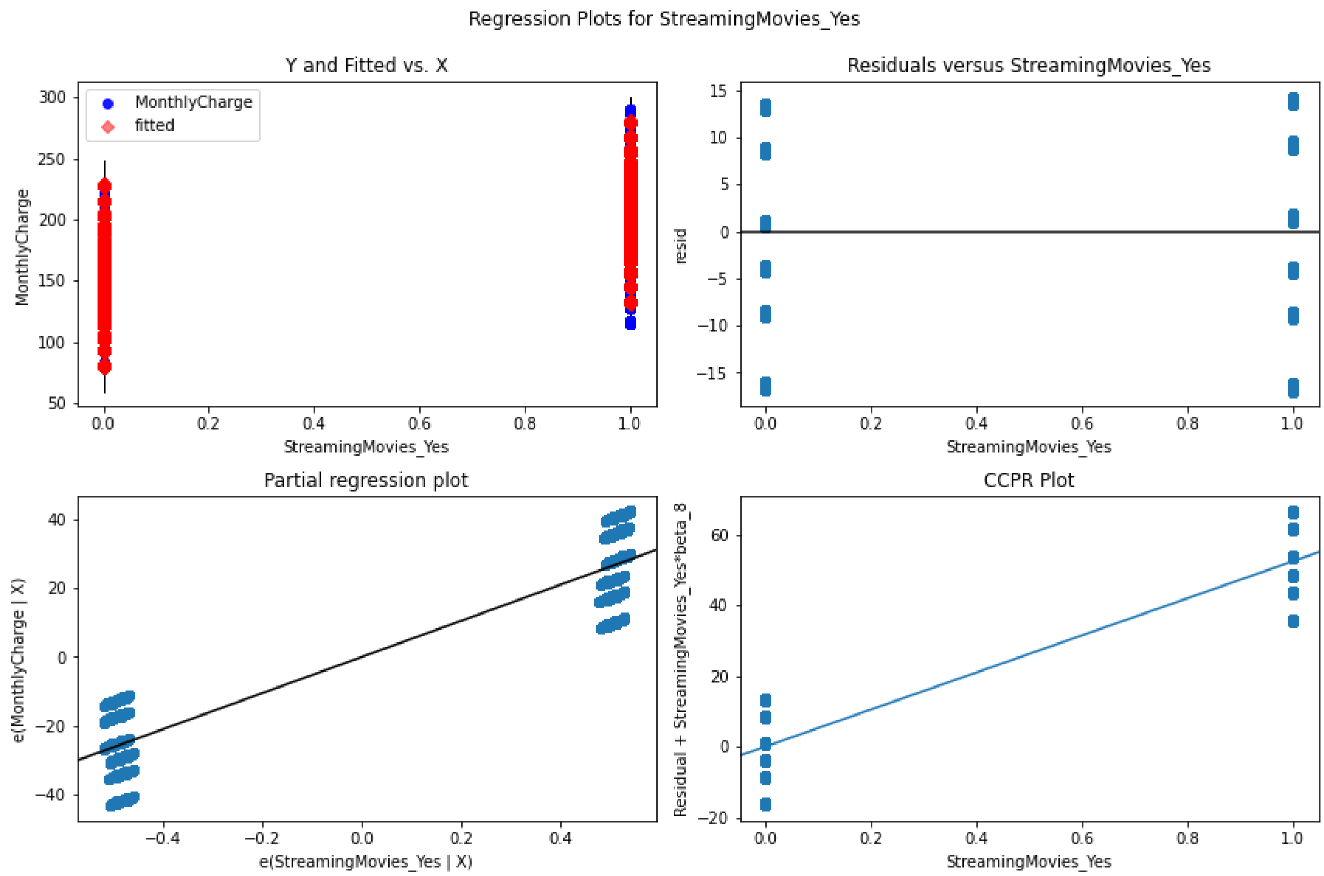

Regression Plots for DeviceProtection_Yes
Y and Fitted vs. X
MonthlyCharge
DeviceProtection_Yes
MonthlyCharge
fitted
Residuals versus DeviceProtection_Yes
resid
DeviceProtection_Yes
Partial regression plot
e(MonthlyCharge | X)
e(DeviceProtection_Yes | X)
CCPR Plot
Residual + DeviceProtection_Yes*beta_5
DeviceProtection_Yes

Regression Plots for TechSupport_Yes



Regression Plots for StreamingTV_Yes





Multivariate Normality. We can see that in each of the residual plots, the values are randomly distributed above and below the zero line. This is an indication of multivariate normality, which is to say that the residuals are normally distributed.

E2. OUTPUT AND CALCULATIONS

2. Provide the output and any calculations of the analysis you performed, including the model's residual error. Note: The output should include the predictions from the refined model you used to perform the analysis.

Output and Calculations. All of the output and calculations are contained within this Jupyter notebook.

E3. PROVIDE CODE

3. Provide the code used to support the implementation of the multiple regression models.

Code. All of the code, calculations and output are contained within this Jupyter notebook.

PART V: DATA SUMMARY AND IMPLICATIONS

F. Summarize your findings and assumptions by doing the following:

F1. FINAL RESULTS

1. Discuss the results of your data analysis, including the following elements: • a regression equation for the reduced model • an interpretation of coefficients of the statistically significant variables of the model • the statistical and practical significance of the model • the limitations of the data analysis

Final Regression Equation. The model is complete. Here is the final regression equation. And, as I predicted, the final model is based on the customer's services. All of the other variables that we started with have been eliminated because of high p-values or high multi-collinearity with other variables.

In [36]:
```
# equation of the regression line/plane
print('Adj. R-squared: {}'.format(model_3.summary2().tables[0][3][0]))
equation = model_3.summary2().tables[1]
print('Estimate [{}] as y = '.format(model_3.summary2().tables[0][1][1]))
for i in equation.itertuples():
    print('   {:+.2f} x ( {} ) '.format(i[1],i[0]))
```

```
Adj. R-squared: 0.946
Estimate [MonthlyCharge] as y =
   +78.84 x ( const )
   +24.74 x ( InternetService_Fiber Optic )
   +32.80 x ( Multiple_Yes )
   +2.80 x ( OnlineSecurity_Yes )
   +22.58 x ( OnlineBackup_Yes )
   +12.46 x ( DeviceProtection_Yes )
   +12.64 x ( TechSupport_Yes )
   +42.18 x ( StreamingTV_Yes )
   +52.34 x ( StreamingMovies_Yes )
```

The final model uses eight (8) predictor variables and has an R-squared value of 94.6% and a condition number of 5 which indicate that this is a pretty good model.

Interpretation of Coefficients. Because the final regression model is based on categorical data, yes and no values, then each of the cooeefficients has the behaviour of adding a given value if yes, or adding zero (0) if no. For example, if the customer only has fiber optic service and nothing else, then you could acurately predict the monthly charge by adding the constant value of 78.84 to the cooefficient value of 24.74 which equals $103.58 in this case.

Limitations of the Data. Limitations of the data_______

F2. RECOMMENDATIONS

2. Recommend a course of action based on your results.

Recommendations. Recommend using the model to (1) predict economic value for a given customer based on the number of services and (2) consider offering discounts or rebates to customers who are paying more than an average monthly charge.

PART VI: DEMONSTRATION

G. VIDEO

G. Provide a Panopto video recording that includes all of the following elements: • a demonstration of the functionality of the code used for the analysis • an identification of the version of the programming environment • a comparison of the two multiple regression models you used in your analysis • an interpretation of the coefficients.

Video. Video created and the .mp4 file is attached to submission. Also, the video is published on the school's Panopto website in the 'D208' dropbox at https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=3721d93d-5d6c-445b-b244-ada2012c96b2.

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=3721d93d-5d6c-445b-b244-ada2012c96b2 (https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=3721d93d-5d6c-445b-b244-ada2012c96b2)

H. REFERENCE

H. List the web sources used to acquire data or segments of third-party code to support the application. Ensure the web sources are reliable.

Agarwal, A. (2021, September). Linear Regression on Boston Housing Dataset. Retrieved from: https://towardsdatascience.com/linear-regression-on-boston-housing-dataset-f409b7e4a155

Bari, A., Chaouchi, M., & Jung, T. (2021, August). How to List Business Objectives for Predictive Analytics. Retrieved from:

https://www.dummies.com/programming/big-data/data-science/how-to-list-business-objectives-for-predictive-analytics/

Geeks For Geeks (2021, May). How to Drop One or Multiple Columns in Pandas Dataframe. Retrieved from: https://www.geeksforgeeks.org/how-to-drop-one-or-multiple-columns-in-pandas-dataframe/

Massaron, L., Boschetti, A. (2016). Regression Analysis with Python. Retrieved from: https://www.packtpub.com/product/regression-analysis-with-python/9781785286315

I. SOURCES

I. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

Massaron, L., Boschetti, A. (2016). Regression Analysis with Python. Retrieved from: https://www.packtpub.com/product/regression-analysis-with-python/9781785286315

Sarkar, T. (2019, June 5). How do you check the quality of your regression model in Python. Retrieved from: https://towardsdatascience.com/how-do-you-check-the-quality-of-your-regression-model-in-python-fa61759ff685

J. PROFESSIONAL

J. Demonstrate professional communication in the content and presentation of your submission.

In [ ]:

```

```