D208_Performance_Assessment_NBM2_Task_2_revision2

July 22, 2021

Logistic Regression for Predictive Modeling Ryan L. Buchanan
Student ID: 001826691
Masters Data Analytics (12/01/2020)
Program Mentor: Dan Estes
385-432-9281 (MST)
rbuch49@wgu.edu

A1. Research Question: Can we determine which individual customers are at high risk of churn? And, can we determine which features are most significant to churn?

A2. Objectives & Goals: Stakeholders in the company will benefit by knowing, with some measure of confidence, which customers are likely to churn soon. This knowledge will provide weight for decisions in marketing improved services to customers with these characteristics & past user experiences.

B1. Summary of Assumptions: Assumptions of a logistic regression model include:

- It is based on Bernoulli (also, Binomial or Boolean) Distribution rather than Gaussian because the dependent variable is binary (in our dataset, to churn or not to churn).
- The predicted values are restricted to a range of nomial values: Yes or No.
- It predicts the probability of a particular outcome rather than the outcome itself.
- There are no high correlations (multicollinearity) among predictors.
- It is the logarithm of the odds of achieving 1. In other words, a regression model, where the output is natural logarithm of the odds, also known as logit.

B2. Tool Benefits: Python & IPython Jupyter notebooks will be used to support this analysis. Python offers an intuitive, simple & versatile programming style & syntax, as well as a large system of mature packages for data science & machine learning. Since, Python is cross-platform, it will work well whether consumers of the analysis are using Windows PCs or a MacBook laptop. It is fast when compared with other possible programming languages like R or MATLAB (Massaron, p. 8). Also, there is strong support for Python as the most popular data science programming language in popular literature & media (CBTNuggets, p. 1).

B3. Appropriate Technique: Logistic regression is an appropriate technique to analyze the research question because or dependent variable is binomial, Yes or No. We want to find out what the likelihood of customer churn is for individual customers, based on a list of independent variables (area type, job, children, age, income, etc.). It will improve our understanding of increased probability of churn as we include or remove different independent variables & find out whether or not they have a positive or negative relationship to our target variable.

C1. Data Goals: My approach will include: 1. Back up my data and the process I am following as a copy to my machine and, since this is a manageable dataset, to GitHub using command line and gitbash. 2. Read the data set into Python using Pandas' read_csv command. 3. Evaluate the data struture to better understand input data. 4. Naming the dataset as a the variable "churn_df" and subsequent useful slices of the dataframe as "df". 5. Examine potential misspellings, awkward variable naming & missing data. 6. Find outliers that may create or hide statistical significance using histograms. 7. Imputing records missing data with meaningful measures of central tendency (mean, median or mode) or simply remove outliers that are several standard deviations above the mean.

Most relevant to our decision making process is the dependent variable of "Churn" which is binary categorical with only two values, Yes or No. Churn will be our categorical target variable.

In cleaning the data, we may discover relevance of the continuous predictor variables:

- Children
- Income
- Outage_sec_perweek
- Email
- Contacts

- Yearly_equip_failure
- Tenure (the number of months the customer has stayed with the provider)
- MonthlyCharge
- Bandwidth_GB_Year

Likewise, we may discover relevance of the categorical predictor variables (all binary categorical with only two values, Yes or No, except where noted):

- Techie: Whether the customer considers themselves technically inclined (based on customer questionnaire when they signed up for services) (yes, no)
- Contract: The contract term of the customer (month-to-month, one year, two year)
- Port_modem: Whether the customer has a portable modem (yes, no)
- Tablet: Whether the customer owns a tablet such as iPad, Surface, etc. (yes, no)
- InternetService: Customer's internet service provider (DSL, fiber optic, None)
- Phone: Whether the customer has a phone service (yes, no)
- Multiple: Whether the customer has multiple lines (yes, no)
- OnlineSecurity: Whether the customer has an online security add-on (yes, no)
- OnlineBackup: Whether the customer has an online backup add-on (yes, no)
- DeviceProtection: Whether the customer has device protection add-on (yes, no)
- TechSupport: Whether the customer has a technical support add-on (yes, no)
- StreamingTV: Whether the customer has streaming TV (yes, no)
- StreamingMovies: Whether the customer has streaming movies (yes, no)

Finally, discrete ordinal predictor variables from the survey responses from customers regarding various customer service features may be relevant in the decision-making process. In the surveys, customers provided ordinal numerical data by rating 8 customer service factors on a scale of 1 to 8 (1 = most important, 8 = least important):

- Item1: Timely response
- Item2: Timely fixes
- Item3: Timely replacements
- Item4: Reliability
- Item5: Options
- Item6: Respectful response
- Item7: Courteous exchange
- Item8: Evidence of active listening

C2. Summary Statistics: As output by Python pandas dataframe methods below, the dataset consists of 50 original columns & 10,000 records. For purposes of this analysis certain user ID & demographic categorical variables (CaseOrder, Customer_id, Interaction, UID, City, State, County, Zip, Lat, Lng, Population, Area, TimeZone, Job, Marital, PaymentMethod) were removed from the dataframe. Also, binomial Yes/No or Male/Female, variables were encoded to 1/0, respectively. This resulted in 34 remaining numerical variables, including the target variable. The dataset appeared to be sufficiently cleaned leaving no null, NAs or missing data points.

Measures of central tendency through histograms & boxplots revealed normal distributions for Monthly_Charge, Outage_sec_perweek & Email. The cleaned dataset no longer retained any outliers. Histograms for Bandwidth_GB_Year & Tenure displayed bimodal distributions, which demonstrated a direct linear relationship with each other in a scatterplot. The average customer was 53 years-old (with a standard deviation of 20 years), had 2 children (with a standard deviation of 2 kids), an income of 39,806 (with a standard deviation of about 30,000), experienced 10 outage-seconds/week, was marketed to by email 12 times, contacted technical support less than one time, had less than 1 yearly equipment failure, has been with the company for 34.5 months, has a monthly charge of approximately 173 & uses 3,392 GBs/year.

C3. Steps to Prepare Data:

- Import dataset to Python dataframe.
- Rename columns/variables of survey to easily recognizable features (ex: Item1 to TimelyResponse).
- Get a description of dataframe, structure (columns & rows) & data types.
- View summary statistics.
- Drop less meaningful identifying (ex: Customer_id) & demographic columns (ex: zip code) from dataframe.
- Check for records with missing data & impute missing data with meaningful measures of central tendency (mean, median or mode) or simply remove outliers that are several standard deviations above the mean.
- Create dummy variables in order to encode categorical, yes/no data points into 1/0 numerical values.
- View univariate & bivariate visualizations.
- Place target variable (the intercept) Churn at end of dataframe
- Finally, the prepared dataset will be extracted & provided as churn_prepared_log.csv.

[62]:
```
# Increase Jupyter display cell-width
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:75% !important; }</style>"))
```

```
<IPython.core.display.HTML object>
```

[63]:
```
# Standard data science imports
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

# Visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Statistics packages
import pylab
from pylab import rcParams
import statsmodels.api as sm
import statistics
from scipy import stats

# Scikit-learn
import sklearn
from sklearn import preprocessing
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report


# Import chisquare from SciPy.stats
from scipy.stats import chisquare
from scipy.stats import chi2_contingency

# Ignore Warning Code
import warnings
warnings.filterwarnings('ignore')
```

[64]:
```
# Change color of Matplotlib font
import matplotlib as mpl

COLOR = 'white'
mpl.rcParams['text.color'] = COLOR
mpl.rcParams['axes.labelcolor'] = COLOR
mpl.rcParams['xtick.color'] = COLOR
```

```
mpl.rcParams['ytick.color'] = COLOR
```

[65]:
```
# Load data set into Pandas dataframe
churn_df = pd.read_csv('churn_clean.csv')

# Rename last 8 survey columns for better description of variables
churn_df.rename(columns = {'Item1':'TimelyResponse',
                    'Item2':'Fixes',
                     'Item3':'Replacements',
                     'Item4':'Reliability',
                     'Item5':'Options',
                     'Item6':'Respectfulness',
                     'Item7':'Courteous',
                     'Item8':'Listening'},
          inplace=True)
```

[66]:
```
# Display Churn dataframe
churn_df
```

[66]:
```
      CaseOrder Customer_id  ... Courteous Listening
0             1     K409198  ...         3         4
1             2     S120509  ...         4         4
2             3     K191035  ...         3         3
3             4      D90850  ...         3         3
4             5     K662701  ...         4         5
...         ...         ...  ...       ...       ...
9995       9996     M324793  ...         2         3
9996       9997     D861732  ...         2         5
9997       9998     I243405  ...         4         5
9998       9999     I641617  ...         5         4
9999      10000      T38070  ...         4         1

[10000 rows x 50 columns]
```

[67]:
```
# List of Dataframe Columns
df = churn_df.columns
print(df)
```

```
Index(['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
       'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job',
       'Children', 'Age', 'Income', 'Marital', 'Gender', 'Churn',
       'Outage_sec_perweek', 'Email', 'Contacts', 'Yearly_equip_failure',
       'Techie', 'Contract', 'Port_modem', 'Tablet', 'InternetService',
       'Phone', 'Multiple', 'OnlineSecurity', 'OnlineBackup',
       'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
       'PaperlessBilling', 'PaymentMethod', 'Tenure', 'MonthlyCharge',
       'Bandwidth_GB_Year', 'TimelyResponse', 'Fixes', 'Replacements',
       'Reliability', 'Options', 'Respectfulness', 'Courteous', 'Listening'],
      dtype='object')
```

[68]:
```
# Find number of records and columns of dataset
churn_df.shape
```

[68]:
```
(10000, 50)
```

[69]:
```
# Describe Churn dataset statistics
churn_df.describe()
```

[69]:
```
         CaseOrder           Zip  ...      Courteous      Listening
count  10000.00000  10000.000000  ...  10000.000000  10000.000000
mean    5000.50000  49153.319600  ...      3.509500      3.495600
std     2886.89568  27532.196108  ...      1.028502      1.028633
min        1.00000    601.000000  ...      1.000000      1.000000
25%     2500.75000  26292.500000  ...      3.000000      3.000000
50%     5000.50000  48869.500000  ...      4.000000      3.000000
75%     7500.25000  71866.500000  ...      4.000000      4.000000
max    10000.00000  99929.000000  ...      7.000000      8.000000

[8 rows x 23 columns]
```

[70]:
```
# Remove less meaningful demographic variables from statistics description
churn_df = churn_df.drop(columns=['CaseOrder', 'Customer_id', 'Interaction',␣
 ↪'UID', 'City',
                             'State', 'County', 'Zip', 'Lat', 'Lng',␣
 ↪'Population',
                             'Area', 'TimeZone', 'Job', 'Marital',␣
 ↪'PaymentMethod'])
churn_df.describe()
```

[70]:
```
         Children           Age  ...      Courteous      Listening
count  10000.0000  10000.000000  ...  10000.000000  10000.000000
mean       2.0877     53.078400  ...      3.509500      3.495600
std        2.1472     20.698882  ...      1.028502      1.028633
min        0.0000     18.000000  ...      1.000000      1.000000
25%        0.0000     35.000000  ...      3.000000      3.000000
50%        1.0000     53.000000  ...      4.000000      3.000000
75%        3.0000     71.000000  ...      4.000000      4.000000
max       10.0000     89.000000  ...      7.000000      8.000000

[8 rows x 18 columns]
```

[71]:
```
# Discover missing data points within dataset
data_nulls = churn_df.isnull().sum()
print(data_nulls)
```

```
Children                0
Age                     0
Income                  0
Gender                  0
Churn                   0
```

```
Outage_sec_perweek      0
Email                   0
Contacts                0
Yearly_equip_failure    0
Techie                  0
Contract                0
Port_modem              0
Tablet                  0
InternetService         0
Phone                   0
Multiple                0
OnlineSecurity          0
OnlineBackup            0
DeviceProtection        0
TechSupport             0
StreamingTV             0
StreamingMovies         0
PaperlessBilling        0
Tenure                  0
MonthlyCharge           0
Bandwidth_GB_Year       0
TimelyResponse          0
Fixes                   0
Replacements            0
Reliability             0
Options                 0
Respectfulness          0
Courteous               0
Listening               0
dtype: int64
```

Dummy variable data preparation

[72]:
```
churn_df['DummyGender'] = [1 if v == 'Male' else 0 for v in churn_df['Gender']]
churn_df['DummyChurn'] = [1 if v == 'Yes' else 0 for v in churn_df['Churn']]␣
 ↪### If the customer left (churned) they get a '1'
churn_df['DummyTechie'] = [1 if v == 'Yes' else 0 for v in churn_df['Techie']]
churn_df['DummyContract'] = [1 if v == 'Two Year' else 0 for v in␣
 ↪churn_df['Contract']]
churn_df['DummyPort_modem'] = [1 if v == 'Yes' else 0 for v in␣
 ↪churn_df['Port_modem']]
churn_df['DummyTablet'] = [1 if v == 'Yes' else 0 for v in churn_df['Tablet']]
churn_df['DummyInternetService'] = [1 if v == 'Fiber Optic' else 0 for v in␣
 ↪churn_df['InternetService']]
churn_df['DummyPhone'] = [1 if v == 'Yes' else 0 for v in churn_df['Phone']]
churn_df['DummyMultiple'] = [1 if v == 'Yes' else 0 for v in␣
 ↪churn_df['Multiple']]
```

```
churn_df['DummyOnlineSecurity'] = [1 if v == 'Yes' else 0 for v in␣
↪churn_df['OnlineSecurity']]
churn_df['DummyOnlineBackup'] = [1 if v == 'Yes' else 0 for v in␣
↪churn_df['OnlineBackup']]
churn_df['DummyDeviceProtection'] = [1 if v == 'Yes' else 0 for v in␣
↪churn_df['DeviceProtection']]
churn_df['DummyTechSupport'] = [1 if v == 'Yes' else 0 for v in␣
↪churn_df['TechSupport']]
churn_df['DummyStreamingTV'] = [1 if v == 'Yes' else 0 for v in␣
↪churn_df['StreamingTV']]
churn_df['StreamingMovies'] = [1 if v == 'Yes' else 0 for v in␣
↪churn_df['StreamingMovies']]
churn_df['DummyPaperlessBilling'] = [1 if v == 'Yes' else 0 for v in␣
↪churn_df['PaperlessBilling']]
```

[73]:
```
churn_df.head()
```

[73]:
```
   Children  Age  ...  DummyStreamingTV  DummyPaperlessBilling
0         0   68  ...                 0                      1
1         1   27  ...                 1                      1
2         4   50  ...                 0                      1
3         1   48  ...                 1                      1
4         0   83  ...                 1                      0

[5 rows x 49 columns]
```

[74]:
```
# Drop original categorical features from dataframe
churn_df = churn_df.drop(columns=['Gender', 'Churn', 'Techie', 'Contract',␣
↪'Port_modem', 'Tablet',
                                  'InternetService', 'Phone', 'Multiple',␣
↪'OnlineSecurity',
                                  'OnlineBackup', 'DeviceProtection',␣
↪'TechSupport',
                                  'StreamingTV', 'StreamingMovies',␣
↪'PaperlessBilling'])
churn_df.describe()
```

[74]:
```
         Children           Age  ...  DummyStreamingTV  DummyPaperlessBilling
count  10000.0000  10000.000000  ...      10000.000000           10000.000000
mean       2.0877     53.078400  ...          0.492900               0.588200
std        2.1472     20.698882  ...          0.499975               0.492184
min        0.0000     18.000000  ...          0.000000               0.000000
25%        0.0000     35.000000  ...          0.000000               0.000000
50%        1.0000     53.000000  ...          0.000000               1.000000
75%        3.0000     71.000000  ...          1.000000               1.000000
max       10.0000     89.000000  ...          1.000000               1.000000

[8 rows x 33 columns]
```

[75]:
```
df = churn_df.columns
print(df)
```

```
Index(['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts',
       'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year',
       'TimelyResponse', 'Fixes', 'Replacements', 'Reliability', 'Options',
       'Respectfulness', 'Courteous', 'Listening', 'DummyGender', 'DummyChurn',
       'DummyTechie', 'DummyContract', 'DummyPort_modem', 'DummyTablet',
       'DummyInternetService', 'DummyPhone', 'DummyMultiple',
       'DummyOnlineSecurity', 'DummyOnlineBackup', 'DummyDeviceProtection',
       'DummyTechSupport', 'DummyStreamingTV', 'DummyPaperlessBilling'],
      dtype='object')
```

[76]:
```
# Move DummyChurn to end of dataset as target
churn_df = churn_df[['Children', 'Age', 'Income', 'Outage_sec_perweek',␣
 ↪'Email', 'Contacts',
       'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year',
        'TimelyResponse', 'Fixes', 'Replacements',
       'Reliability', 'Options', 'Respectfulness', 'Courteous', 'Listening',
       'DummyGender', 'DummyTechie', 'DummyContract',
       'DummyPort_modem', 'DummyTablet', 'DummyInternetService', 'DummyPhone',
       'DummyMultiple', 'DummyOnlineSecurity', 'DummyOnlineBackup',
       'DummyDeviceProtection', 'DummyTechSupport', 'DummyStreamingTV',
       'DummyPaperlessBilling', 'DummyChurn']]
```

[77]:
```
df = churn_df.columns
print(df)
```

```
Index(['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts',
       'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year',
       'TimelyResponse', 'Fixes', 'Replacements', 'Reliability', 'Options',
       'Respectfulness', 'Courteous', 'Listening', 'DummyGender',
       'DummyTechie', 'DummyContract', 'DummyPort_modem', 'DummyTablet',
       'DummyInternetService', 'DummyPhone', 'DummyMultiple',
       'DummyOnlineSecurity', 'DummyOnlineBackup', 'DummyDeviceProtection',
       'DummyTechSupport', 'DummyStreamingTV', 'DummyPaperlessBilling',
       'DummyChurn'],
      dtype='object')
```

C4. Visualizations:

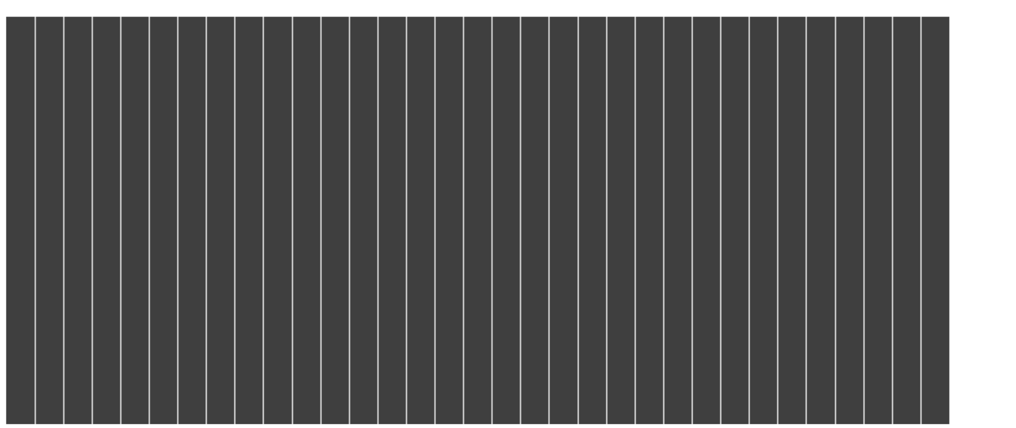
[78]:
```
# Visualize missing values in dataset

# Install appropriate library
!pip install missingno

# Importing the libraries
import missingno as msno
```

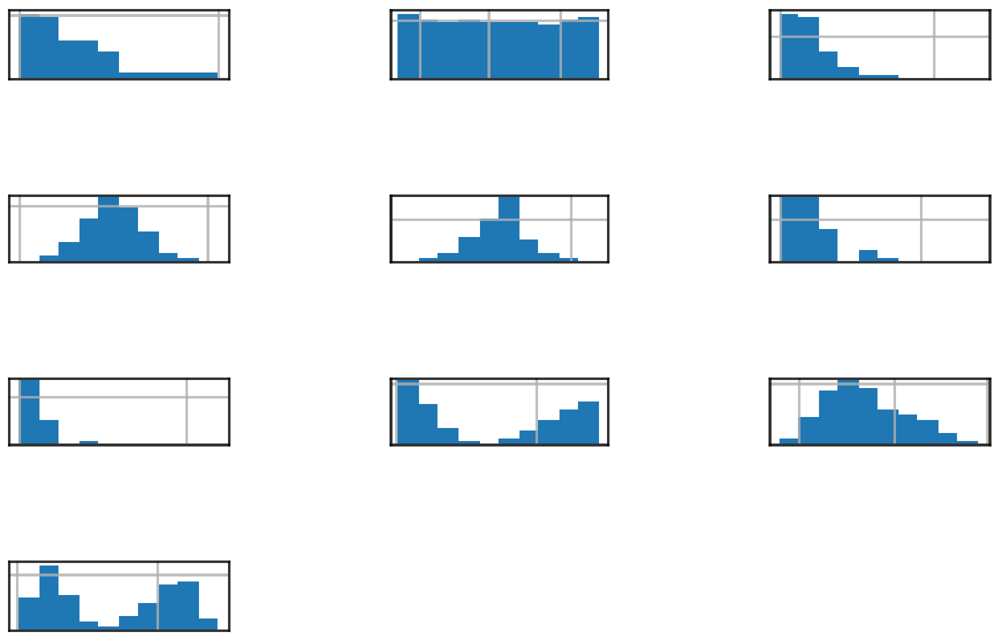
```python
# Visualize missing values as a matrix
msno.matrix(churn_df);
```

```
Requirement already satisfied: missingno in /usr/local/lib/python3.7/dist-
packages (0.5.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages
(from missingno) (0.11.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-
packages (from missingno) (3.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages
(from missingno) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from missingno) (1.19.5)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-
packages (from matplotlib->missingno) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7
/dist-packages (from matplotlib->missingno) (1.3.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7
/dist-packages (from matplotlib->missingno) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->missingno) (2.4.7)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from cycler>=0.10->matplotlib->missingno) (1.15.0)
Requirement already satisfied: pandas>=0.23 in /usr/local/lib/python3.7/dist-
packages (from seaborn->missingno) (1.1.5)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-
packages (from pandas>=0.23->seaborn->missingno) (2018.9)
```

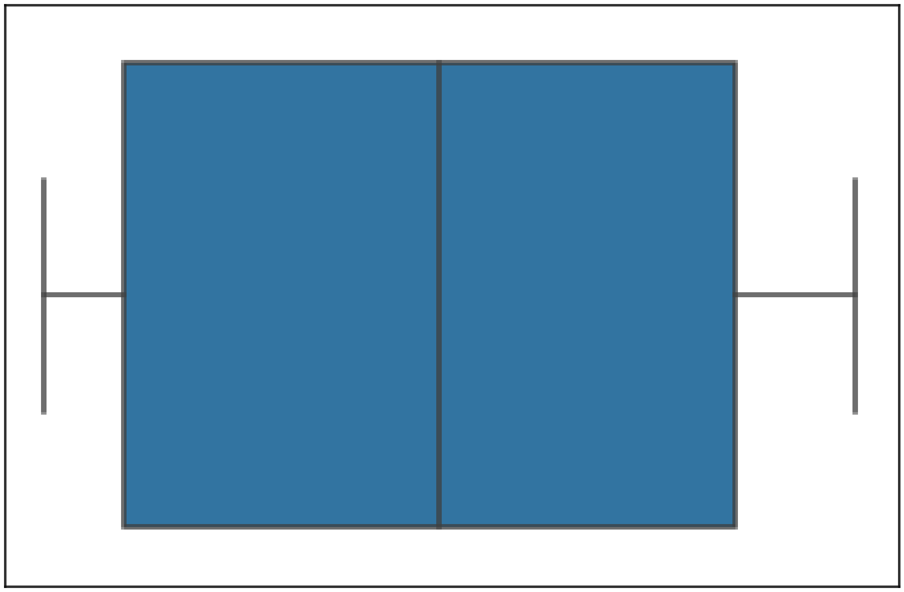


Univariate Statistics

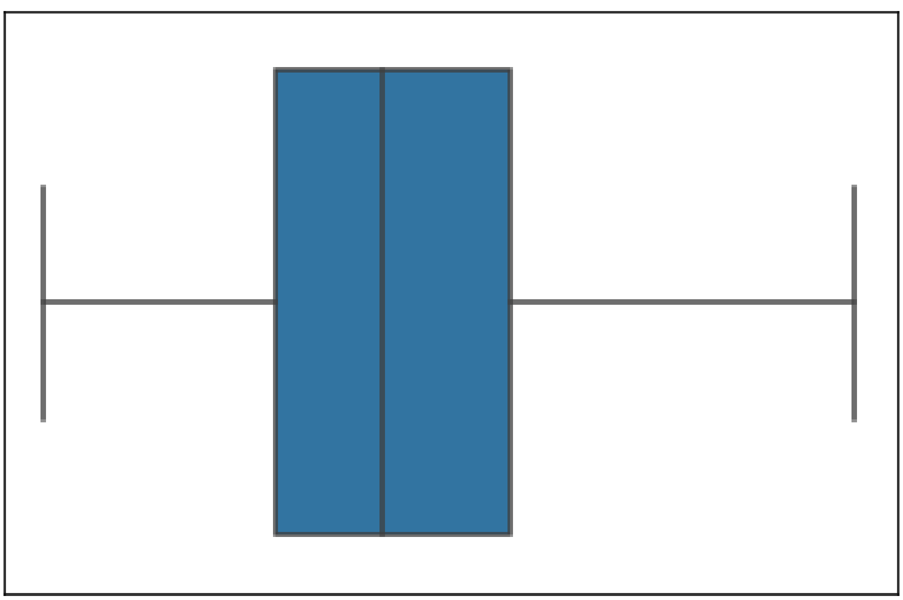
[79]:
```
# Create histograms of contiuous variables
churn_df[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email',
          'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
          'Bandwidth_GB_Year']].hist()
plt.savefig('churn_pyplot.jpg')
plt.tight_layout()
```



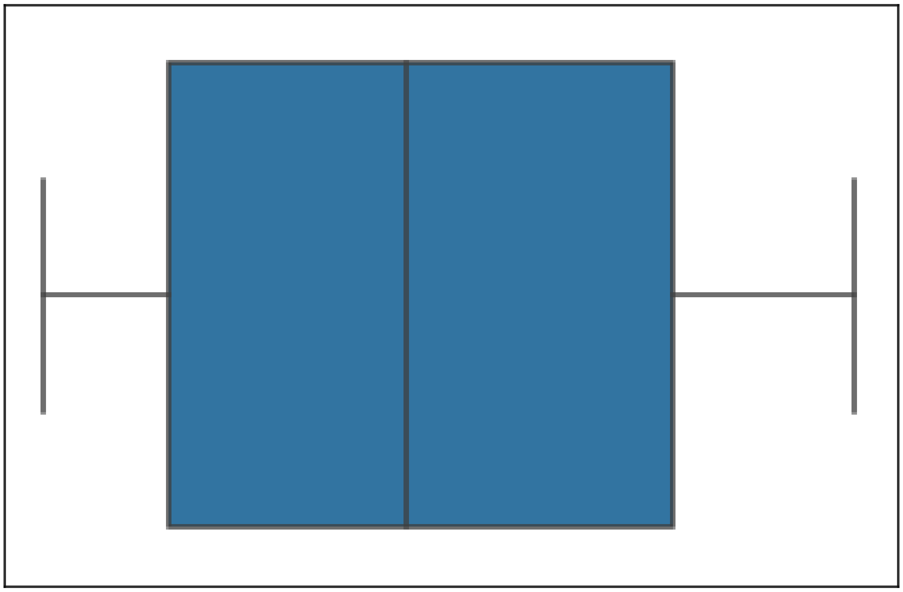
[80]:
```
# Create Seaborn boxplots for continuous variables
sns.boxplot('Tenure', data = churn_df)
plt.show()
```



[81]:
```
sns.boxplot('MonthlyCharge', data = churn_df)
plt.show()
```



[82]:
```
sns.boxplot('Bandwidth_GB_Year', data = churn_df)
plt.show()
```



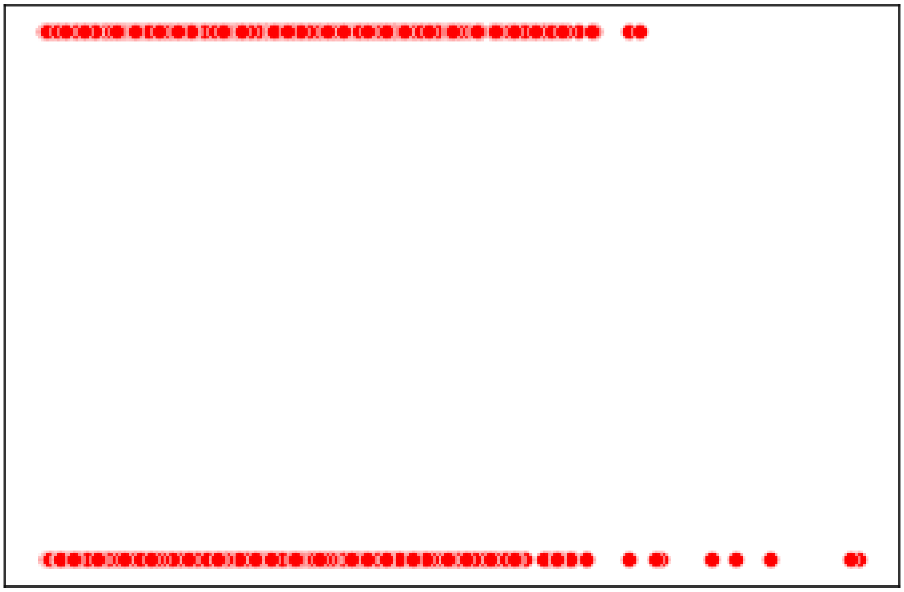
Anomalies It appears that anomolies have been removed from the supplied dataset, churn_clean.csv. There are no remaining outliers.

Bivariate Statistics

[83]:
```
# Run scatterplots to show direct or inverse relationships between target &␣
↪independent variables
sns.scatterplot(x=churn_df['Children'], y=churn_df['DummyChurn'], color='red')
plt.show();
```

[84]:
```
sns.scatterplot(x=churn_df['Age'], y=churn_df['DummyChurn'], color='red')
plt.show();
```

[85]:
```
sns.scatterplot(x=churn_df['Income'], y=churn_df['DummyChurn'], color='red')
plt.show();
```

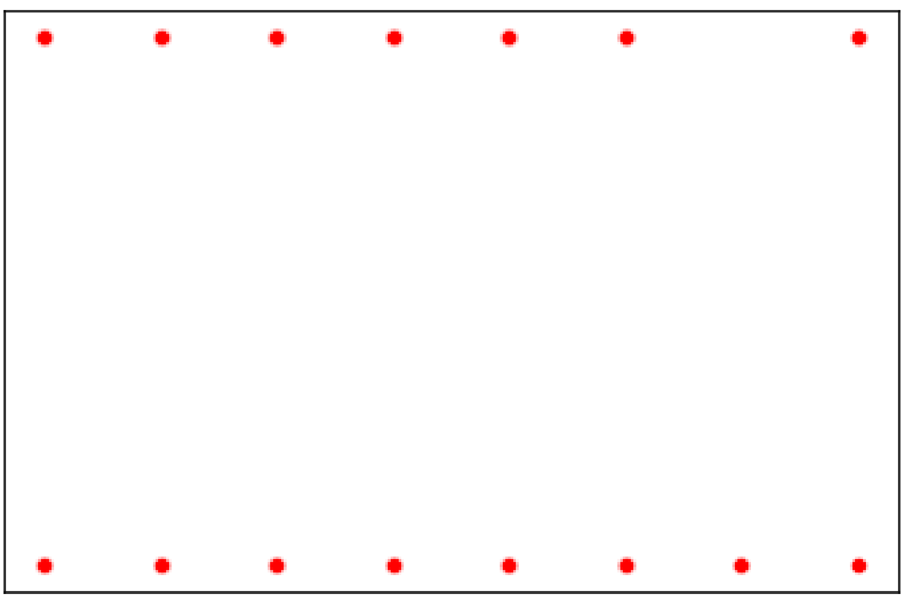


[86]:
```
sns.scatterplot(x=churn_df['DummyGender'], y=churn_df['DummyChurn'],␣
↪color='red')
plt.show();
```

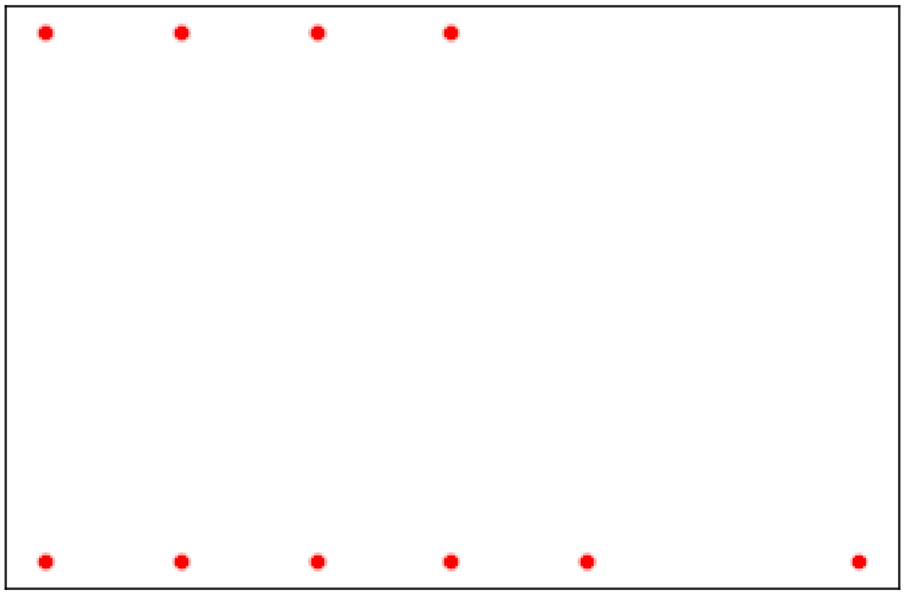
[87]:
```
sns.scatterplot(x=churn_df['Outage_sec_perweek'], y=churn_df['DummyChurn'],␣
 ↪color='red')
plt.show();
```

[88]:
```
sns.scatterplot(x=churn_df['Email'], y=churn_df['DummyChurn'], color='red')
plt.show();
```

[89]:
```
sns.scatterplot(x=churn_df['Contacts'], y=churn_df['DummyChurn'], color='red')
plt.show();
```

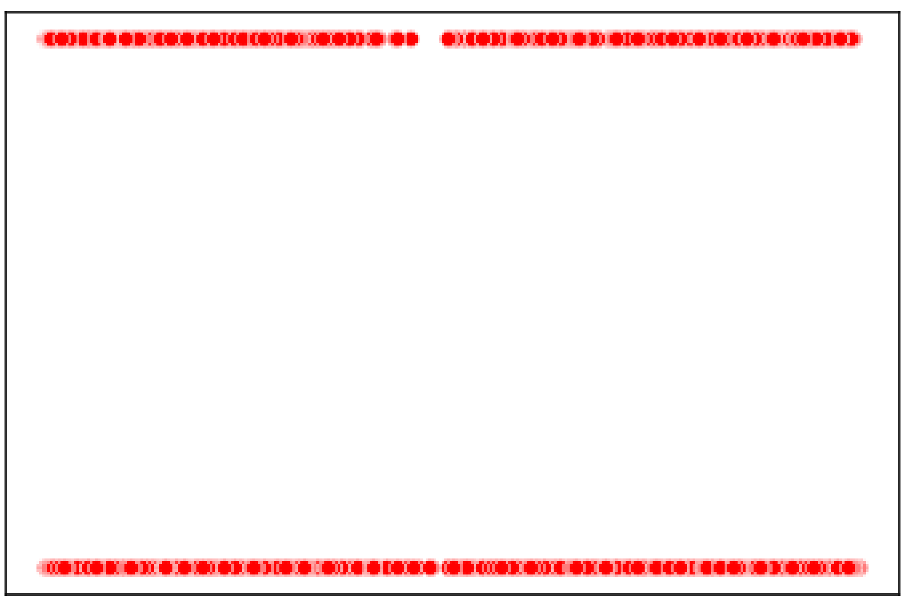


[90]:
```
sns.scatterplot(x=churn_df['Yearly_equip_failure'], y=churn_df['DummyChurn'],␣
 ↪color='red')
plt.show();
```



[91]:
```
sns.scatterplot(x=churn_df['DummyTechie'], y=churn_df['DummyChurn'],␣
 ↪color='red')
plt.show();
```

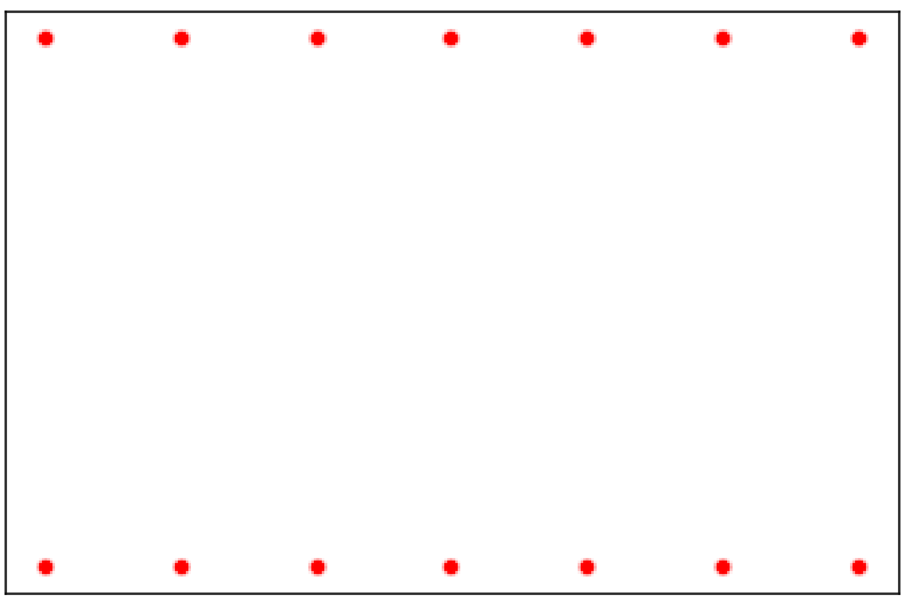
[92]:
```
sns.scatterplot(x=churn_df['Tenure'], y=churn_df['DummyChurn'], color='red')
plt.show();
```



[93]:
```
sns.scatterplot(x=churn_df['MonthlyCharge'], y=churn_df['DummyChurn'],␣
↪color='red')
plt.show();
```

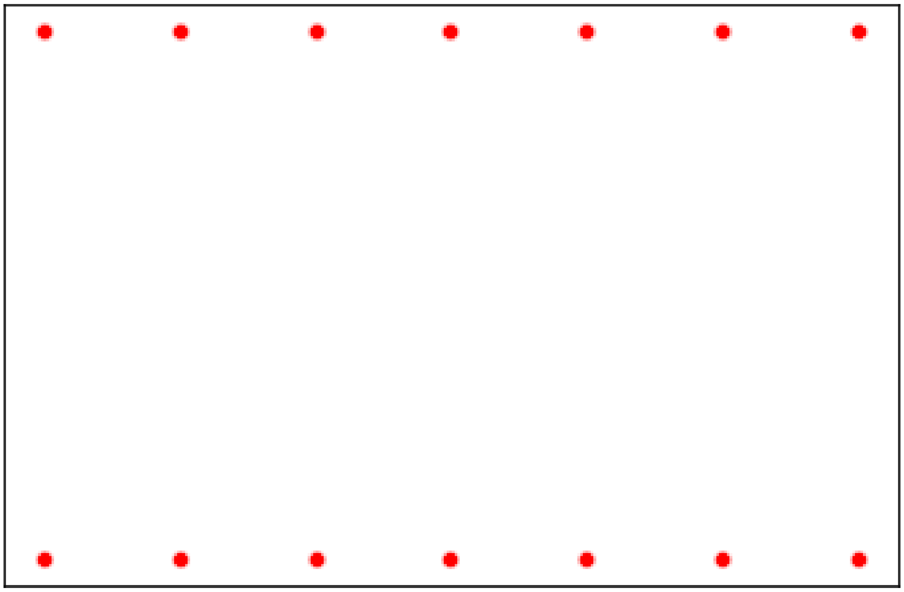
[94]:
```
sns.scatterplot(x=churn_df['Bandwidth_GB_Year'], y=churn_df['DummyChurn'],␣
↪color='red')
plt.show();
```

[95]:
```
sns.scatterplot(x=churn_df['TimelyResponse'], y=churn_df['DummyChurn'],␣
↪color='red')
plt.show();
```



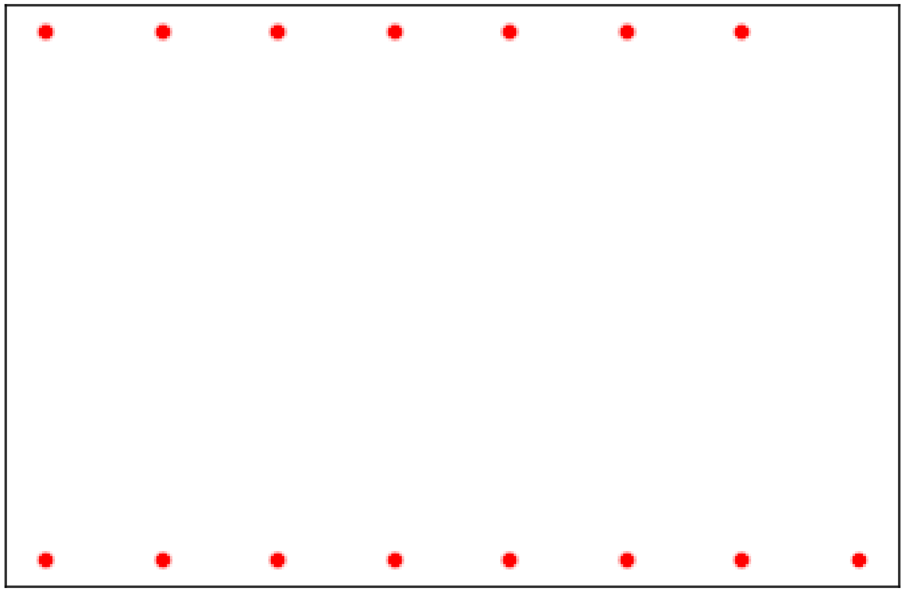
[96]:

```
sns.scatterplot(x=churn_df['Fixes'], y=churn_df['DummyChurn'], color='red')
plt.show();
```

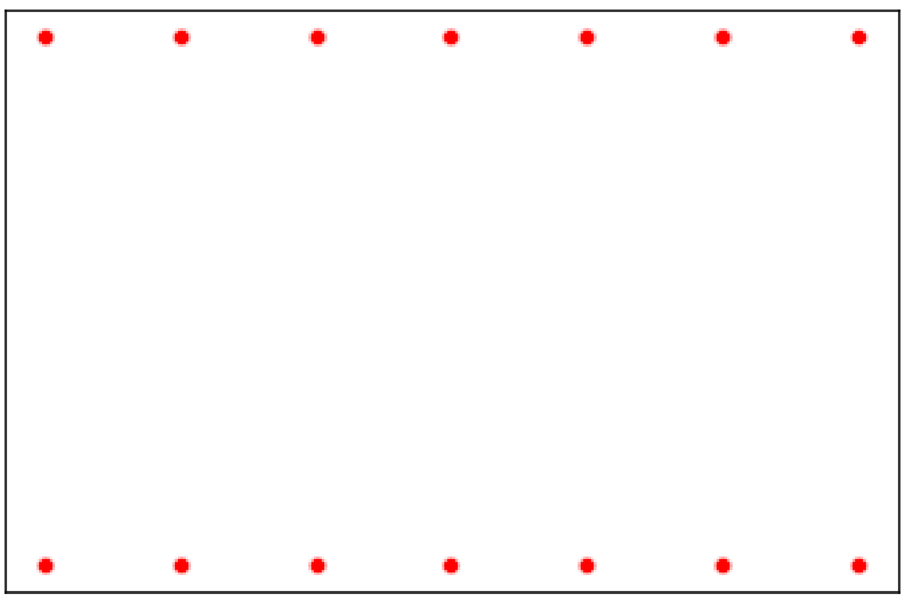


[97]:

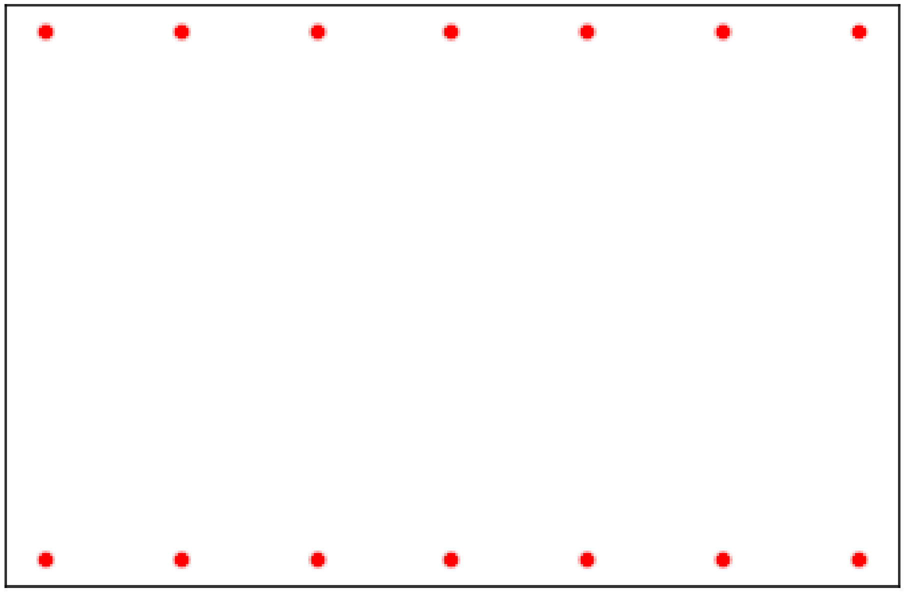
```
sns.scatterplot(x=churn_df['Replacements'], y=churn_df['DummyChurn'],␣
↪color='red')
plt.show();
```



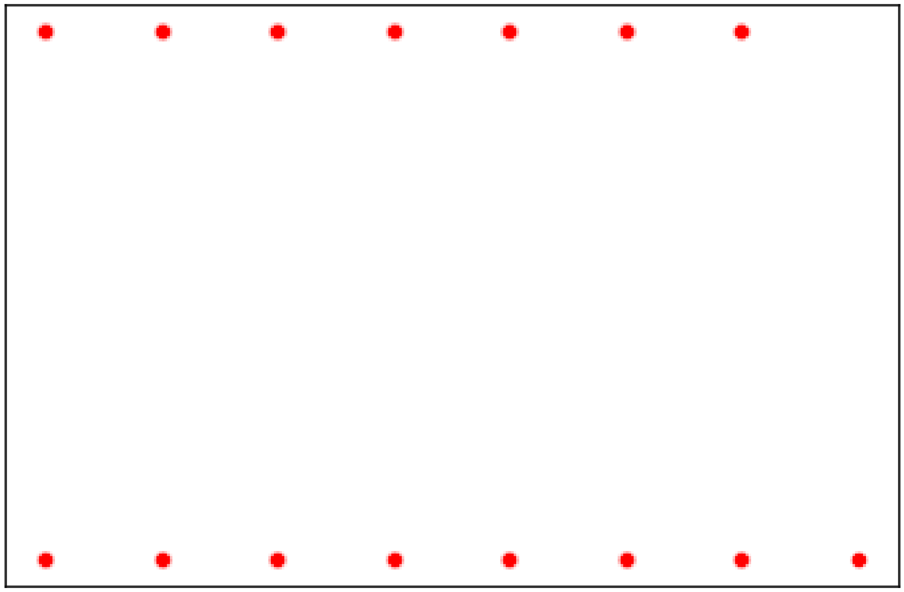
[98]:
```
sns.scatterplot(x=churn_df['Reliability'], y=churn_df['DummyChurn'],␣
 ↪color='red')
plt.show();
```



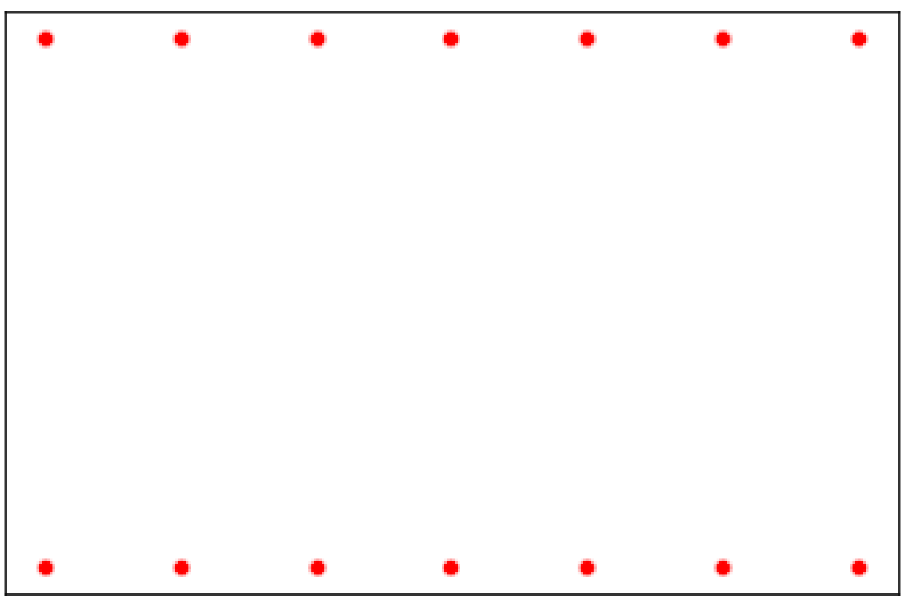
[99]:
```
sns.scatterplot(x=churn_df['Options'], y=churn_df['DummyChurn'], color='red')
plt.show();
```



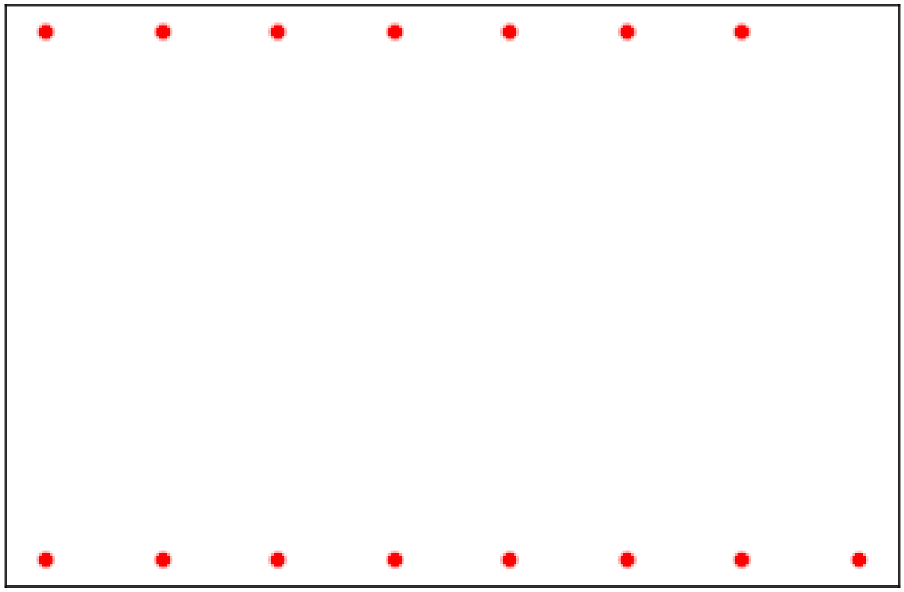
[100]:
```
sns.scatterplot(x=churn_df['Respectfulness'], y=churn_df['DummyChurn'],␣
↪color='red')
plt.show();
```



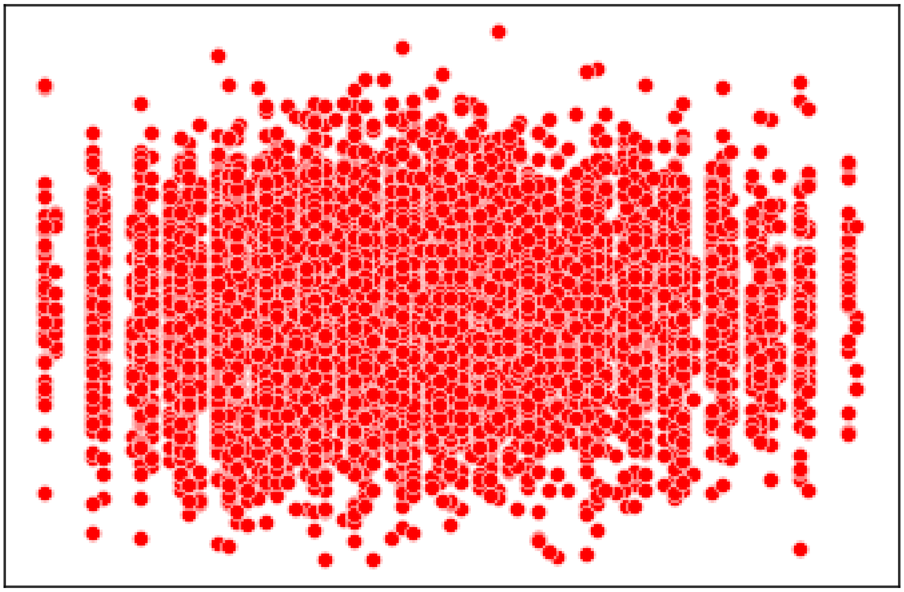
[101]:
```
sns.scatterplot(x=churn_df['Courteous'], y=churn_df['DummyChurn'], color='red')
plt.show();
```



[102]:
```
sns.scatterplot(x=churn_df['Listening'], y=churn_df['DummyChurn'], color='red')
plt.show();
```



[103]:
```
sns.scatterplot(x=churn_df['MonthlyCharge'], y=churn_df['Outage_sec_perweek'],␣
↪color='red')
plt.show();
```



Scatterplot Summary These scatterplots suggest no correlation between a customer churning (Churn = 1) & any of our continous user data points or categorical responses to survey data points.

C5. Prepared Dataset:

[104]:
```
# Extract Clean dataset
churn_df.to_csv('churn_prepared_log.csv')
```

D1. Initial Model

[105]:
```
"""Develop the initial estimated regression equation that could be used to␣
↪predict the probability of customer churn, given the only continuous␣
↪variables"""
churn_df = pd.read_csv('churn_prepared_log.csv')
churn_df['intercept'] = 1
churn_df = pd.get_dummies(churn_df, drop_first=True)

churn_logit_model = sm.Logit(churn_df['DummyChurn'], churn_df[['Children',␣
↪'Age',
                                                                      'Income',
                                                                     ␣
↪'Outage_sec_perweek',
                                                                      'Email',␣
↪'Contacts',
```

```
                                                                                ␣
 ↪'Yearly_equip_failure',
                                                                                 'Tenure',␣
 ↪'MonthlyCharge',
                                                                                ␣
 ↪'Bandwidth_GB_Year',
                                                                                ␣
 ↪'TimelyResponse', 'Fixes',
                                                                                 'Replacements',␣
 ↪'Reliability',
                                                                                 'Options',␣
 ↪'Respectfulness',
                                                                                 'Courteous',␣
 ↪'Listening',
                                                                                 'intercept']]).
 ↪fit()
print(churn_logit_model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.319573
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:            DummyChurn   No. Observations:                10000
Model:                         Logit   Df Residuals:                     9981
Method:                          MLE   Df Model:                           18
Date:               Thu, 22 Jul 2021   Pseudo R-squ.:                  0.4473
Time:                       21:52:18   Log-Likelihood:                -3195.7
converged:                      True   LL-Null:                       -5782.2
Covariance Type:           nonrobust   LLR p-value:                     0.000
================================================================================
========
                           coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
Children                -0.0980      0.016     -6.318      0.000      -0.128
-0.068
Age                      0.0114      0.002      7.130      0.000       0.008
0.015
Income                5.015e-07   1.12e-06      0.450      0.653   -1.68e-06
2.69e-06
Outage_sec_perweek      -0.0009      0.011     -0.087      0.931      -0.022
0.020
Email                    0.0018      0.010      0.169      0.866      -0.019
0.022
Contacts                 0.0243      0.032      0.764      0.445      -0.038
```

```
0.087
Yearly_equip_failure    -0.0267      0.050     -0.539      0.590      -0.124
0.071
Tenure                  -0.3156      0.012    -25.482      0.000      -0.340
-0.291
MonthlyCharge            0.0262      0.001     27.344      0.000       0.024
0.028
Bandwidth_GB_Year        0.0029      0.000     20.156      0.000       0.003
0.003
TimelyResponse          -0.0201      0.045     -0.447      0.655      -0.108
0.068
Fixes                   -0.0162      0.042     -0.384      0.701      -0.099
0.067
Replacements            -0.0053      0.039     -0.138      0.890      -0.081
0.070
Reliability             -0.0376      0.034     -1.096      0.273      -0.105
0.030
Options                 -0.0439      0.036     -1.223      0.221      -0.114
0.026
Respectfulness          -0.0044      0.037     -0.119      0.906      -0.076
0.068
Courteous               -0.0203      0.035     -0.580      0.562      -0.089
0.048
Listening               -0.0024      0.033     -0.071      0.943      -0.067
0.062
intercept               -5.4290      0.369    -14.709      0.000      -6.152
-4.706
================================================================================
========
```

Dummy Variables Now, we will run a model including all encoded categorical dummy variables.

[106]:
```
"""Model including all dummy variables"""
churn_df = pd.read_csv('churn_prepared_log.csv')
churn_df['intercept'] = 1
churn_df = pd.get_dummies(churn_df, drop_first=True)

churn_logit_model2 = sm.Logit(churn_df['DummyChurn'], churn_df[['Children',␣
 ↪'Age',
                                                                'Income',␣
 ↪'Outage_sec_perweek',
                                                                'Email',␣
 ↪'Contacts',
                                                               ␣
 ↪'Yearly_equip_failure',
                                                                'DummyTechie',␣
 ↪'DummyContract',
```

```
↪'DummyPort_modem', 'DummyTablet',
↪'DummyInternetService', 'DummyPhone',
                                                          'DummyMultiple',␣
↪'DummyOnlineSecurity',
↪'DummyOnlineBackup', 'DummyDeviceProtection',
↪'DummyTechSupport', 'DummyStreamingTV',
↪'DummyPaperlessBilling',
                                                          'Tenure',␣
↪'MonthlyCharge', 'Bandwidth_GB_Year',
↪'TimelyResponse', 'Fixes',
                                                          'Replacements',␣
↪'Reliability',
                                                          'Options',␣
↪'Respectfulness',
                                                          'Courteous',␣
↪'Listening',
                                                          'intercept']]).
↪fit()
print(churn_logit_model2.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.271990
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:            DummyChurn   No. Observations:                10000
Model:                         Logit   Df Residuals:                     9968
Method:                          MLE   Df Model:                           31
Date:               Thu, 22 Jul 2021   Pseudo R-squ.:                  0.5296
Time:                       21:52:20   Log-Likelihood:                -2719.9
converged:                      True   LL-Null:                       -5782.2
Covariance Type:           nonrobust   LLR p-value:                     0.000
================================================================================
=========
                            coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---------
Children                 -0.0395      0.018     -2.232      0.026      -0.074
-0.005
```

Age	0.0069	0.002	3.659	0.000	0.003	0.011
Income	1.199e-07	1.22e-06	0.099	0.921	-2.26e-06	2.5e-06
Outage_sec_perweek	0.0020	0.011	0.176	0.860	-0.021	0.025
Email	-0.0015	0.011	-0.133	0.894	-0.024	0.021
Contacts	0.0301	0.035	0.871	0.384	-0.038	0.098
Yearly_equip_failure	-0.0308	0.054	-0.570	0.569	-0.137	0.075
DummyTechie	0.7956	0.089	8.960	0.000	0.622	0.970
DummyContract	-2.2950	0.104	-22.135	0.000	-2.498	-2.092
DummyPort_modem	0.1610	0.068	2.353	0.019	0.027	0.295
DummyTablet	-0.0796	0.074	-1.071	0.284	-0.225	0.066
DummyInternetService	-1.4252	0.126	-11.314	0.000	-1.672	-1.178
DummyPhone	-0.3157	0.117	-2.707	0.007	-0.544	-0.087
DummyMultiple	-0.2908	0.080	-3.646	0.000	-0.447	-0.134
DummyOnlineSecurity	-0.3280	0.074	-4.452	0.000	-0.472	-0.184
DummyOnlineBackup	-0.5125	0.074	-6.931	0.000	-0.657	-0.368
DummyDeviceProtection	-0.4100	0.071	-5.764	0.000	-0.549	-0.271
DummyTechSupport	-0.3461	0.073	-4.717	0.000	-0.490	-0.202
DummyStreamingTV	0.0311	0.083	0.374	0.708	-0.132	0.194
DummyPaperlessBilling	0.1126	0.070	1.618	0.106	-0.024	0.249
Tenure	-0.2043	0.021	-9.693	0.000	-0.246	-0.163
MonthlyCharge	0.0461	0.002	24.371	0.000	0.042	0.050
Bandwidth_GB_Year	0.0013	0.000	5.215	0.000	0.001	0.002
TimelyResponse	-0.0167	0.049	-0.342	0.732	-0.112	0.079
Fixes	0.0143	0.046	0.311	0.755	-0.076	0.104

```
Replacements                -0.0158      0.042     -0.377      0.706      -0.098
0.066
Reliability                 -0.0250      0.037     -0.673      0.501      -0.098
0.048
Options                     -0.0341      0.039     -0.877      0.380      -0.110
0.042
Respectfulness              -0.0309      0.040     -0.776      0.438      -0.109
0.047
Courteous                    0.0047      0.038      0.124      0.901      -0.070
0.079
Listening                   -0.0090      0.036     -0.251      0.802      -0.079
0.061
intercept                   -5.8583      0.425    -13.793      0.000      -6.691
-5.026
================================================================================
=========
```

Early Model Comparison Following the second run of our MLE model, our pseudo R went up from 0.4473 to 0.5296 as we added in our categorical dummy variables to our continuous variables. We will take that as a good sign that some of the explanation of our variance is within the categorical data points. We will use those 31 variables as our initial regression equation.

Initial Multiple Linear Regression Model With 31 independent variables (18 continuous & 13 categorical): y = -5.8583 + (-0.0395 * Children) + (0.0069 * Age) + (1.199e-07 * Income) + (-0.0020 * Outage_sec_perweek) + (-0.0015 * Email) + (0.0301 * Contacts) + (-0.0308 * Yearly_equip_failure) + (0.7956 * DummyTechie) + (-2.295 * DummyContract) + (0.161 * DummyPort_modem) + (-0.0796 * DummyTablet) + (-1.4252 * DummyInternetService) + (-0.3157 * DummyPhone) + (-0.2908 * DummyMultiple) + (-0.3280 * DummyOnlineSecurity) + (-0.5125 * DummyOnlineBackup) + (-0.41 * DummyDeviceProtection) + (-0.3461 * DummyTechSupport) + (0.0311 * DummyStreamingTV) + (0.1126 * DummyPaperlessBilling) + (-0.2043 * Tenure) + (0.0461 * MonthlyCharge) + (0.0013 * Bandwidth_GB_Year) + (-0.0167 * TimelyResponse) + (0.0143 * Fixes) + (-0.0158 * Replacements) + (-0.025 * Reliability) + (-0.0341 * Options) + (-0.0309 * Respectfulness) + (0.0047 * Courteous) + (-0.009 * Listening)

D2. Justification of Model Reduction Based on the above MLE model we created, we have a pseudo R value = 0.5296, which is clearly not very good for the variance of our model. Also, coefficients on the above model are very low (less than 0.5) with the exception of variables DummyTechie, DummyContract, DummyInternetService & DummyOnlineBackup. Those variables also have p-values less than 0.000 & appear, therefore, significant.
Subsequently, let us choose a p-value of 0.05 & include all variables with p-values 0.05. We will remove any predictor variable with a p-value greater than 0.05 as not statistically significant to our model.

Our next MLE run will include the continuous predictor variables:

- Age
- Tenure

- MonthlyCharge
- Bandwidth_GB_Year

And, categorical predictor variables:

- DummyTechie
- DummyContract
- DummyPort_modem
- DummyInternetService
- DummyPhone
- DummyMultiple
- DummyOnlineSecurity
- DummyOnlineBackup
- DummyDeviceProtection
- DummyTechSupport

We will run that reduced number of predictor variables against our DummyChurn dependent variable in another MLE model.

D3. Reduced Multiple Regression Model

[107]:
```
# Run reduced OLS multiple regression
churn_df['intercept'] = 1
churn_logit_model_reduced = sm.Logit(churn_df['DummyChurn'],
                                     churn_df[['Children', 'Age',␣
 ↪'DummyTechie', 'DummyContract', 'DummyPort_modem',
                                               'DummyInternetService',␣
 ↪'DummyPhone','DummyMultiple',
                                               'DummyOnlineSecurity',␣
 ↪'DummyOnlineBackup', 'DummyDeviceProtection',
                                               'DummyTechSupport', 'Tenure',␣
 ↪'MonthlyCharge', 'Bandwidth_GB_Year',
                                               'intercept']]).fit()
print(churn_logit_model_reduced.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.272362
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:             DummyChurn   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9984
Method:                           MLE   Df Model:                           15
Date:                Thu, 22 Jul 2021   Pseudo R-squ.:                  0.5290
Time:                        21:52:24   Log-Likelihood:                -2723.6
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
```

```
=========
                             coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---------
Children                  -0.0391      0.018     -2.221      0.026      -0.074
-0.005
Age                        0.0070      0.002      3.735      0.000       0.003
0.011
DummyTechie                0.7970      0.089      8.996      0.000       0.623
0.971
DummyContract             -2.2895      0.103    -22.136      0.000      -2.492
-2.087
DummyPort_modem            0.1598      0.068      2.339      0.019       0.026
0.294
DummyInternetService      -1.4240      0.125    -11.359      0.000      -1.670
-1.178
DummyPhone                -0.3193      0.116     -2.749      0.006      -0.547
-0.092
DummyMultiple             -0.2964      0.077     -3.857      0.000      -0.447
-0.146
DummyOnlineSecurity       -0.3303      0.073     -4.497      0.000      -0.474
-0.186
DummyOnlineBackup         -0.5146      0.072     -7.125      0.000      -0.656
-0.373
DummyDeviceProtection     -0.4075      0.070     -5.790      0.000      -0.545
-0.270
DummyTechSupport          -0.3555      0.073     -4.892      0.000      -0.498
-0.213
Tenure                    -0.2049      0.021     -9.770      0.000      -0.246
-0.164
MonthlyCharge              0.0463      0.002     25.620      0.000       0.043
0.050
Bandwidth_GB_Year          0.0013      0.000      5.279      0.000       0.001
0.002
intercept                 -6.1973      0.236    -26.280      0.000      -6.659
-5.735
================================================================================
=========
```

Reduced Logistic Regression Model With 15 independent variables (5 continuous & 10 categorical):

y = -6.1973 + (-0.0391 * Children) + (0.0070 * Age) + (0.7970 * DummyTechie) + (-2.2895 * DummyContract) + (0.1598 * DummyPort_modem) + (-1.4240 * DummyInternetService) + (-0.3193 * DummyPhone) + (-0.2964 * DummyMultiple) + (-0.3303 * DummyOnlineSecurity) + (-0.5146 * DummyOnlineBackup) + (-0.41 * DummyDeviceProtection) + (-0.3461 * DummyTechSupport) + (-0.2049 * Tenure) + (0.0463 * MonthlyCharge) + (0.0013 * Bandwidth_GB_Year)

E1. Model Comparison The second model still explains 52% of variance, as demonstrated by the pseudo R, even though we have reduced the number of variables in half (from 31 to 15). We have suggested an alpha threshold of 0.05 to retain predictor variables. We can see that, as Churn = 1 & that our majority of our dummy variables (which are additional services that a customer may add on to their contract) have negative values.
What is important to decision-makers & marketers is that those inverse relationships suggest that as a customer subscribes to more services that the company provided, an additional port modem or online backup for example, they are less likely to churn & leave the company. Cleary, it is in the best interest of retaining customers to provide them with more services & improve their experience with the company by helping customers understand all the services that are available to them as a subscriber, not simply mobile phone service.

Confusion Matrix

[108]:
```
# Import the prepared dataset
dataset = pd.read_csv('churn_prepared_log.csv')
X = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
```

[109]:
```
# Split the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 0)
```

[110]:
```
# Training the Logistic Regression model on the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

[110]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

[111]:
```
# Predict the Test set results
y_pred = classifier.predict(X_test)
```

[112]:
```
# Make the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[1356  130]
 [ 181  333]]
```

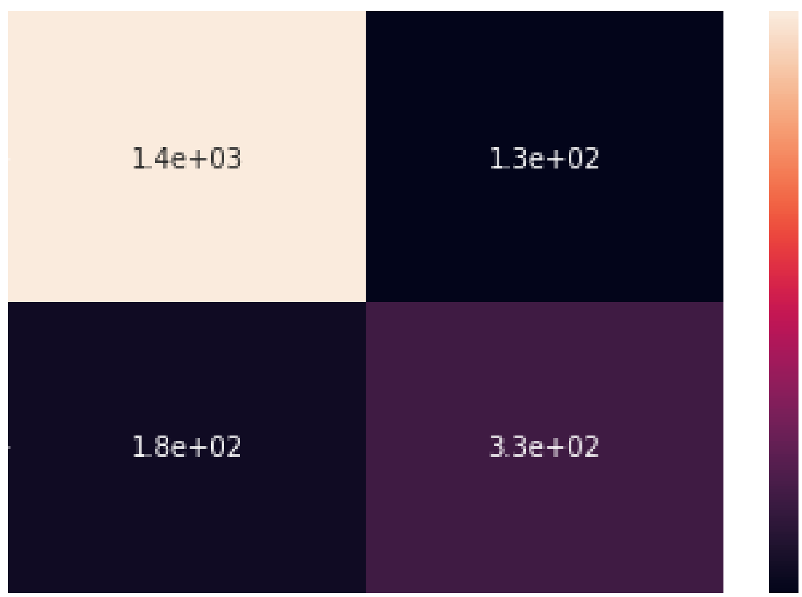
[113]:
```
## Compute the accuracy with k-Fold Cross Validation
from sklearn.model_selection import cross_val_score
```

```
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train,␣
 ↪cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 83.00 %
Standard Deviation: 1.30 %
```

[114]:
```
y_predict_test = classifier.predict(X_test)
cm2 = confusion_matrix(y_test, y_predict_test)
sns.heatmap(cm2, annot=True)
```

[114]: `<matplotlib.axes._subplots.AxesSubplot at 0x7fa17cbb9950>`



Classification Report

[115]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict_test))
```

```
              precision    recall  f1-score   support

           0       0.88      0.91      0.90      1486
           1       0.72      0.65      0.68       514

    accuracy                           0.84      2000
```

```
   macro avg       0.80      0.78      0.79      2000
weighted avg       0.84      0.84      0.84      2000
```

E2. Output & Calculations Calculations & code output above.

E3. Code All code for analysis include above.

F1. Results The final multiple regression equation with 15 predictor variables: y = -6.1973 - 0.0391 * Children + 0.0070 * Age + 0.7970 * DummyTechie - 2.2895 * DummyContract + 0.1598 * DummyPort_modem - 1.4240 * DummyInternetService - 0.3193 * DummyPhone - 0.2964 * DummyMultiple - 0.3303 * DummyOnlineSecurity - 0.5146 * DummyOnlineBackup - 0.4075 * DummyDeviceProtection - 0.3555 * DummyTechSupport - 0.2049 * Tenure + 0.0463 * MonthlyCharge + 0.0013 * Bandwidth_GB_Year

F2. Recommendations It is critical that decision-makers & marketers understand that there is inverse relationship between our target variable of Churn & several of our predictor variables. This suggests that as a customer subscribes to more services that the company provided, an additional port modem or online backup for example, they are less likely to leave the company. Clearly, it is the best interest of retaining customers to provide them with more services & improve their experience with the company by helping customers understand all the services that are available to them as a subscriber, not simple mobile phone service. Given the negative coefficients of additional services, we suggest additional marketing efforts for contracts & internet services as those with contract appear less likely to leave the company.
Also, with such a direct linear relationship between bandwidth used yearly & tenure with the telecom company it makes sense to suggest the company do everything within marketing & customer service capability to retain the customers gained as the longer they stay with the company the more bandwidth they tend to use. This would include making sure that fixes to customer problems are prompt & that the equipment provided is high quality to avoid fewer replacements of equipment.

G. Video link

H. Sources for Third-Party Code GeeksForGeeks. (2019, July 4). Python | Visualize missing values (NaN) values using Missingno Library. GeeksForGeeks. https://www.geeksforgeeks.org/python-visualize-missing-values-nan-values-using-missingno-library/

I. Sources CBTNuggets. (2018, September 20). Why Data Scientists Love Python. https://www.cbtnuggets.com/blog/technology/data/why-data-scientists-love-python
Massaron, L. & Boschetti, A. (2016). Regression Analysis with Python. Packt Publishing.

[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('D208_Performance_Assessment_NBM2_Task_2_revision2.ipynb')
```