D208 Performance Assessment Task 2

April 29, 2022

Kiet Nguyen

ID: 001601720

Email: kngu179@wgu.edu

0.0.1 A. Purpose of Analysis

A1. Research Question Which independent variables had the most significant impact on customer `Churn`?

A2. Objectives The objective of this analysis was to identify which independent variables were significant in predicting customer `Churn`. Stakeholders could use the results of this analysis to see what factors influenced why customers disconnected their service.

0.0.2 B. Logistic Regression Description

B1. Summarize Assumptions Logistic regression made five assumptions before the analysis (Statistic Solutions, 2021):

1. The dependent variable must be binary.
2. Observations are independent of each other.
3. Independent variables should not be highly correlated with each other.
4. Independent variables are related linearly to the log odds.
5. A large sample size.

B2. Tool Benefits I chose Python for this project because it is a flexible and powerful programming language. The syntax was also easy to read. On top of that, Python had a great ecosystem of libraries that made data analysis tasks much easier (Rane, 2021). The libraries used in this project were:

- `NumPy`: high-performance numerical computation.
- `Pandas`: fats and flexible dataframes.
- `Matplotlib` and `Seaborn`: data visualizations.
- `Statsmodels`: classes and functions for different statistical models.

The environment for this project was Conda, an open-source package manager and environment management system.

B3. Logistic Regression Justification Logistic regression is a statistical method for predicting the outcome of a categorical dependent variable (Joby, 2021). This predictive modeling technique is great since the dependent variable `Churn` is a categorical variable with `Yes` or `No` values. The logistic model could be used to explain strengths of the independent variables on `Churn`.

0.0.3 C. Data Preparation Process

C1. Preparation Goals The goals of data preparation included:

- Learn about the dataset and its variables.
- Explore measures of central tendency (mean, median, and mode).
- Check for missing data and handle them as necessary.
- Visualize data through univariate and bivariate plots.
- Remove highly correlated columns using heatmap and variance inflation factor (VIF).

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
from statsmodels.stats.outliers_influence import variance_inflation_factor

churn = pd.read_csv('churn_clean.csv')
churn_cp = churn.copy(deep=True)
```

[2]:
```
churn_cp.head()
```

[2]:
```
   CaseOrder Customer_id                           Interaction  \
0          1     K409198  aa90260b-4141-4a24-8e36-b04ce1f4f77b
1          2     S120509  fb76459f-c047-4a9d-8af9-e0f7d4ac2524
2          3     K191035  344d114c-3736-4be5-98f7-c72c281e2d35
3          4      D90850  abfa2b40-2d43-4994-b15a-989b8c79e311
4          5     K662701  68a861fd-0d20-4e51-a587-8a90407ee574

                                UID         City State                 County  \
0  e885b299883d4f9fb18e39c75155d990  Point Baker    AK  Prince of Wales-Hyder
1  f2de8bef964785f41a2959829830fb8a  West Branch    MI                Ogemaw
2  f1784cfa9f6d92ae816197eb175d3c71      Yamhill    OR               Yamhill
3  dc8a365077241bb5cd5ccd305136b05e      Del Mar    CA             San Diego
4  aabb64a116e83fdc4befc1fbab1663f9    Needville    TX             Fort Bend

     Zip       Lat        Lng  …  MonthlyCharge Bandwidth_GB_Year Item1  \
0  99927  56.25100 -133.37571  …     172.455519        904.536110     5
1  48661  44.32893  -84.24080  …     242.632554        800.982766     3
2  97148  45.35589 -123.24657  …     159.947583       2054.706961     4
3  92014  32.96687 -117.24798  …     119.956840       2164.579412     4
4  77461  29.38012  -95.80673  …     149.948316        271.493436     4
```

```
  Item2  Item3  Item4  Item5 Item6 Item7 Item8
0     5      5      3      4     4     3     4
1     4      3      3      4     3     4     4
2     4      2      4      4     3     3     3
3     4      4      2      5     4     3     3
4     4      4      3      4     4     4     5

[5 rows x 50 columns]
```

[3]:
```
# Check statistic of Churn column
churn_cp['Churn'].describe()
```

[3]:
```
count     10000
unique        2
top          No
freq       7350
Name: Churn, dtype: object
```

[4]:
```
# Rename survey responses columns
churn_cp.rename(columns={
    'Item1': 'SurveyResponse',
    'Item2': 'SurveyFixes',
    'Item3': 'SurveyReplacements',
    'Item4': 'SurveyReliability',
    'Item5': 'SurveyOptions',
    'Item6': 'SurveyRespect',
    'Item7': 'SurveyCourteous',
    'Item8': 'SurveyListening'
}, inplace=True)
```

[5]:
```
# Drop customer demographic and survey columns that are not important to the␣
↪regression analysis
churn_cp.drop(['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City',␣
↪'State', 'County', 'Zip', 'Lat', 'Lng', 'TimeZone', 'Job',␣
↪'PaperlessBilling', 'PaymentMethod'], axis=1, inplace=True)

churn_cp.head()
```

[5]:
```
   Population      Area  Children  Age    Income    Marital  Gender Churn  \
0          38     Urban         0   68  28561.99    Widowed    Male    No
1       10446     Urban         1   27  21704.77    Married  Female   Yes
2        3735     Urban         4   50   9609.57    Widowed  Female    No
3       13863  Suburban         1   48  18925.23    Married    Male    No
4       11352  Suburban         0   83  40074.19  Separated    Male   Yes

   Outage_sec_perweek  Email  …  MonthlyCharge  Bandwidth_GB_Year  \
0            7.978323     10  …     172.455519         904.536110
```

```
1          11.699080     12  …      242.632554        800.982766
2          10.752800      9  …      159.947583       2054.706961
3          14.913540     15  …      119.956840       2164.579412
4           8.147417     16  …      149.948316        271.493436

  SurveyResponse SurveyFixes SurveyReplacements SurveyReliability  \
0              5           5                  5                 3
1              3           4                  3                 3
2              4           4                  2                 4
3              4           4                  4                 2
4              4           4                  4                 3

  SurveyOptions SurveyRespect SurveyCourteous SurveyListening
0             4             4               3               4
1             4             3               4               4
2             4             3               3               3
3             5             4               3               3
4             4             4               4               5

[5 rows x 36 columns]
```

C2. Summary Statistics The original dataset contained 10,000 rows and 50 columns. Fourteen columns were dropped from the dataset because they were not relevant to the logistic regression analysis. These columns were customer demographic data: `CaseOrder`, `Customer_id`, `Interaction`, `UID`, `City`, `State`, `County`, `Zip`, `Lat`, `Lng`, `TimeZone`, `Job`, `PaperlessBilling`, and `PaymentMethod`. Each of these demographic columns contained multiple unique categorical values that would generate too many dummy variables.

The last eight columns were survey responses, `Item1` through `Item8`. They were renamed appropriately to their respective category names.

The preliminary dataset contained 35 independent variables and one dependent variable. The target dependent variable is `Churn`, a binary column of whether customer discontinued service within the last month. Its values were 2650 `Yes` and 7350 `No`.

The nineteen continuous independent variables were: `Population`, `Children`, `Age`, `Income`, `Outage_sec_perweek`, `Email`, `Contacts`, `Yearly_equip_failure`, `Tenure`, `MonthlyCharge`, `Bandwidth_GB_Year`, `SurveyResponse`, `SurveyFixes`, `SurveyReplacements`, `SurveyReliability`, `SurveyOptions`, `SurveyRespect`, `SurveyCourteous`, and `SurveyListening`.

The seventeen categorical independent variables were: `Area`, `Marital`, `Gender`, `Churn`, `Techie`, `Contract`, `Port_modem`, `Tablet`, `InternetService`, `Phone`, `Multiple`, `OnlineSecurity`, `OnlineBackup`, `DeviceProtection`, `TechSupport`, `StreamingTV`, and `StreamingMovies`.

[6]:
```
churn_cp.describe()
```

[6]:
```
         Population    Children           Age        Income  \
count  10000.000000  10000.0000  10000.000000  10000.000000
mean    9756.562400      2.0877     53.078400  39806.926771
```

```
std     14432.698671      2.1472    20.698882   28199.916702
min         0.000000      0.0000    18.000000     348.670000
25%       738.000000      0.0000    35.000000   19224.717500
50%      2910.500000      1.0000    53.000000   33170.605000
75%     13168.000000      3.0000    71.000000   53246.170000
max    111850.000000     10.0000    89.000000  258900.700000

       Outage_sec_perweek         Email      Contacts  Yearly_equip_failure  \
count        10000.000000  10000.000000  10000.000000          10000.000000
mean            10.001848     12.016000      0.994200              0.398000
std              2.976019      3.025898      0.988466              0.635953
min              0.099747      1.000000      0.000000              0.000000
25%              8.018214     10.000000      0.000000              0.000000
50%             10.018560     12.000000      1.000000              0.000000
75%             11.969485     14.000000      2.000000              1.000000
max             21.207230     23.000000      7.000000              6.000000

             Tenure  MonthlyCharge  Bandwidth_GB_Year  SurveyResponse  \
count  10000.000000   10000.000000       10000.000000    10000.000000
mean      34.526188     172.624816        3392.341550        3.490800
std       26.443063      42.943094        2185.294852        1.037797
min        1.000259      79.978860         155.506715        1.000000
25%        7.917694     139.979239        1236.470827        3.000000
50%       35.430507     167.484700        3279.536903        3.000000
75%       61.479795     200.734725        5586.141370        4.000000
max       71.999280     290.160419        7158.981530        7.000000

        SurveyFixes  SurveyReplacements  SurveyReliability  SurveyOptions  \
count  10000.000000        10000.000000       10000.000000   10000.000000
mean       3.505100            3.487000           3.497500       3.492900
std        1.034641            1.027977           1.025816       1.024819
min        1.000000            1.000000           1.000000       1.000000
25%        3.000000            3.000000           3.000000       3.000000
50%        4.000000            3.000000           3.000000       3.000000
75%        4.000000            4.000000           4.000000       4.000000
max        7.000000            8.000000           7.000000       7.000000

       SurveyRespect  SurveyCourteous  SurveyListening
count   10000.000000     10000.000000     10000.000000
mean        3.497300         3.509500         3.495600
std         1.033586         1.028502         1.028633
min         1.000000         1.000000         1.000000
25%         3.000000         3.000000         3.000000
50%         3.000000         4.000000         3.000000
75%         4.000000         4.000000         4.000000
max         8.000000         7.000000         8.000000
```

```
[7]: churn_cp.describe(include=object)
```

```
[7]:            Area   Marital  Gender  Churn Techie        Contract Port_modem  \
count        10000     10000   10000  10000  10000           10000      10000
unique           3         5       3      2      2               3          2
top       Suburban  Divorced  Female     No     No  Month-to-month         No
freq          3346      2092    5025   7350   8321            5456       5166

       Tablet InternetService  Phone Multiple OnlineSecurity OnlineBackup  \
count   10000           10000  10000    10000          10000        10000
unique      2               3      2        2              2            2
top        No     Fiber Optic    Yes       No             No           No
freq     7009            4408   9067     5392           6424         5494

       DeviceProtection TechSupport StreamingTV StreamingMovies
count             10000       10000       10000           10000
unique                2           2           2               2
top                  No          No          No              No
freq               5614        6250        5071            5110
```

C3. Preparation Steps The steps used to prepare the data are:

1. Get an overview of the imported data.
2. Rename survey columns with ambiguous names.
3. Drop demographic columns unnecessary for logistic regression.
4. Check statistics of continuous and categorical variables.
5. Check for any duplicate or missing values.
6. Plot univariate and bivariate plots.
7. Encode categorical variables into numerical values.
8. Run a heatmap and drop highly correlated columns.
9. Calculate VIF and drop columns with VIF $>$ 5.

```
[8]: # Check if there is any duplicates
churn_cp.duplicated().any()
```

```
[8]: False
```

```
[9]: # Check if there is any missing values
churn_cp.isnull().values.any()
```

```
[9]: False
```

```
[10]: # Convert categorical columns into dummy variables
churn_dmy = pd.get_dummies(churn_cp, drop_first=True)

# Replace space in column names with underscore
churn_dmy.columns = churn_dmy.columns.str.replace(' ', '_')
```

```python
churn_dmy.describe()
```

[10]:

```
         Population    Children           Age         Income  \
count  10000.000000  10000.0000  10000.000000   10000.000000
mean    9756.562400      2.0877     53.078400   39806.926771
std    14432.698671      2.1472     20.698882   28199.916702
min        0.000000      0.0000     18.000000     348.670000
25%      738.000000      0.0000     35.000000   19224.717500
50%     2910.500000      1.0000     53.000000   33170.605000
75%    13168.000000      3.0000     71.000000   53246.170000
max   111850.000000     10.0000     89.000000  258900.700000

       Outage_sec_perweek         Email      Contacts  Yearly_equip_failure  \
count        10000.000000  10000.000000  10000.000000          10000.000000
mean            10.001848     12.016000      0.994200              0.398000
std              2.976019      3.025898      0.988466              0.635953
min              0.099747      1.000000      0.000000              0.000000
25%              8.018214     10.000000      0.000000              0.000000
50%             10.018560     12.000000      1.000000              0.000000
75%             11.969485     14.000000      2.000000              1.000000
max             21.207230     23.000000      7.000000              6.000000

             Tenure  MonthlyCharge  …  InternetService_Fiber_Optic  \
count  10000.000000   10000.000000  …                 10000.000000
mean      34.526188     172.624816  …                     0.440800
std       26.443063      42.943094  …                     0.496508
min        1.000259      79.978860  …                     0.000000
25%        7.917694     139.979239  …                     0.000000
50%       35.430507     167.484700  …                     0.000000
75%       61.479795     200.734725  …                     1.000000
max       71.999280     290.160419  …                     1.000000

       InternetService_None     Phone_Yes  Multiple_Yes  OnlineSecurity_Yes  \
count          10000.000000  10000.000000  10000.000000        10000.000000
mean               0.212900      0.906700      0.460800            0.357600
std                0.409378      0.290867      0.498486            0.479317
min                0.000000      0.000000      0.000000            0.000000
25%                0.000000      1.000000      0.000000            0.000000
50%                0.000000      1.000000      0.000000            0.000000
75%                0.000000      1.000000      1.000000            1.000000
max                1.000000      1.000000      1.000000            1.000000

       OnlineBackup_Yes  DeviceProtection_Yes  TechSupport_Yes  \
count      10000.000000          10000.000000     10000.000000
mean           0.450600              0.438600         0.375000
std            0.497579              0.496241         0.484147
```

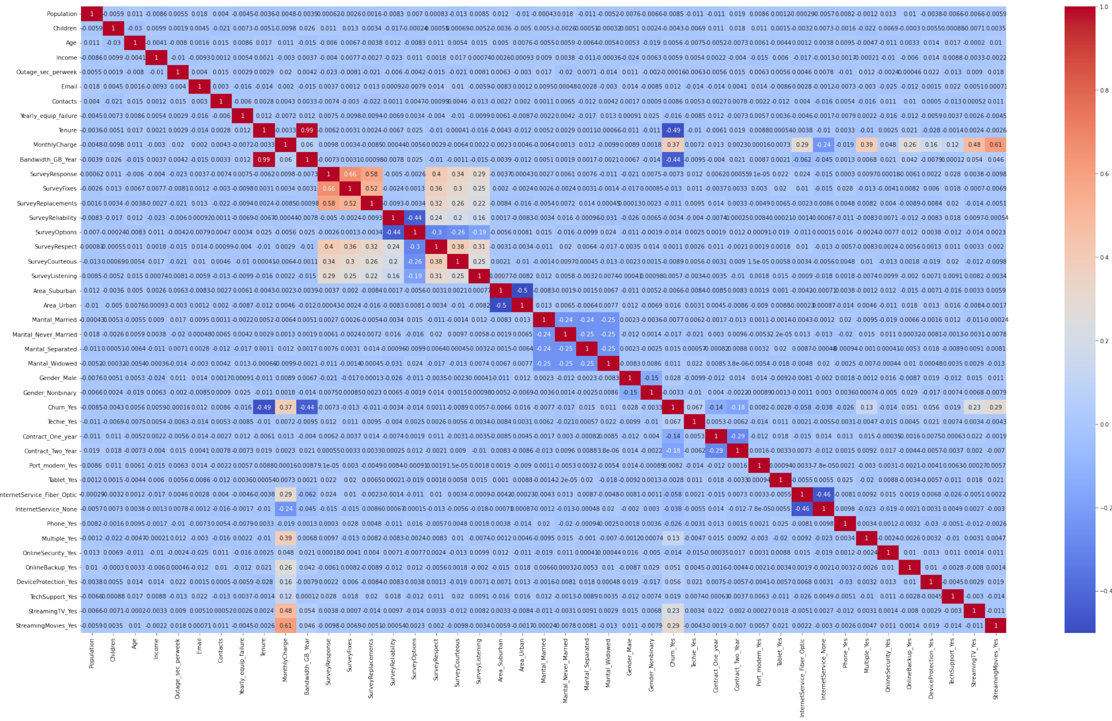
```
min          0.000000           0.000000         0.000000
25%          0.000000           0.000000         0.000000
50%          0.000000           0.000000         0.000000
75%          1.000000           1.000000         1.000000
max          1.000000           1.000000         1.000000

       StreamingTV_Yes  StreamingMovies_Yes
count     10000.000000         10000.000000
mean          0.492900             0.489000
std           0.499975             0.499904
min           0.000000             0.000000
25%           0.000000             0.000000
50%           0.000000             0.000000
75%           1.000000             1.000000
max           1.000000             1.000000

[8 rows x 43 columns]
```

[11]:
```python
# Run a correlation matrix
churn_corr = churn_dmy.corr()

# Plot a heatmap of the correlations
plt.figure(figsize=(36, 20))
sns.heatmap(churn_corr, annot=True, xticklabels=churn_corr.columns,
 ↪yticklabels=churn_corr.columns, cmap='coolwarm')
plt.show()
```



[12]:
```python
# Drop highly correlated columns
churn_dmy.drop(['Bandwidth_GB_Year'], axis=1, inplace=True)

churn_dmy.head()
```

[12]:
```
   Population  Children  Age    Income  Outage_sec_perweek  Email  Contacts  \
0          38         0   68  28561.99            7.978323     10         0
1       10446         1   27  21704.77           11.699080     12         0
2        3735         4   50   9609.57           10.752800      9         0
3       13863         1   48  18925.23           14.913540     15         2
4       11352         0   83  40074.19            8.147417     16         2

   Yearly_equip_failure     Tenure  MonthlyCharge  …  \
0                     1   6.795513     172.455519  …
1                     1   1.156681     242.632554  …
2                     1  15.754144     159.947583  …
3                     0  17.087227     119.956840  …
4                     1   1.670972     149.948316  …

   InternetService_Fiber_Optic  InternetService_None  Phone_Yes  Multiple_Yes  \
0                            1                     0          1             0
1                            1                     0          1             1
2                            0                     0          1             1
3                            0                     0          1             0
4                            1                     0          0             0

   OnlineSecurity_Yes  OnlineBackup_Yes  DeviceProtection_Yes  \
0                   1                 1                     0
1                   1                 0                     0
2                   0                 0                     0
3                   1                 0                     0
4                   0                 0                     0

   TechSupport_Yes  StreamingTV_Yes  StreamingMovies_Yes
0                0                0                    1
1                0                1                    1
2                0                0                    1
3                0                1                    0
4                1                1                    0

[5 rows x 42 columns]
```

[13]:
```python
# Adapted from Detecting Multicollinearity with VIF (GeeksforGeeks, 2020)
# https://www.geeksforgeeks.org/detecting-multicollinearity-with-vif-python/
```

```python
# Create independent variables set
X = churn_dmy.drop('Churn_Yes', axis=1)

# VIF dataframe
vif = pd.DataFrame()
vif['Variables'] = X.columns

# Calculating VIF for each variable
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(len(X.
 ↪columns))]

vif
```

[13]:

	Variables	VIF
0	Population	1.459176
1	Children	1.942902
2	Age	7.425424
3	Income	2.977824
4	Outage_sec_perweek	11.700163
5	Email	15.692474
6	Contacts	2.010362
7	Yearly_equip_failure	1.392277
8	Tenure	2.698238
9	MonthlyCharge	249.504282
10	SurveyResponse	27.308743
11	SurveyFixes	24.149048
12	SurveyReplacements	19.988065
13	SurveyReliability	14.520559
14	SurveyOptions	14.349580
15	SurveyRespect	18.206588
16	SurveyCourteous	16.293513
17	SurveyListening	14.683202
18	Area_Suburban	1.994485
19	Area_Urban	1.991253
20	Marital_Married	1.905247
21	Marital_Never_Married	1.920322
22	Marital_Separated	1.948212
23	Marital_Widowed	1.957485
24	Gender_Male	1.941049
25	Gender_Nonbinary	1.048496
26	Techie_Yes	1.205360
27	Contract_One_year	1.388670
28	Contract_Two_Year	1.451514
29	Port_modem_Yes	1.931974
30	Tablet_Yes	1.430785
31	InternetService_Fiber_Optic	3.888479
32	InternetService_None	1.808071

```
33                Phone_Yes   10.280773
34             Multiple_Yes    6.007040
35       OnlineSecurity_Yes    1.595057
36         OnlineBackup_Yes    3.799647
37     DeviceProtection_Yes    2.411337
38          TechSupport_Yes    2.127839
39          StreamingTV_Yes    9.288338
40      StreamingMovies_Yes   12.942925
```

[14]:
```python
# Get a list of variables with VIF > 5
high_vif = vif[vif['VIF'] > 5]
high_vif = high_vif['Variables'].tolist()

# Drop columns with high VIF
churn_dmy.drop(high_vif, axis=1, inplace=True)

churn_dmy.head()
```

[14]:
```
   Population  Children    Income  Contacts  Yearly_equip_failure     Tenure  \
0          38         0  28561.99         0                     1   6.795513
1       10446         1  21704.77         0                     1   1.156681
2        3735         4   9609.57         0                     1  15.754144
3       13863         1  18925.23         2                     0  17.087227
4       11352         0  40074.19         2                     1   1.670972

   Area_Suburban  Area_Urban  Marital_Married  Marital_Never_Married  …  \
0              0           1                0                      0  …
1              0           1                1                      0  …
2              0           1                0                      0  …
3              1           0                1                      0  …
4              1           0                0                      0  …

   Contract_One_year  Contract_Two_Year  Port_modem_Yes  Tablet_Yes  \
0                  1                  0               1           1
1                  0                  0               0           1
2                  0                  1               1           0
3                  0                  1               0           0
4                  0                  0               1           0

   InternetService_Fiber_Optic  InternetService_None  OnlineSecurity_Yes  \
0                            1                     0                   1
1                            1                     0                   1
2                            0                     0                   0
3                            0                     0                   1
4                            1                     0                   0

   OnlineBackup_Yes  DeviceProtection_Yes  TechSupport_Yes
```

```
0                    1                    0                    0
1                    0                    0                    0
2                    0                    0                    0
3                    0                    0                    0
4                    0                    0                    1

[5 rows x 26 columns]
```

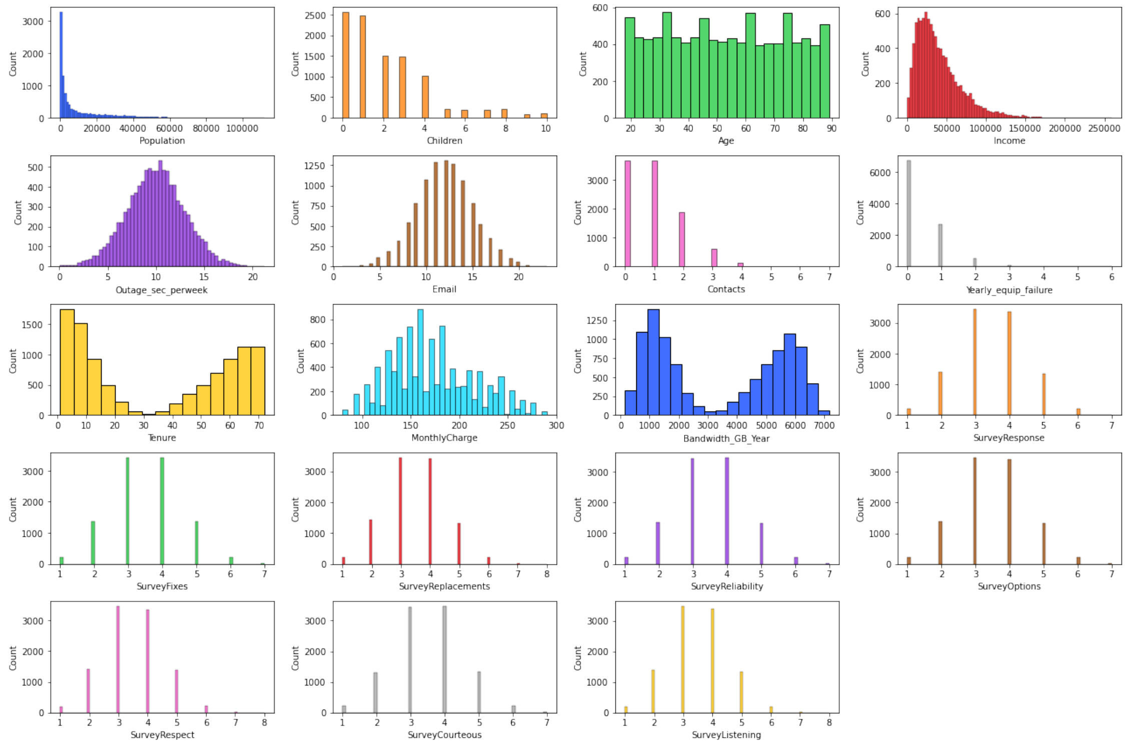
C4. Generate Visualizations Univariate Plots

[15]:
```python
# Plot histograms for continuous columns
continuous = churn_cp.select_dtypes(include='number').columns.tolist()

fig, axes = plt.subplots(5, 4, figsize=(18, 12))
palette1 = sns.color_palette('bright')

# Adapted from seaborn documentation (Waskom, 2022)
# https://seaborn.pydata.org/tutorial/color_palettes.html
x = 0
y = 0
color = 0
for col in continuous:
    if color == 10:
        color = 0
    if y == 4:
        x += 1
        y = 0
    sns.histplot(ax=axes[x, y], data=churn_cp[col], color=palette1[color])
    color += 1
    y += 1

# Delete empty subplot
fig.delaxes(axes[4, 3])
plt.tight_layout()
plt.show()
```



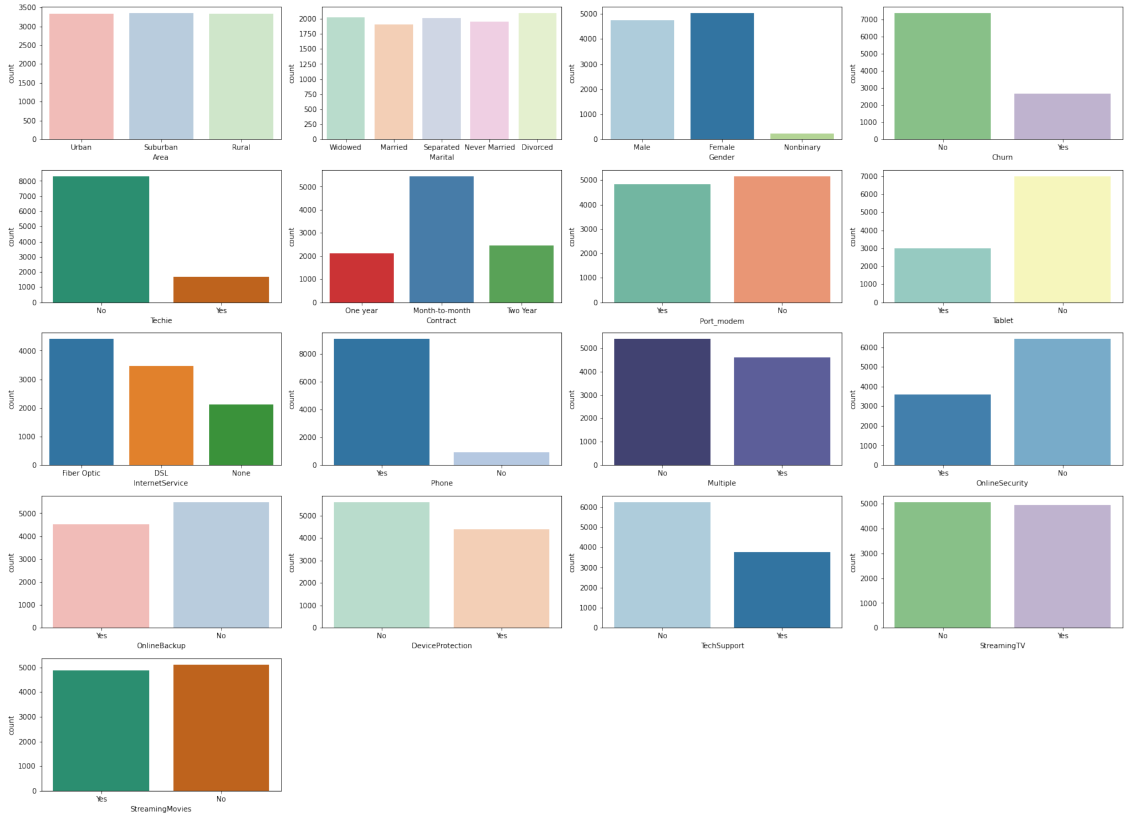
[16]:
```
# Plot barplots for categorical columns
categorical = churn_cp.select_dtypes(include='object').columns.tolist()

fig, axes = plt.subplots(5, 4, figsize=(22, 16))
palettes = ['Pastel1', 'Pastel2', 'Paired', 'Accent', 'Dark2', 'Set1', 'Set2',␣
↪'Set3', 'tab10', 'tab20', 'tab20b', 'tab20c']

x = 0
y = 0
color = 0
for col in categorical:
    if color == len(palettes):
        color = 0
    if y == 4:
        x += 1
        y = 0
    sns.countplot(ax=axes[x, y], x=col, data=churn_cp, palette=sns.
↪color_palette(f'{palettes[color]}'))
    color += 1
    y += 1

# Remove empty subplots
i = 3
```

```
while i > 0:
    axes[4, i].remove()
    i -= 1
plt.tight_layout()
plt.show()
```



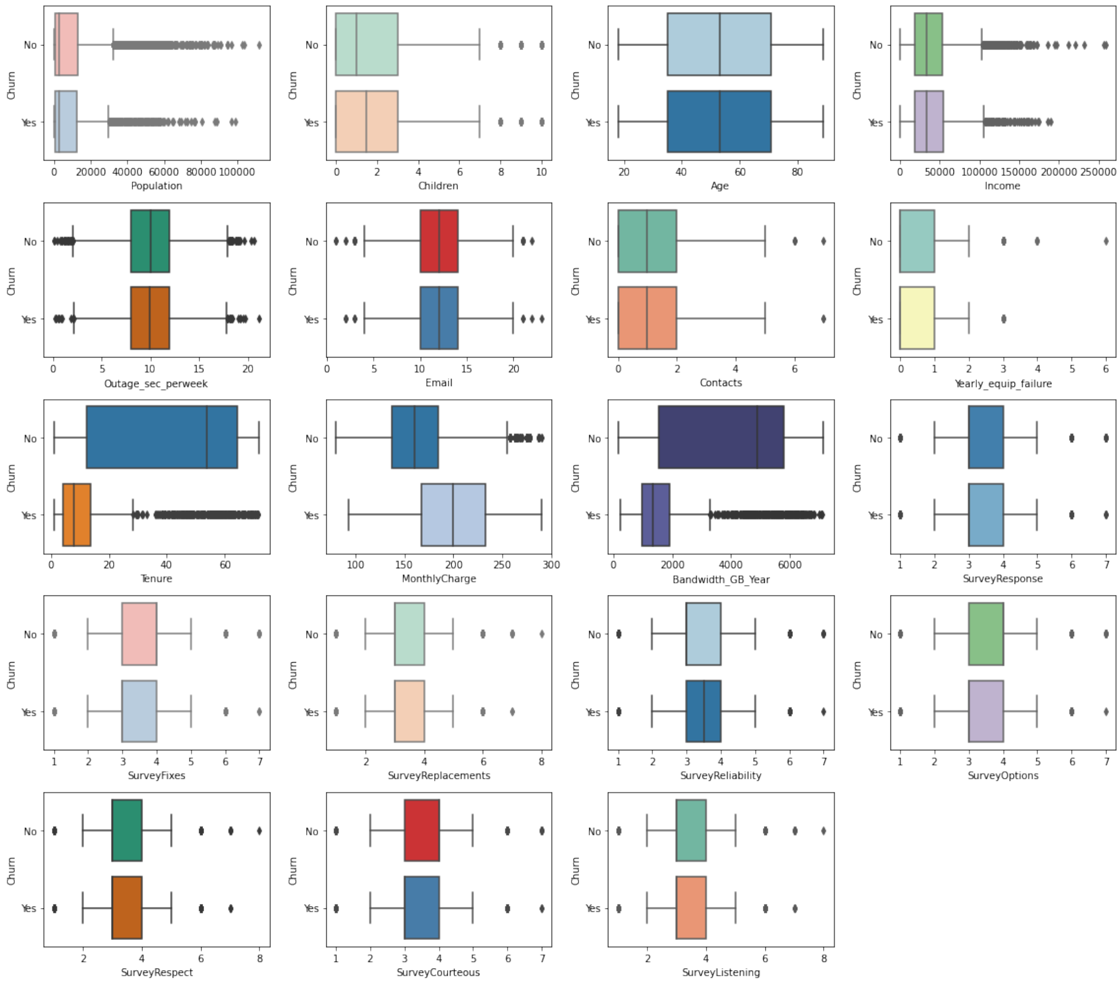
Bivariate Plots

[17]:
```
# Generate catplot with numerical columns
fig, axes = plt.subplots(5, 4, figsize=(16, 14))

x = 0
y = 0
color = 0
for col in continuous:
    if color == len(palettes):
        color = 0
    if y == 4:
        x += 1
        y = 0
    sns.boxplot(ax=axes[x, y], x=col, y='Churn', data=churn_cp, palette=sns.
↪color_palette(f'{palettes[color]}'))
```

```
    y += 1
    color += 1

# Delete empty subplot
fig.delaxes(axes[4, 3])
plt.tight_layout()
plt.show()
```

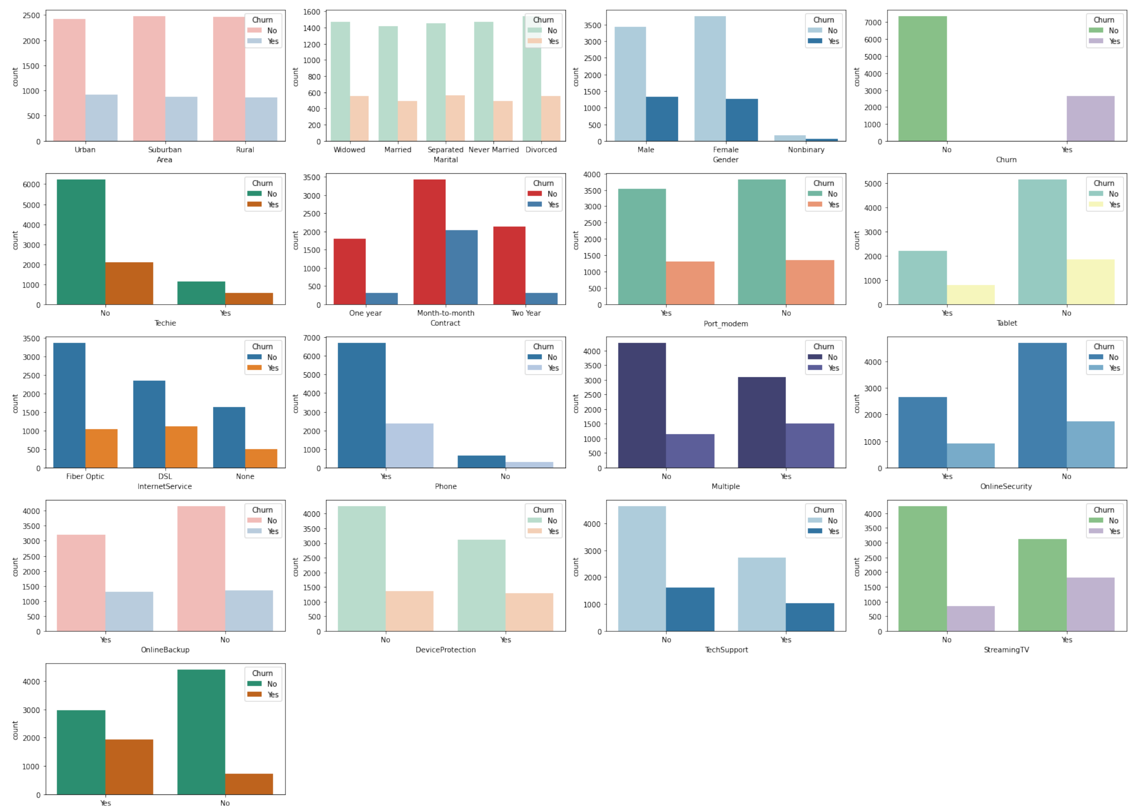


[18]:
```
# Generate countplot with Churn hue for categorical columns
fig, axes = plt.subplots(5, 4, figsize=(22, 16))

x = 0
y = 0
color = 0
for col in categorical:
    if color == len(palettes):
        color = 0
```

```
    if y == 4:
        x += 1
        y = 0
    sns.countplot(ax=axes[x, y], x=col, hue='Churn', data=churn_cp, palette=sns.
↪color_palette(f'{palettes[color]}'))
    y += 1
    color += 1

# Remove empty subplots
i = 3
while i > 0:
    axes[4, i].remove()
    i -= 1
plt.tight_layout()
plt.show()
```



*C5. Copy of Prepared Data

```
[19]: # Create a copy of the prepared data
churn_dmy.to_csv('churn_prepare_task_2.csv', index=False)
```

0.0.4 D. Compare Initial and Reduced Models

D1. Construct Initial Model

[20]:
```
# Adapted from Multiple Logistic Regression (Broeck, 2022a)
# https://app.datacamp.com/learn/courses/
 ↪intermediate-regression-with-statsmodels-in-python

# Create formula string for logistic regression
init_formula = 'Churn_Yes ~ '
init_predictors = churn_dmy.drop('Churn_Yes', axis=1).columns.tolist()
init_predictors = ' + '.join(init_predictors)
init_formula += init_predictors

# Run logistic regression
model = smf.logit(init_formula, data=churn_dmy).fit()
print(model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.379676
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:              Churn_Yes   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9974
Method:                           MLE   Df Model:                           25
Date:                Fri, 29 Apr 2022   Pseudo R-squ.:                  0.3434
Time:                        08:39:25   Log-Likelihood:                -3796.8
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
===============
                                  coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
---------------
Intercept                       1.1347      0.130      8.749      0.000
0.881       1.389
Population                  -5.714e-07      2e-06     -0.286      0.775
-4.48e-06    3.34e-06
Children                        0.0037      0.013      0.278      0.781
-0.023       0.030
Income                       3.876e-07   1.02e-06      0.382      0.703
-1.6e-06    2.38e-06
Contacts                        0.0361      0.029      1.247      0.212
-0.021       0.093
Yearly_equip_failure           -0.0285      0.045     -0.627      0.531
-0.118       0.061
Tenure                         -0.0637      0.002    -42.074      0.000
```

```
-0.067      -0.061
Area_Suburban                    -0.0085      0.071     -0.120      0.905
-0.147       0.130
Area_Urban                        0.0301      0.070      0.427      0.669
-0.108       0.168
Marital_Married                  -0.0209      0.091     -0.231      0.817
-0.198       0.157
Marital_Never_Married            -0.0448      0.090     -0.495      0.621
-0.222       0.133
Marital_Separated                 0.1538      0.089      1.732      0.083
-0.020       0.328
Marital_Widowed                   0.1205      0.089      1.355      0.176
-0.054       0.295
Gender_Male                       0.1463      0.058      2.516      0.012
0.032       0.260
Gender_Nonbinary                 -0.1077      0.189     -0.568      0.570
-0.479       0.264
Techie_Yes                        0.5451      0.075      7.300      0.000
0.399       0.691
Contract_One_year                -1.7845      0.080    -22.347      0.000
-1.941      -1.628
Contract_Two_Year                -1.8991      0.078    -24.217      0.000
-2.053      -1.745
Port_modem_Yes                    0.1055      0.057      1.838      0.066
-0.007       0.218
Tablet_Yes                       -0.0479      0.063     -0.762      0.446
-0.171       0.075
InternetService_Fiber_Optic      -0.7576      0.065    -11.576      0.000
-0.886      -0.629
InternetService_None             -0.7929      0.080     -9.919      0.000
-0.950      -0.636
OnlineSecurity_Yes               -0.0739      0.060     -1.230      0.219
-0.192       0.044
OnlineBackup_Yes                  0.4337      0.058      7.488      0.000
0.320       0.547
DeviceProtection_Yes              0.3149      0.058      5.455      0.000
0.202       0.428
TechSupport_Yes                   0.1021      0.059      1.727      0.084
-0.014       0.218
================================================================================
===============
```

D2. Variable Selection Procedure

[21]:
```python
# Get a list of variables with p > 0.05
high_p = []
for col in churn_dmy.columns:
    if col == 'Churn_Yes':
```

```
            pass
        else:
            p_val = model.pvalues[col]
            if p_val > 0.05:
                high_p.append(col)

# Perform backward selection by dropping columns with high p-values
churn_dmy.drop(high_p, axis=1, inplace=True)

churn_dmy.head()
```

[21]:

```
      Tenure  Gender_Male  Churn_Yes  Techie_Yes  Contract_One_year  \
0   6.795513            1          0           0                  1
1   1.156681            0          1           1                  0
2  15.754144            0          0           1                  0
3  17.087227            1          0           1                  0
4   1.670972            1          1           0                  0

   Contract_Two_Year  InternetService_Fiber_Optic  InternetService_None  \
0                  0                            1                     0
1                  0                            1                     0
2                  1                            0                     0
3                  1                            0                     0
4                  0                            1                     0

   OnlineBackup_Yes  DeviceProtection_Yes
0                 1                     0
1                 0                     0
2                 0                     0
3                 0                     0
4                 0                     0
```

D3. Reduced Regression Model

[22]:

```
# Create formula string for logistic regression
reduced_formula = 'Churn_Yes ~ '
reduced_predictors = churn_dmy.drop('Churn_Yes', axis=1).columns.tolist()
reduced_predictors = ' + '.join(reduced_predictors)
reduced_formula += reduced_predictors

# Run logistic regression
reduced_model = smf.logit(reduced_formula, data=churn_dmy).fit()
print(reduced_model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.380613
         Iterations 7
                           Logit Regression Results
```

```
==============================================================================
Dep. Variable:              Churn_Yes   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9990
Method:                           MLE   Df Model:                            9
Date:                Fri, 29 Apr 2022   Pseudo R-squ.:                  0.3418
Time:                        08:39:25   Log-Likelihood:                -3806.1
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
===============
                                 coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
---------------
Intercept                      1.2667      0.078     16.162      0.000
1.113       1.420
Tenure                        -0.0636      0.002    -42.099      0.000
-0.067      -0.061
Gender_Male                    0.1513      0.057      2.641      0.008
0.039       0.264
Techie_Yes                     0.5478      0.074      7.362      0.000
0.402       0.694
Contract_One_year             -1.7803      0.080    -22.355      0.000
-1.936      -1.624
Contract_Two_Year             -1.8963      0.078    -24.247      0.000
-2.050      -1.743
InternetService_Fiber_Optic   -0.7590      0.065    -11.623      0.000
-0.887      -0.631
InternetService_None          -0.7879      0.080     -9.887      0.000
-0.944      -0.632
OnlineBackup_Yes               0.4320      0.058      7.478      0.000
0.319       0.545
DeviceProtection_Yes           0.3150      0.058      5.468      0.000
0.202       0.428
================================================================================
===============
```

0.0.5 E. Analyze Dataset

E1. Analysis Process Variable Selection Logic

The process of backward selection removed the variables with high p-values in the initial model. The stopping rule is the p-value threshold of 0.05 (JMP, 2019). Another factor to consider with these variables was that they had small coefficients. `Income` was a variable that had a high p-value of 0.703 and an extremely small coefficient of 3.876e-07. Removing this variable had little impact on the model. The same thing could be applied to other variables with high p-value. They also had small coefficients that were unlikely to affect the model when removed.

Model Evaluation Metric

The basis of all performance metrics for logistic regression required a confusion matrix. This matrix contained the counts of each actual and predicted response pair (Broeck, 2022b).

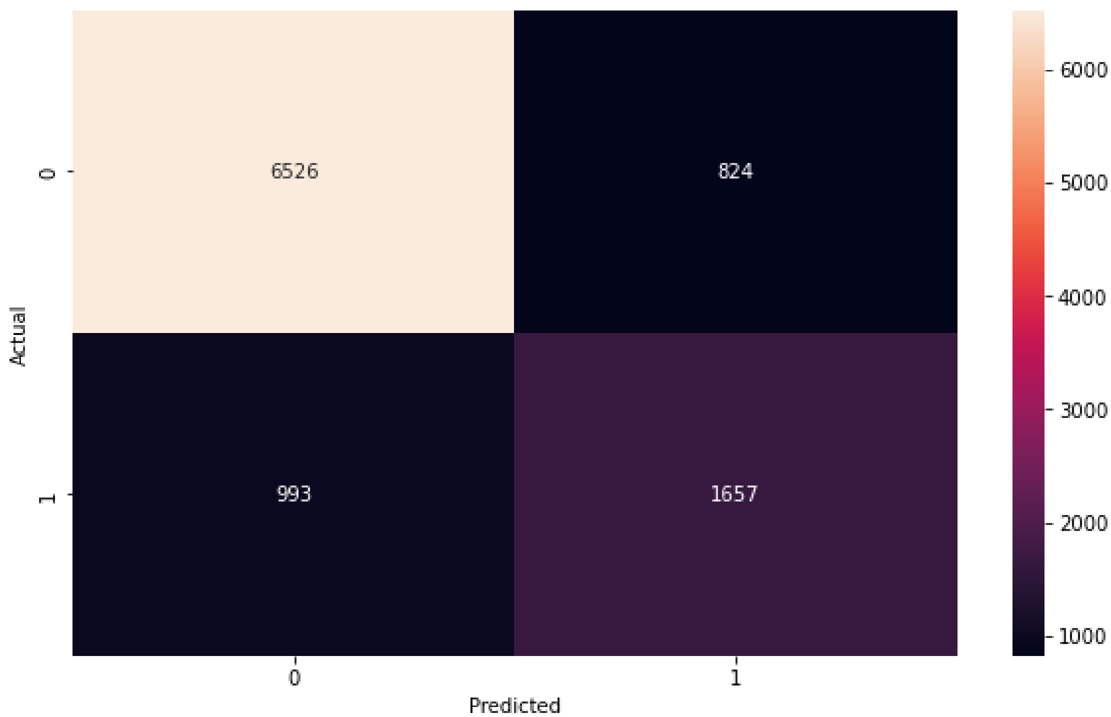
[23]:
```
# Confusion matrix of the reduced model
conf_matrix = reduced_model.pred_table()

# Extract TN, TP, FN and FP from conf_matrix
TN = conf_matrix[0,0]
TP = conf_matrix[1,1]
FN = conf_matrix[1,0]
FP = conf_matrix[0,1]

plt.figure(figsize=(10, 6))

# Plot matrix
p = sns.heatmap(conf_matrix, annot=True, fmt='g')

p.set_xlabel('Predicted')
p.set_ylabel('Actual')
plt.show()
```



[24]:
```
accuracy = (TN + TP) / (TN + FN + FP + TP)
sensitivity = TP / (TP + FN)
specificity = TN / (TN + FP)
```

```
print(f'Accuracy: {accuracy:.4f}')
print(f'Sensitivity: {sensitivity:.4f}')
print(f'Specificity: {specificity:.4f}')
```

```
Accuracy: 0.8183
Sensitivity: 0.6253
Specificity: 0.8879
```

The three metrics to evaluate model fit for logistic regression:

1. Accuracy - proportion of correct predictions.
2. Sensitivity - proportion of observations where both the actual and predicted responses were true.
3. Specificity - proportion of observations where both the actual and predicted responses were false.

The model accuracy of 0.8183 meant that the percentage of correct predictions was 81.83%. Given that the model only used ten independent variables, this level of accuracy could be acceptable. However, the low sensitivity score of 62.53% indicated the percentage where the model correctly predicted the customers would churn. This does not make the model very good at predicting actual churn since its correct predictions were only a little better than half. This was also due to the fact that there was a trade-off where a lower sensitivity would led to higher specificity. The specificity of 88.79% was good since the model predicted correctly the majority of the time where customers would not churn.

E2. Analysis Output The code and output for the initial and reduced model could be found in section D1 and D3, respectively. The confusion matrix and the evaluation metrics could be found in section E1 above.

The predictions from the reduced model could be found below:

[25]:
```
predictions = np.round(reduced_model.predict())
predictions
```

[25]:
```
array([0., 1., 0., …, 0., 0., 0.])
```

E3. Logistic Regression Code The code that implemented the logistic regression could be found in section D1 for the initial model and D3 for the reduced model.

0.0.6 F. Summarize Findings

F1. Results of Analysis **Regression Equation**

[26]:
```
# Lists of variables and their coefficients
cols = churn_dmy.columns.tolist()
coefs = reduced_model.params.tolist()

# Create equation string for the reduced model
```

```python
equation = f'y = {coefs[0]:.2f}'
for col, coef in zip(cols[1:], coefs[1:]):
    equation += f' + ({coef:.2f} * {col})'

print(f'Regression Equation:\n{equation}')
```

```
Regression Equation:
y = 1.27 + (-0.06 * Gender_Male) + (0.15 * Churn_Yes) + (0.55 * Techie_Yes) +
(-1.78 * Contract_One_year) + (-1.90 * Contract_Two_Year) + (-0.76 *
InternetService_Fiber_Optic) + (-0.79 * InternetService_None) + (0.43 *
OnlineBackup_Yes) + (0.31 * DeviceProtection_Yes)
```

Interpretation of Coefficients

Having a contract indicated that customers would not churn. This made sense since customers who signed contracts were unlikely to disconnect their service due to early termination fee and other costs involved. The positive coefficients were `Techie`, `OnlineBackup`, and `DeviceProtection`. These variables were related to customers who tend to have additional devices and services, which also indicated a higher cost. An interesting thing to note was that customers were also less likely to churn whether they had internet service or not. Since these two variables were mutually exclusive, they should be opposite of each other. This could indicate that a factor outside these variables influenced the model.

Model Significance

All of the independent variables in the equation are statistically significant. This means that the relationships between these variables and `Churn` are statistically significant. Changes in these independent variables are associated with changes in the dependent variable when applied to the population (Frost, 2021). At the same time, there could be other factors that could influence either the coefficients or the p-values of these variables. The original dataset contained many independent variables that were reduced for the analysis. This would be an important case where having domain expertise will help the analysis identify the important variables.

Limitations of Analysis

There were multiple limitations to this analysis. The first was mentioned above, where reducing too many variables could have affected the accuracy of the final result. The second was that creating dummy variables for the categorical columns could have created some redundancy. The data generated from these dummy variables could have created some bias in the model, leading to some inaccuracy. The third limitation was that many variables in the original dataset indicated multicollinearity. This led to those variables being removed when using variance inflation factor as a measure, even though some of the variables could be significant in the model.

F2. Recommendations Based on the logistic regression model, there were three recommendations that stakeholders could implement to decrease the churn rate:

1. Have customers sign up for a contract when they start service. The model indicated that having a contract will lead to no churn on those customers. Customers who signed up for month-to-month service were significantly more likely to disconnect their services.
2. Look out for customers who were technically inclined with multiple devices and services. These customers were more likely to churn due to the fact that they might be paying more

than other customers. Stakeholders could potentially retain these customers by offering lower bundle prices for additional services.

3. Offer internet service to customers at a discount rate. The model indicated that customers would continue service whether they signed up for internet service or not. However, customers should have internet service since that could lead to additional upsell on other services like streaming TV and streaming movies. Customers would also be less likely to leave if they had multiple services with our company.

0.0.7 G. Panopto Recording

Link: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=b1325549-e836-48bb-9497-ae860014c2b6

0.0.8 H. Third-Party Code

GeeksforGeeks. (2020, August 30). Detecting Multicollinearity with VIF - Python. Retrieved April 27, 2022, from https://www.geeksforgeeks.org/detecting-multicollinearity-with-vif-python/

Waskom, M. (2022). Choosing color palettes — seaborn 0.11.2 documentation. Seaborn Documentation. Retrieved April 27, 2022, from https://seaborn.pydata.org/tutorial/color_palettes.html

0.0.9 I. References

Broeck, M. (2022a). Intermediate Regression with statsmodels in Python. DataCamp. Retrieved April 28, 2022, from https://app.datacamp.com/learn/courses/intermediate-regression-with-statsmodels-in-python

Broeck, M. (2022b). Introduction to Regression with statsmodels in Python. DataCamp. Retrieved April 28, 2022, from https://app.datacamp.com/learn/courses/introduction-to-regression-with-statsmodels-in-python

Frost, J. (2022, February 27). How to Interpret P-values and Coefficients in Regression Analysis. Statistics By Jim. Retrieved April 28, 2022, from https://statisticsbyjim.com/regression/interpret-coefficients-p-values-regression/

Joby, A. (2021, July 29). What Is Logistic Regression? Learn When to Use It. G2. Retrieved April 27, 2022, from https://learn.g2.com/logistic-regression

Rane, Z. (2021, August 19). 10 compelling reasons to learn Python for data science. Medium. Retrieved April 20, 2022, from https://towardsdatascience.com/10-compelling-reasons-to-learn-python-for-data-science-fa31160321cb#1faf

Statistics Solutions. (2021, August 11). Assumptions of Logistic Regression. Retrieved April 27, 2022, from https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/assumptions-of-logistic-regression/

Variable Selection in Multiple Regression. (2019). Introduction to Statistics | JMP. Retrieved April 28, 2022, from https://www.jmp.com/en_in/statistics-knowledge-portal/what-is-multiple-regression/variable-selection.html

[26]:

```
```