D208_Performance_Assessment_NBM2_Task_2

July 19, 2021

1 D208 Performance Assessment NBM2 Task 2

1.1 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

1.1.1 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?

1.1.2 A2. Objectives & Goals:

Stakeholders in the company will benefit by knowing, with some measure of confidence, which customers are likely to churn soon because this will provide weight for decisions in marketing improved services to customers with these characteristics and past user experiences.

1.1.3 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'.

1.1.4 B2. Tool Benefits:

Python & IPython Jupyter notebooks will be used to support this analysis. Python offers very 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).

1.1.5 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 increaed 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.

1.1.6 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

1.1.7 C2. Summary Statistics:

As output by Python pandas dataframe methods below, there 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 a bimodal distributions, which demonstrated a direct linear relationship 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.

1.1.8 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 "Churn" at end of dataframe
- Finally, the prepared dataset will be extracted & provided as "churn_prepared_log.csv"

[61]:
```python
# 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>
```

[62]:
```
# 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')
```

[63]:
```
# 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
```

[64]:
```
# 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)
```

[65]:
```
# Display Churn dataframe
churn_df
```

[65]:
```
      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]
```

[66]:
```
# 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')
```

[67]:
```
# Find number of records and columns of dataset
churn_df.shape
```

[67]: (10000, 50)

[68]:
```
# Describe Churn dataset statistics
churn_df.describe()
```

[68]:
```
          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]
```

[69]:
```python
# 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()
```

[69]:
```
         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]
```

[70]:
```python
# 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
```

1.1.9 Dummy variable data preparation

[71]:
```python
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']]
```

[72]:
```
churn_df.head()
```

[72]:
```
   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]
```

[73]:
```
# 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()
```

[73]:
```
         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]
```

[74]:
```
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')
```

[75]:
```
# 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']]
```

[76]:
```
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')
```

1.1.10 C4. Visualizations:

Generate univariate and bivariate visualizations of the distributions of variables in the cleaned data set. Include the target variable in your bivariate visualizations.

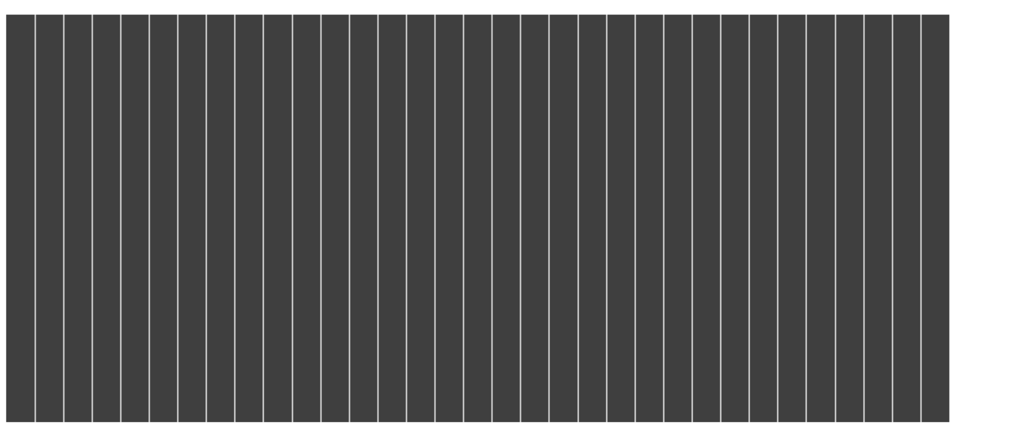
[77]:
```
# Visualize missing values in dataset

# Install appropriate library
!pip install missingno

# Importing the libraries
import missingno as msno

# 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: numpy in /usr/local/lib/python3.7/dist-packages
(from missingno) (1.19.5)
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: 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: python-dateutil>=2.1 in /usr/local/lib/python3.7
/dist-packages (from matplotlib->missingno) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7
/dist-packages (from matplotlib->missingno) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-
packages (from matplotlib->missingno) (0.10.0)
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)
```



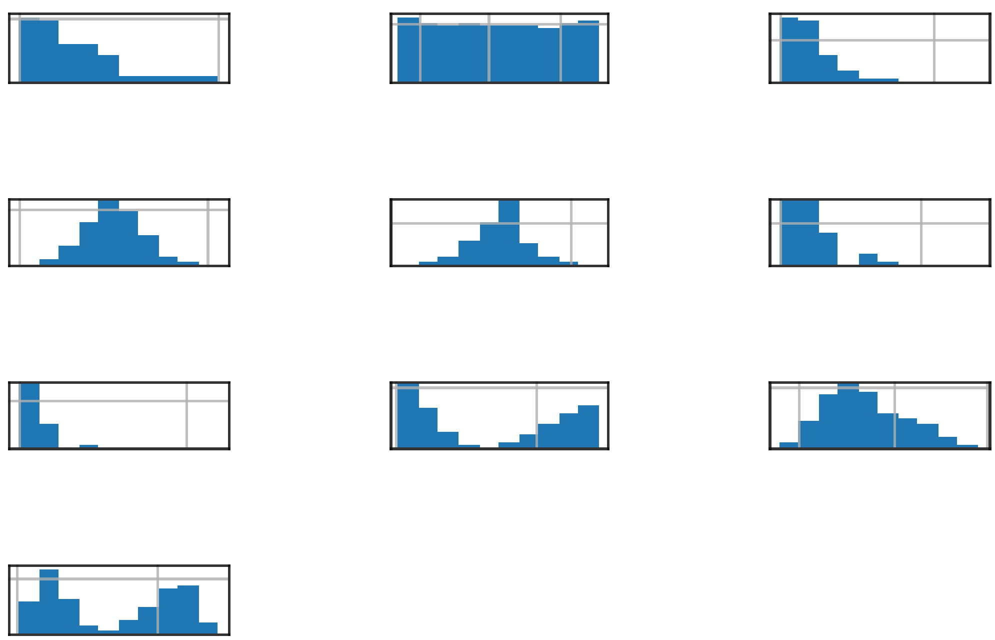
[78]:
```
'''No need to impute an missing values as the dataset appears complete/cleaned'''
# Impute missing fields for variables Children, Age, Income, Tenure and Bandwidth_GB_Year with median or mean
```

```
# churn_df['Children'] = churn_df['Children'].fillna(churn_df['Children'].
↪median())
# churn_df['Age'] = churn_df['Age'].fillna(churn_df['Age'].median())
# churn_df['Income'] = churn_df['Income'].fillna(churn_df['Income'].median())
# churn_df['Tenure'] = churn_df['Tenure'].fillna(churn_df['Tenure'].median())
# churn_df['Bandwidth_GB_Year'] = churn_df['Bandwidth_GB_Year'].
↪fillna(churn_df['Bandwidth_GB_Year'].median())
```

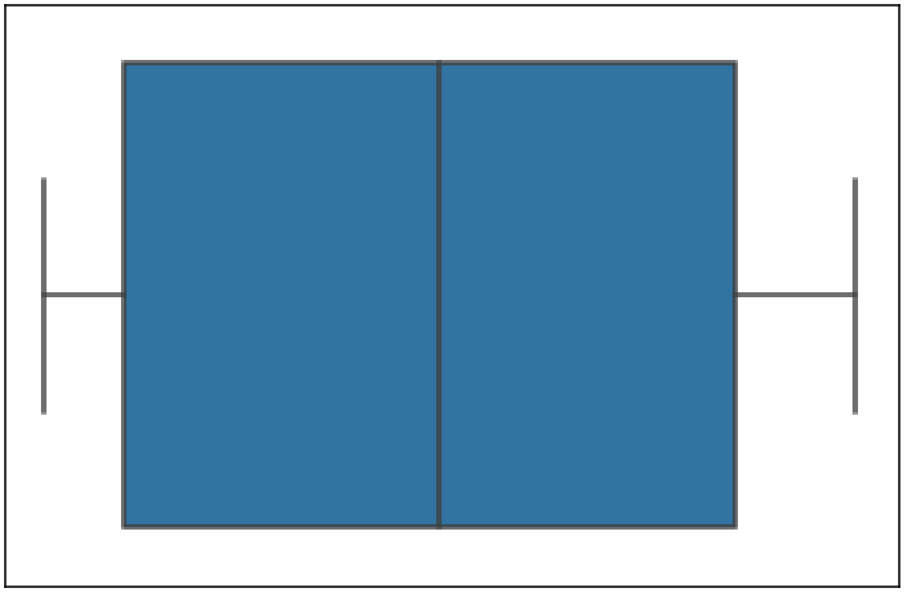
[78]: 'No need to impute an missing values as the dataset appears complete/cleaned'

1.2 Univariate Statistics

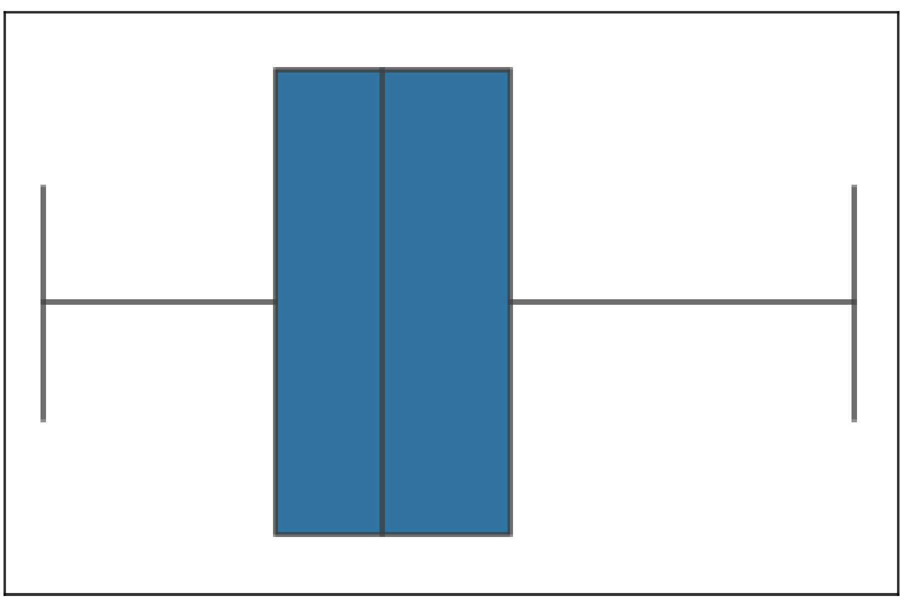
[79]:
```
# Create histograms of contiuous variables
churn_df[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email',
          'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
          'Bandwidth_GB_Year']].hist()
plt.savefig('churn_pyplot.jpg')
plt.tight_layout()
```



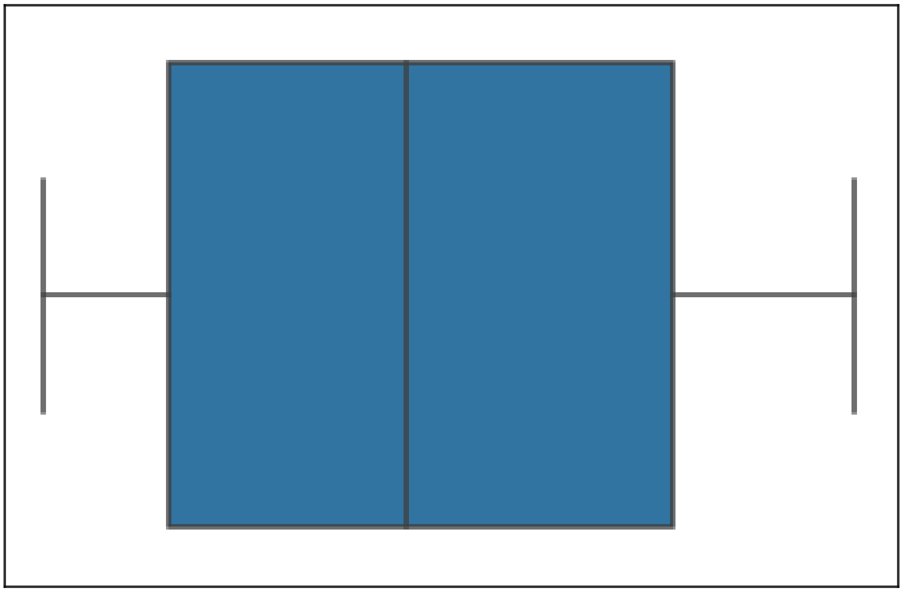
[80]:
```
# Create Seaborn boxplots for continuous variables
sns.boxplot('Tenure', data = churn_df)
plt.show()
```



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



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



1.2.1 It appears that anomolies have been removed from the dataset present "churn_clean.csv" as there are no remaining outliers.

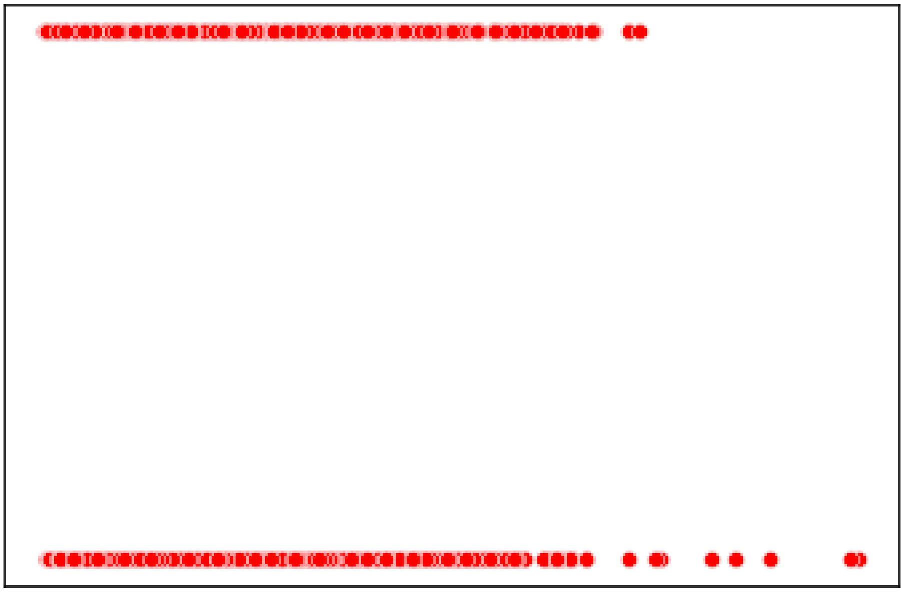
1.3 Bivariate Statistics

1.3.1 Let's run some scatterplots to get an idea of our linear relationships with our target variable of "Bandwidth_GB_Year" usage & some of the respective predictor variables.

[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();
```

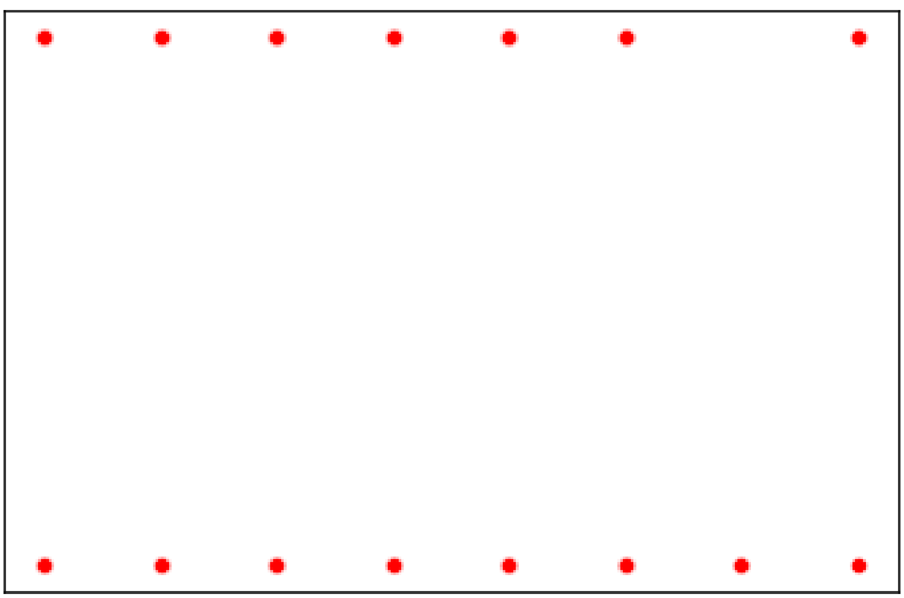


[86]:
```
sns.scatterplot(x=churn_df['DummyGender'], y=churn_df['DummyChurn'],␣
 ↪color='red')
plt.show();
```

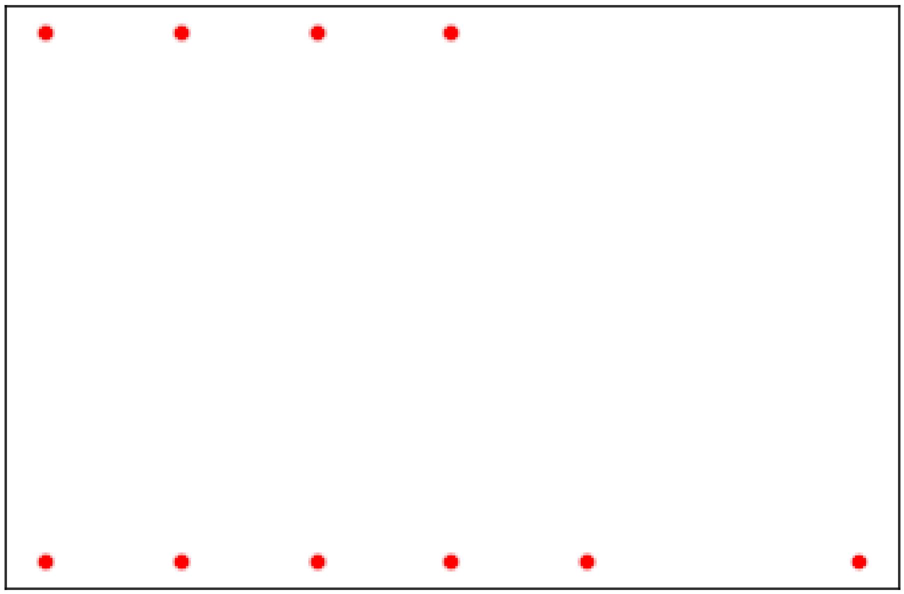
[87]:
```
sns.scatterplot(x=churn_df['Outage_sec_perweek'], y=churn_df['DummyChurn'],␣
 ↪color='red')
plt.show();
```

[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();
```

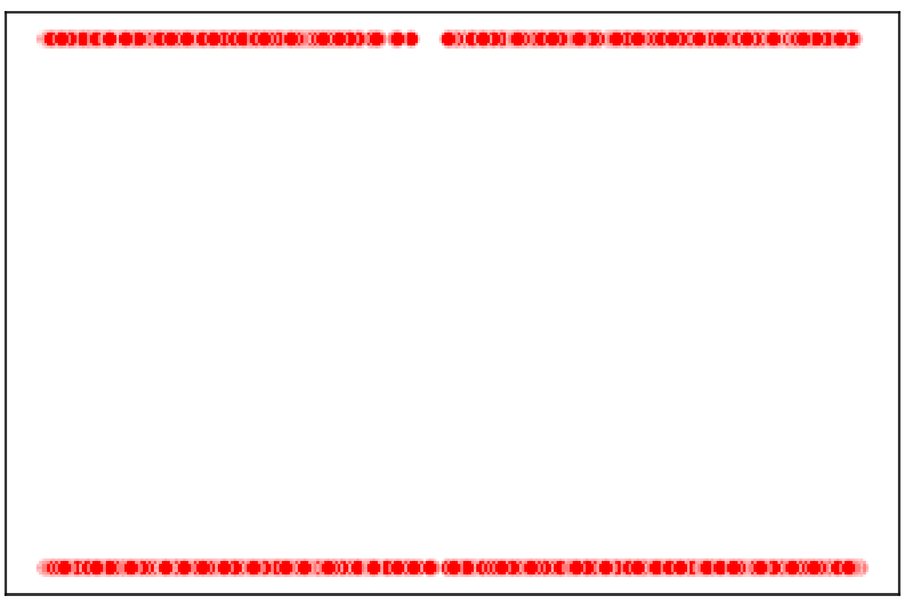


[90]:
```
sns.scatterplot(x=churn_df['Yearly_equip_failure'], y=churn_df['DummyChurn'],␣
 ↪color='red')
plt.show();
```



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

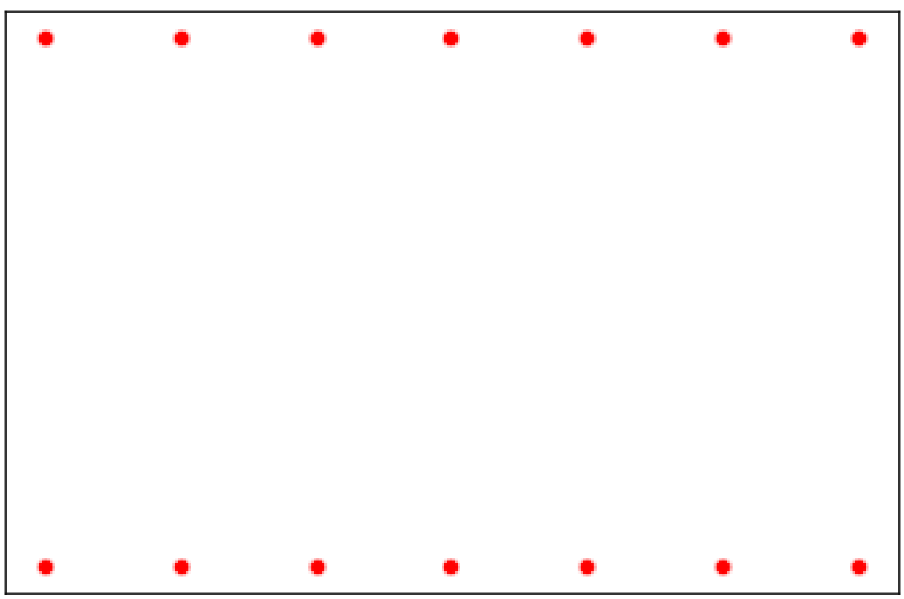
[92]:
```
sns.scatterplot(x=churn_df['Tenure'], y=churn_df['DummyChurn'], color='red')
plt.show();
```



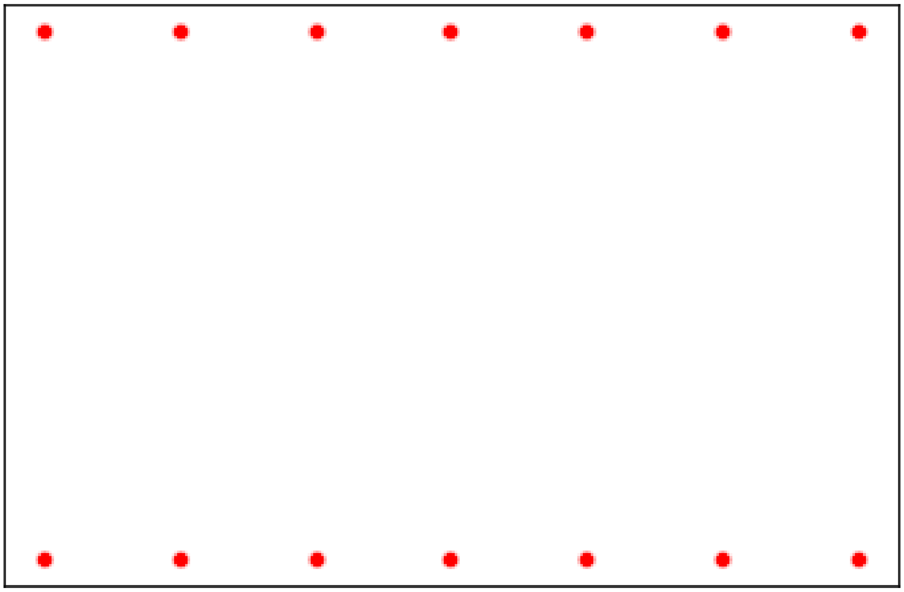
[93]:
```
sns.scatterplot(x=churn_df['MonthlyCharge'], y=churn_df['DummyChurn'],␣
 ↪color='red')
plt.show();
```

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

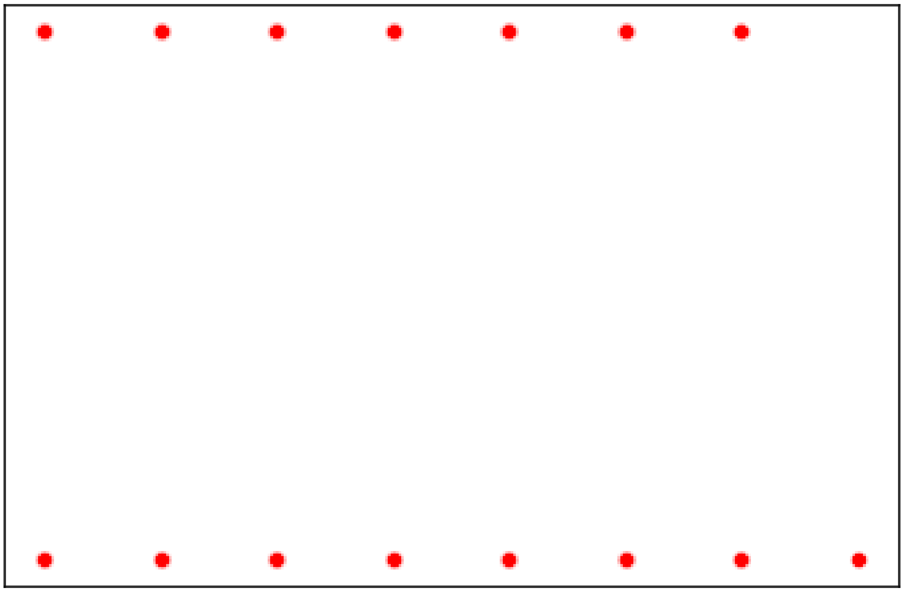
[95]:
```
sns.scatterplot(x=churn_df['TimelyResponse'], y=churn_df['DummyChurn'],␣
 ↪color='red')
plt.show();
```



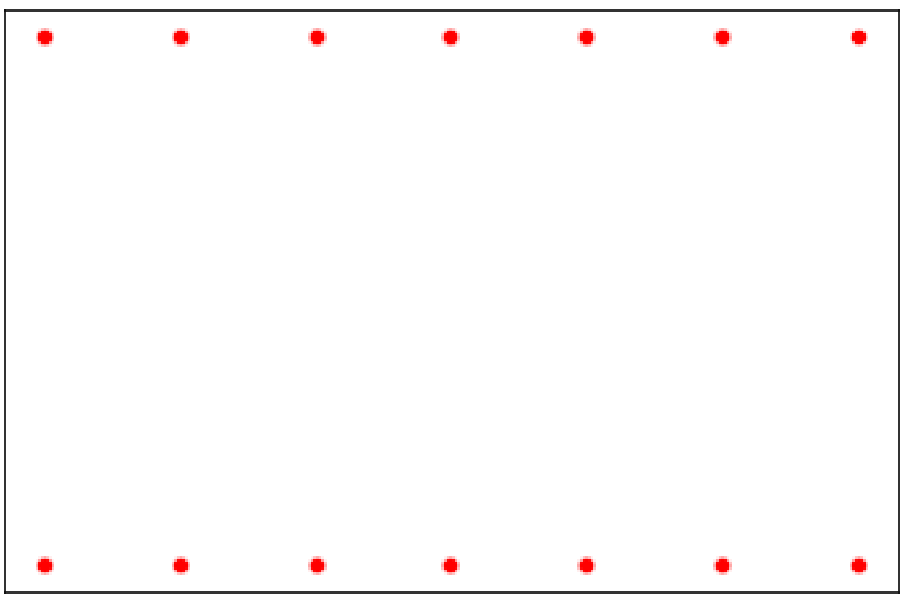
[96]:
```
sns.scatterplot(x=churn_df['Fixes'], y=churn_df['DummyChurn'], color='red')
plt.show();
```



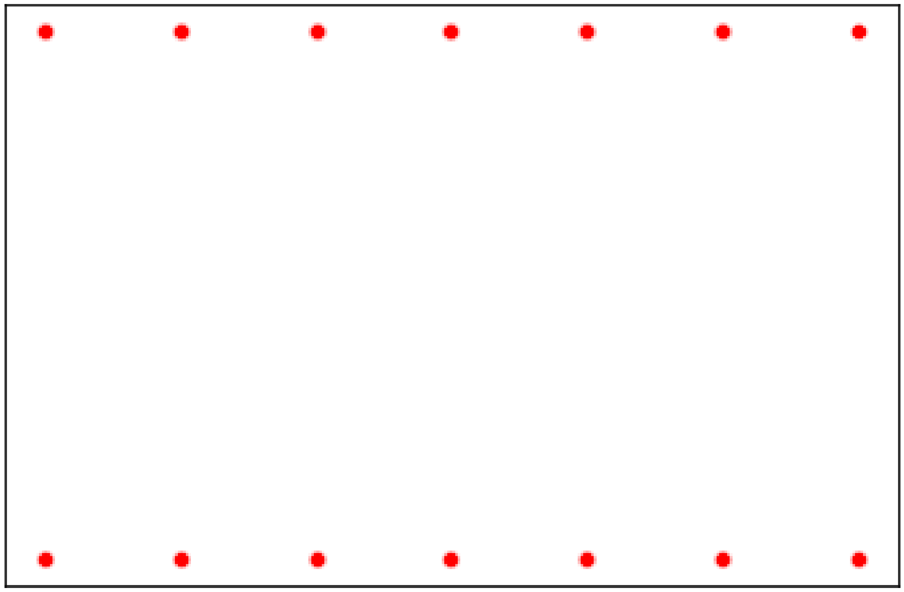
[97]:
```
sns.scatterplot(x=churn_df['Replacements'], y=churn_df['DummyChurn'],␣
 ↪color='red')
plt.show();
```



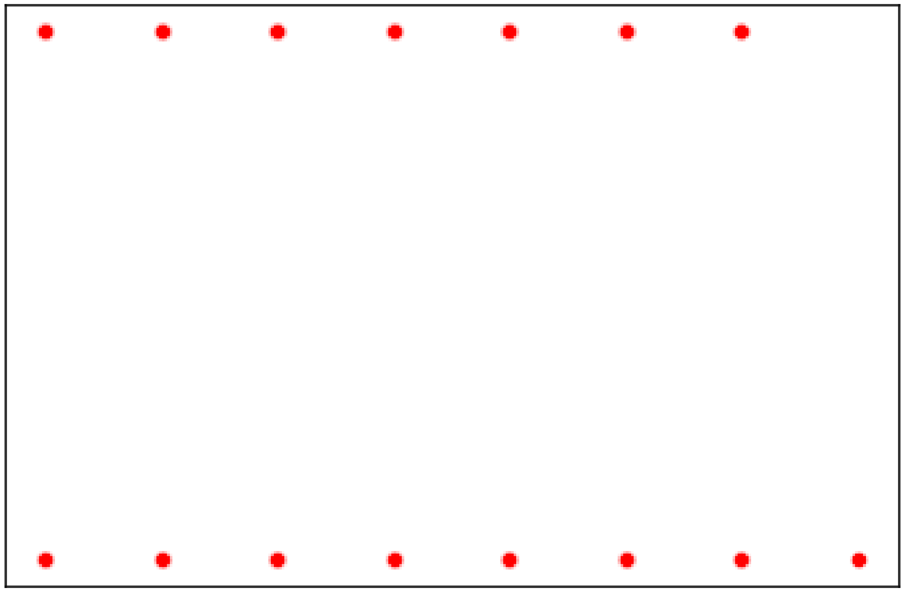
[98]:
```
sns.scatterplot(x=churn_df['Reliability'], y=churn_df['DummyChurn'],␣
↪color='red')
plt.show();
```



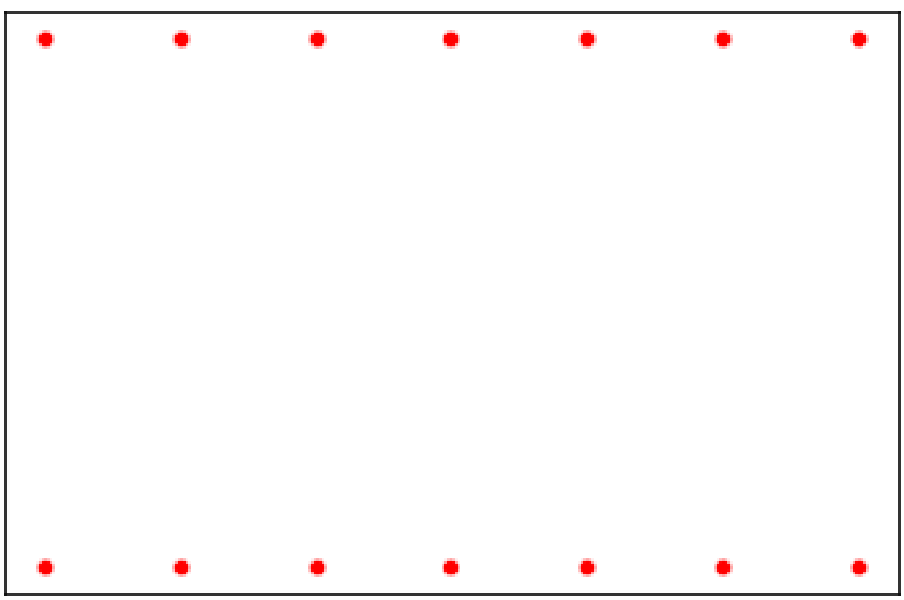
[99]:
```
sns.scatterplot(x=churn_df['Options'], y=churn_df['DummyChurn'], color='red')
plt.show();
```



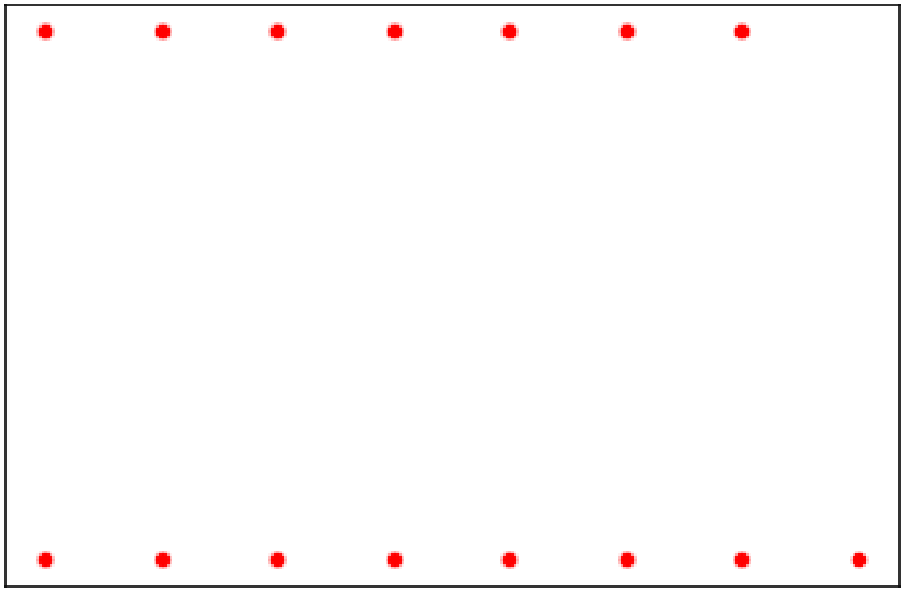
[100]:
```
sns.scatterplot(x=churn_df['Respectfulness'], y=churn_df['DummyChurn'],␣
↪color='red')
plt.show();
```



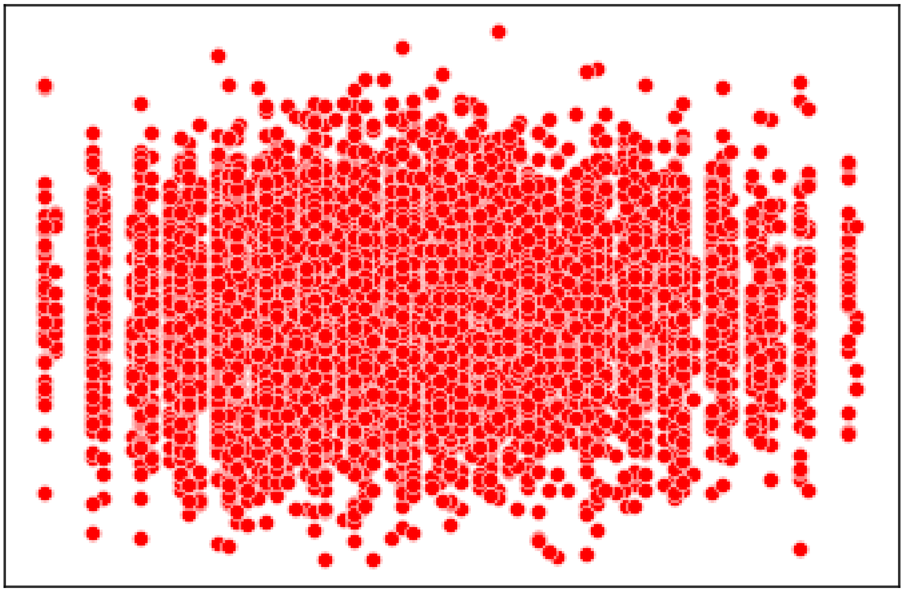
[101]:
```
sns.scatterplot(x=churn_df['Courteous'], y=churn_df['DummyChurn'], color='red')
plt.show();
```



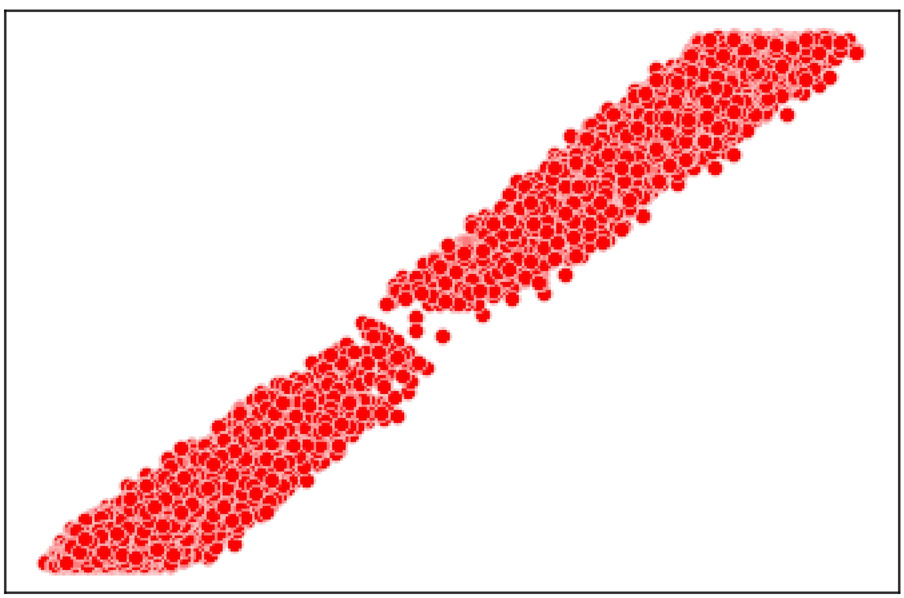
[102]:
```
sns.scatterplot(x=churn_df['Listening'], y=churn_df['DummyChurn'], color='red')
plt.show();
```



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



[104]:
```
sns.scatterplot(x=churn_df['Bandwidth_GB_Year'], y=churn_df['Tenure'],␣
 ↪color='red')
plt.show();
```



1.3.2 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. What gives witht this dataset, honestly?!

1.3.3 C5. Prepared Dataset:

Provide a copy of the prepared data set.

[105]:
```
# Extract Clean dataset
churn_df.to_csv('churn_prepared_log.csv')
```

1.3.4 D1. Initial Model

[106]:
```
"""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:               Mon, 19 Jul 2021   Pseudo R-squ.:                  0.4473
Time:                       17:00:28   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
================================================================================
========
```

1.3.5 Now, let's run a model including all encoded categorical dummy variables.

[107]:
```
"""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:                Mon, 19 Jul 2021   Pseudo R-squ.:                  0.5296
Time:                        17:00:31   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
================================================================================
=========
```

1.3.6 Initial Multiple Linear Regression Model

With 30 indpendent variables (17 continuous & 13 categorical): y = -5.86 - 0.0395 * Children + 0.0069 * Age + 1.199e-07 * Income - 0.0020 * Outage_sec_perweek - 0.0015 * Email + 0.0301 *

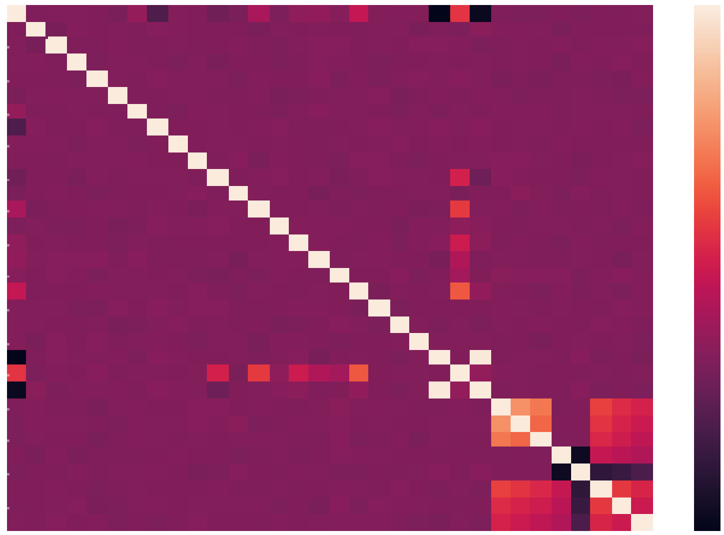
Contacts - 0.0308 * Yearly_equip_failure + 0.7956 * DummyTechie - 2.295 * DummyContract + 0.161 * DummyPort_modem - 0.0796 * DummyTablet - 1.4252 * DummyInternetService - 0.3157 * DummyPhone - 0.2908 * DummyMultiple - 0.3280 * DummyOnlineSecurity - 0.5125 * DummyOnlineBackup - 0.41 * DummyDeviceProtection - 0.3461 * DummyTechSupport + 0.0311 * DummyStreamingTV + 0.1126 * DummyPaperlessBilling - 0.2043 * Tenure + 0.0461 * MonthlyCharge - 0.0167 * TimelyResponse + 0.0143 * Fixes - 0.0158 * Replacements - 0.025 * Reliability - 0.0341 * Options - 0.0309 * Respectfulness + 0.0047 * Courteous - 0.009 * Listening

1.3.7 So, we have a pseudo R value = 0.5296, which is obviously not very good for the variance of our model. Coefficients on the "kitchen-sink" model above 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.

1.3.8 D2. Justification of Model Reduction

[108]:
```
# Create dataframe for heatmap bivariate analysis of correlation
churn_bivariate = churn_df[['DummyChurn', 'Children', 'Age', 'Income',
                             'Outage_sec_perweek', 'Yearly_equip_failure',␣
 ↪'DummyTechie', 'DummyContract',
                             'DummyPort_modem', 'DummyTablet',␣
 ↪'DummyInternetService',
                             'DummyPhone', 'DummyMultiple',␣
 ↪'DummyOnlineSecurity',
                             'DummyOnlineBackup', 'DummyDeviceProtection',
                             'DummyTechSupport', 'DummyStreamingTV',
                             'DummyPaperlessBilling','Email', 'Contacts',
                             'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year',␣
 ↪'TimelyResponse', 'Fixes',
                             'Replacements', 'Reliability', 'Options',␣
 ↪'Respectfulness',
                             'Courteous', 'Listening']]
```

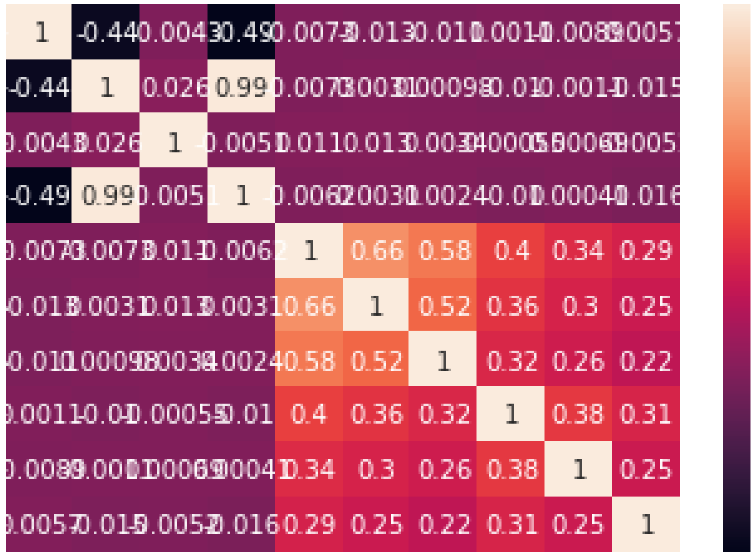
[109]:
```
# Run Seaborn heatmap
sns.heatmap(churn_bivariate.corr(), annot=False)
plt.show()
```



1.3.9 Alrighty, let's try that without some demographic, contacting-customer & options variables, basically purple or darker.

[110]:
```
churn_bivariate = churn_df[['DummyChurn', 'Bandwidth_GB_Year', 'Children',
                            'Tenure', 'TimelyResponse', 'Fixes',
                            'Replacements', 'Respectfulness',
                            'Courteous', 'Listening']]

sns.heatmap(churn_bivariate.corr(), annot=True)
plt.show()
```



1.3.10 DummyChurn (1 = the customer left the company) does not appear to be well correlated with any of our variables.

1.3.11 Again, based on the above MLE model we created, we have a pseudo R value = 0.5296, which is obviously not very good for the variance of our model. Coefficients on the "kitchen-sink" model above 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. Let's run just those variables against DummyChurn & see if we get any help.

1.3.12 D3. Reduced Multiple Regression Model

[111]:
```python
# Run reduced OLS multiple regression
churn_df['intercept'] = 1
churn_logit_model_reduced = sm.Logit(churn_df['DummyChurn'], churn_df[['DummyTechie', 'DummyContract', 'DummyInternetService', 'DummyOnlineBackup', 'intercept']]).fit()
print(churn_logit_model_reduced.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.555318
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:            DummyChurn   No. Observations:                10000
Model:                         Logit   Df Residuals:                     9995
Method:                          MLE   Df Model:                            4
Date:               Mon, 19 Jul 2021   Pseudo R-squ.:                 0.03961
Time:                       17:00:47   Log-Likelihood:                -5553.2
converged:                      True   LL-Null:                       -5782.2
Covariance Type:           nonrobust   LLR p-value:                 7.814e-98
================================================================================
========
                          coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
DummyTechie             0.3952      0.059      6.702      0.000       0.280
0.511
DummyContract          -1.1367      0.066    -17.215      0.000      -1.266
-1.007
DummyInternetService   -0.2771      0.047     -5.886      0.000      -0.369
-0.185
DummyOnlineBackup       0.2365      0.046      5.099      0.000       0.146
0.327
intercept              -0.8634      0.040    -21.602      0.000      -0.942
-0.785
================================================================================
========
```

1.3.13 Reduced Logistic Regression Model

With 4 indpendent variables: y = -0.8634 + 0.3952 * DummyTechie - 1.1367 * DummyContract - 0.2771 * DummyInternetServices + 0.2365 * DummyOnlineBackup

1.3.14 E1. Model Comparison

1.3.15 Confusion Matrix

[112]:
```python
# Import the prepared dataset
dataset = pd.read_csv('churn_prepared_log.csv')
X = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
```

[113]:
```python
# 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)
```

[114]:
```
# 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)
```

[114]:
```
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)
```

[115]:
```
# Predict the Test set results
y_pred = classifier.predict(X_test)
```

[116]:
```
# Make the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

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

[117]:
```
## 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: 82.93 %
Standard Deviation: 1.35 %
```

[118]:
```
y_predict_test = classifier.predict(X_test)
cm2 = confusion_matrix(y_test, y_predict_test)
sns.heatmap(cm2, annot=True)
```

[118]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f91adda6dd0>
```

1.3.16 Classification Report

[119]:
```
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
```

1.3.17 E2. Output & Calculations

Calculations & code output above.

1.3.18 E3. Code

All code for analysis include above.

1.3.19 F1. Results

Discuss the results of your data analysis, including the following elements:

```
<li>
The final multiple regression equation with <b>4</b> indpendent variables:
```

y = -0.8634 + 0.3952 * DummyTechie - 1.1367 * DummyContract - 0.2771 * DummyInternetServices + 0.2365 * DummyOnlineBackup

```
<li>
The coefficients suggest that for every 1 unit of:
    <ul>
        <li>Children - Bandwidth_GB_Year will increase 31.18 units</li>
        <li>Tenure - Bandwidth_GB_Year will increase 81.94 units</li>
        <li>Fixes - Bandwidth_GB_Year will increase 1.07 units</li>
        <li>Replacements -  - Bandwidth_GB_Year will decrease 3.66 units</li>
    </ul>
</li>
<li>
P-values for Children & Tenure are statistically significant at 0.000, while p-values for Fixes
</li>
<li>
The limitations of this analysis are that the data set is a bit small & that perhaps more years
</li>
```

1.3.20 F2. Recommendations

For the purposes of this analysis & to make the time spent on the analysis acceptable & provide actionable information:

Given the negtive coefficients of DummyContract & DummyInternetServices, we suggest additional marketing 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.

1.3.21 G. Video

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=fc1531be-edf7-442a-a724-ad6a0117d872

1.3.22 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/

1.3.23 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.ipynb')
```

```
File colab_pdf.py already there; not retrieving.

Mounted at /content/drive/

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```

[ ]:
```
```