Predictive Modeling
Task 2

Briana Churchill

Student ID: 011009463

Western Governors University, Data Analytics M.S. D208: Predictive Modeling

Dr. Eric Straw

September 3, 2023

Table of Contents

Part I: Research Question

A1. Question and Method Used to Analyze

A research question I am interested in answering given the available dataset is "What factors are primarily contributed to customer churn?" Answering this question involves analyzing one dependent variable, and multiple independent variables utilizing logistic regression. Assessing this question could determine potential relationships between certain factors and customers who have signed churned. Churn and acquiring new customers can be costly to the business. Maximizing the probability of customers remaining loyal consumers longer would be a financial benefit to the company.

A2. Goals of the Data Analysis

Determining which factors contribute most significantly to customer churn is the main goal for this analysis. Using a logistic regression model will allow this objective to be obtained through analysis of the regression statistics and visualizations of the data. Narrowing down the independent variables and pinpointing which factors are significant to the research question will give the telecommunications company information that could improve business practices.

Part II: Method Justification

B1. Four Assumptions of a Logistic Regression Model

According to Zach from Statology, there are five assumptions of a logistic regression model. Here are four them:

- "The Response Variable is Binary
- The Observations are Independent
- There is No Multicollinearity Among Explanatory Variables
- There are No Extreme Outliers" (Zach, 2020)

B2. Benefits of Using Python

In previous assessments I have been utilizing Python so the first great benefit of this choice is that I had reliable scripts from 2D07 and D208 task 1 that I could utilize again for this assessment. These scripts took care of the data cleaning code needed. The previous scripts allowed me to build off code used previously for univariate and bivariate explorations.

However, I still needed the help of resources cited below for various aspects of this analysis. Learning new scripts and implementing them can sometime end in error. The error messages from Python are descriptive and show exactly what line needs to be corrected. Being able to understand any errors makes the process smoother and user friendly.

B3. Why Logistic Regression is an Appropriate Technique

To answer the research question, I will be seeking to understand relationships between the dependent and independent variables. In doing so, I will also estimate probabilities. For this analysis, the dependent variable (churn) is categorical. The independent variables mentioned above vary between categorical and numeric variables. Given the nature of the research question and the necessary steps to answer it, logistic regression is an appropriate technique according to the course material. (2023, Middleton)

Part III: Data Preparation

C1. Data cleaning goals and the steps used

Given that the CSV file for this assessment is the same as previous assessments, the scripts used in D207 and D208 were used again to clean the data. Confirmation that there were no null or duplicate values was done again anyway, to be sure. The main focus of the cleaning was to rename variables. Many of the original values were two words without spaces, and others were longer than necessary. The survey questions being named "Item1" through "Item8" was quite confusing, so they were renamed the applicable questions asked in the survey. The functions and scripts I used can be seen here:

```
# Clean up and prepare the data
# Rename the Monthlycharge column for proper spacing
df = df.rename(columns={'MonthlyCharge': 'Monthly_Charge'});

# Rename the Bandwidth_GB_Year column for added clarification
df = df.rename(columns={'Bandwidth_GB_Year': 'Bandwidth_Usage'});

# Rename the Yearly_equip_failure column for proper spacing
df = df.rename(columns={'Yearly_equip_failure': 'Equipment_failure'})

# Rename the Outage_sec_perweek column for proper spacing
df = df.rename(columns={'Outage_sec_perweek': 'Outages'})

# Rename the Onlinesecurity column for proper spacing
df = df.rename(columns={'OnlineSecurity': 'Online_Security'});

# Rename the Onlinebackup column for proper spacing
df = df.rename(columns={'OnlineBackup': 'Online_Backup'});

# Rename the Internetservice column for proper spacing
df = df.rename(columns={'InternetService': 'Internet_Service'});

# Rename the Item1 column for added clarification
df = df.rename(columns={'Item1': 'Timely_Response'});

# Rename the Item2 column for added clarification
df = df.rename(columns={'Item2': 'Timely_Fixes'});

# Rename the Item3 column for added clarification
df = df.rename(columns={'Item3': 'Timely_Replacements'});

# Rename the Item4 column for added clarification
df = df.rename(columns={'Item4': 'Reliability'});

# Rename the Item5 column for added clarification
df = df.rename(columns={'Item5': 'Options'});

# Rename the Item6 column for added clarification
df = df.rename(columns={'Item6': 'Respectful_Response'});

# Rename the Item7 column for added clarification
df = df.rename(columns={'Item7': 'Courteous_Exchange'});

# Rename the Item8 column for added clarification
df = df.rename(columns={'Item8': 'Active_Listening'});
```

C2. Describe and show the summary statistics for all variables used

Updating previously used scripts that utilized the .value_counts() and .describe() functions, the summary statistics were obtained for all variables used. For the dependent variable and all categorical independent variables, the .value_counts function showed the number of customers within each applicable data point (i.e.: Churn yes/no). For the remaining independent variables which were continuous and numeric in nature, .describe() showed the summary statistics (i.e.: min, max, etc.) for the data within each variable.

The summary statistics for the independent variable (churn) were obtained prior to the remaining columns. Of the 10,000 customers in the data set, 7,350 customers had not churned and 2,650 had.

```
# View data counts to ensure appropriate values for Churn
df.Churn.value_counts()
```

```
]:  No      7350
    Yes     2650
    Name: Churn, dtype: int64
```

Customer Income varies between $348.67 and $258,900. The data found in the income column are a yearly value and is in USD. The average income is $39,806.93.

```
# View data counts to ensure appropriate values for Income
df.Income.describe()
```

```
0]:  count     10000.000000
     mean      39806.926771
     std       28199.916702
     min         348.670000
     25%       19224.717500
     50%       33170.605000
     75%       53246.170000
     max      258900.700000
     Name: Income, dtype: float64
```

Service outages are measured in seconds per week. The length of outages customers have experienced weekly is between 0.0997 seconds and 21.207 seconds.

```
# View data counts to ensure appropriate values for Outages
df.Outage_sec_perweek.describe()
```

```
]:  count    10000.000000
    mean        10.001848
    std          2.976019
    min          0.099747
    25%          8.018214
    50%         10.018560
    75%         11.969485
    max         21.207230
    Name: Outage_sec_perweek, dtype: float64
```

Equipment failures are measured in occurrences. 75% of customers experienced at least one equipment failure per year. The maximum number of failures is 6 occurrences in one year.

```
# View values for Yearly equipment failure
df.Yearly_equip_failure.describe()
```

```
3]:  count    10000.000000
     mean         0.398000
     std          0.635953
     min          0.000000
     25%          0.000000
     50%          0.000000
     75%          1.000000
     max          6.000000
     Name: Yearly_equip_failure, dtype: float64
```

Customers typically seem to lean towards month-to-month contracts, as those customers make up the majority at 5,456 customers with that contract. 2,442 customers have a two-year contract. The remaining 2,102 customers have a one-year contract.

```
# View data counts to ensure appropriate values for Contract
df.Contract.value_counts()
```

```
]: Month-to-month    5456
   Two Year          2442
   One year          2102
   Name: Contract, dtype: int64
```

Most customers have Fiber Optic internet service, with 4,408 customers being in that category. 3,463 customers utilize DSL internet service. There are 2,129 customers that have no internet service plan at all.

```
# View data counts to ensure appropriate values for Internet Service
df.InternetService.value_counts()
```

```
5]: Fiber Optic    4408
    DSL            3463
    None           2129
    Name: InternetService, dtype: int64
```

At 4,608 almost half of all customers have multiple phone lines. Most customers (5,392) do not.

```
# View data counts to ensure appropriate values for Multiple
df.Multiple.value_counts()
```

```
5]: No     5392
    Yes    4608
    Name: Multiple, dtype: int64
```

More customers do not opt into Online Security or Online Backup. Only 3,576 customers opted into Online Security. 4,506 opt into online backup.

```
# View data counts to ensure appropriate values for OnlineSecurity
df.OnlineSecurity.value_counts()
```

```
']: No     6424
    Yes    3576
    Name: OnlineSecurity, dtype: int64
```

```
# View data counts to ensure appropriate values for OnlineBackup
df.OnlineBackup.value_counts()
```

```
]: No     5494
   Yes    4506
   Name: OnlineBackup, dtype: int64
```

Tenure varies between ~1 month and ~72 months (about 6 years). The average tenure per customer is 34.53 months (about 3 years).

```
# View data counts to ensure appropriate values for Tenure
df.Tenure.describe()
```

```
9]: count    10000.000000
    mean        34.526188
    std         26.443063
    min          1.000259
    25%          7.917694
    50%         35.430507
    75%         61.479795
    max         71.999280
    Name: Tenure, dtype: float64
```

The amount for which customers are charged monthly varies between 79.98 to 290.16. The average monthly charge is 172.62

```
# View data counts to ensure appropriate values for Monthly Charge
df.MonthlyCharge.describe()
```

```
)]: count     10000.000000
    mean        172.624816
    std          42.943094
    min          79.978860
    25%         139.979239
    50%         167.484700
    75%         200.734725
    max         290.160419
    Name: MonthlyCharge, dtype: float64
```

Bandwidth usage is measured yearly in GB. The values vary between 155.51GB and 7,158.98GB. The average amount of usage is 3,392.34GB yearly.

```
# View data counts to ensure appropriate values for Bandwidth
df.Bandwidth_GB_Year.describe()
```

```
1]: count     10000.000000
    mean       3392.341550
    std        2185.294852
    min         155.506715
    25%        1236.470827
    50%        3279.536903
    75%        5586.141370
    max        7158.981530
    Name: Bandwidth_GB_Year, dtype: float64
```

Item1 has values related to the survey question relating to the importance of timely response from the company. More customers lean towards “important” (value: 1) with 3,448 customers responding “3” and only 19 customers voting “7”, which is closer to “not important”.

```
# View data counts to ensure appropriate values for Item1
df.Item1.value_counts()
```

```
2]: 3    3448
    4    3358
    2    1393
    5    1359
    1     224
    6     199
    7      19
    Name: Item1, dtype: int64
```

Item2 has values related to the survey question relating to the importance of timely fixes by the company. More customers lean towards “important” (value: 1) with 3,415 customers responding “3” and only 13 customers voting “7”, which is closer to “not important”.

```
# View data counts to ensure appropriate values for Item2
df.Item2.value_counts()
```

```
3]: 3    3415
    4    3412
    5    1368
    2    1360
    1     217
    6     215
    7      13
    Name: Item2, dtype: int64
```

Item3 has values related to the survey question relating to the importance of timely replacements by the company. More customers lean towards "important" (value: 1) with 3,435 customers responding "3" and only 1 customer voting "8", which is "not important".

```
# View data counts to ensure appropriate values for Item3
df.Item3.value_counts()
```

```
4]: 3    3435
    4    3410
    2    1424
    5    1313
    6     203
    1     202
    7      12
    8       1
    Name: Item3, dtype: int64
```

Item4 has values related to the survey question relating to the importance of the reliability of the company. More customers lean towards "important" (value: 1) with 3,452 customers responding "3" and only 12 customers voting "7", which is closer to "not important".

```
# View data counts to ensure appropriate values for Item4
df.Item4.value_counts()
```

```
5]: 4    3452
    3    3430
    2    1350
    5    1335
    1     221
    6     203
    7       9
    Name: Item4, dtype: int64
```

Item5 has values related to the survey question relating to the importance of options for the customer. More customers lean towards "important" (value: 1) with 3,462 customers responding "3" and only 12 customers voting "7", which is closer to "not important".

```
# View data counts to ensure appropriate values for Item5
df.Item5.value_counts()
```

```
5]: 3    3462
    4    3417
    2    1378
    5    1321
    1     206
    6     204
    7      12
    Name: Item5, dtype: int64
```

Item6 has values related to the survey question relating to the importance of respectful responses from the company. More customers lean towards “important” (value: 1) with 3,445 customers responding “3” and only 1 customer voting “8”, which is “not important”.

```
# View data counts to ensure appropriate values for Item6
df.Item6.value_counts()
```

```
7]: 3    3445
    4    3333
    2    1427
    5    1382
    6     210
    1     190
    7      12
    8       1
    Name: Item6, dtype: int64
```

Item7 has values related to the survey question relating to the importance of a courteous exchange with the company. More customers lean towards “important” (value: 1) with 3,456 customers responding with “4” and 3,446 customers responding “3.” Only 11 customers voted for “7”, which is closer to “not important”.

```
# View data counts to ensure appropriate values for Item7
df.Item7.value_counts()
```

```
8]: 4    3456
    3    3446
    5    1335
    2    1309
    6     224
    1     219
    7      11
    Name: Item7, dtype: int64
```

Item8 has values related to the survey question relating to the importance of active listening on the company's behalf. More customers lean towards “important” (value: 1) with 3,461 customers responding “3” and only 1 customer voting “8”, which is “not important”.

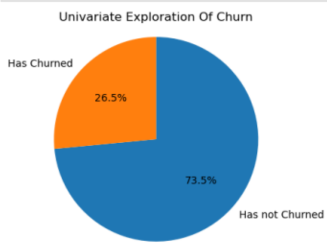
```
# View data counts to ensure appropriate values for Item8
df.Item8.value_counts()
```

```
3    3461
4    3400
2    1378
5    1335
1     206
6     205
7      14
8       1
Name: Item8, dtype: int64
```

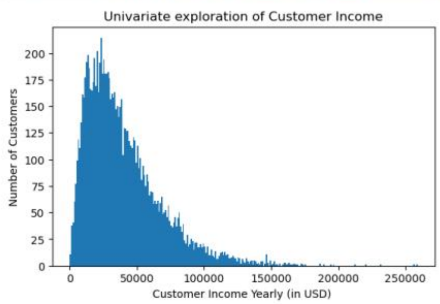
C3. Generate univariate and bivariate visualizations

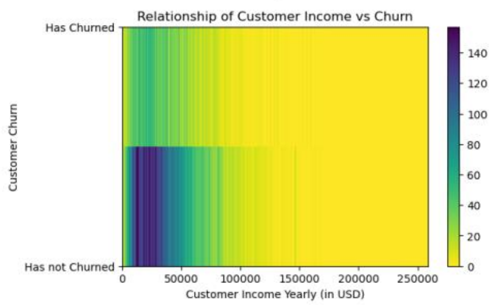
For the first plot, I completed a univariate exploration of the dependent variable Churn. The remaining plots are of several types as can be seen (pie chart, histogram, etc.) and are of univariate explorations of the independent variables. There are additional plots (i.e.: violin plot, mosaic, etc.) for bivariate explorations of each independent variable with the dependent variable.

```
# First plot: Univariate exploration of Churn
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Univariate Exploration Of Churn")
Churn_counts = model_df["Churn"].value_counts().sort_index()
plt.pie(Churn_counts, labels=["Has not Churned", "Has Churned"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');
```

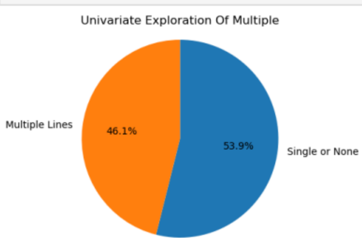


```
# First plot: Univariate exploration of Income
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title('Univariate exploration of Customer Income')
bins = np.arange(0, df.Income.max() + 500, 1000)
plt.hist(data=model_df, x="Income", bins=bins)
plt.xlabel('Customer Income Yearly (in USD) ')
plt.ylabel("Number of Customers")

# Second plot: Bivariate exploration of Income vs Churn
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 2)
plt.title("Relationship of Customer Income vs Churn")
bins_y = np.arange(0, 1.25, 0.5)
plt.hist2d(data= model_df, x="Income", y="Churn", bins=[bins, bins_y], cmap= "viridis_r")
plt.colorbar()
plt.xlabel("Customer Income Yearly (in USD)")
plt.ylabel("Customer Churn")
plt.yticks([0,1], ["Has not Churned", "Has Churned"]);
```



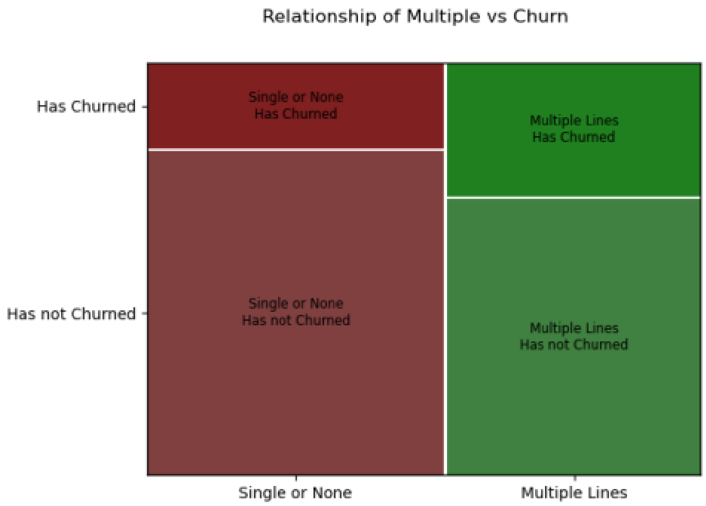



```
# First plot: Univariate exploration of Multiple variable
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Univariate Exploration Of Multiple")
Multiple_counts = model_df["Multiple"].value_counts().sort_index()
plt.pie(Multiple_counts, labels=["Single or None", "Multiple Lines"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');
```

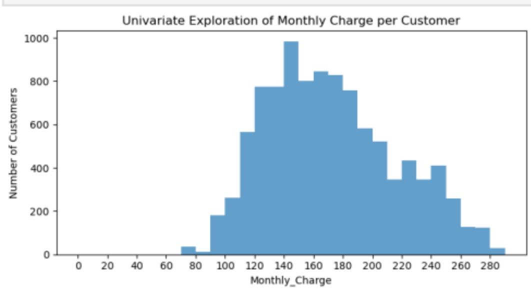


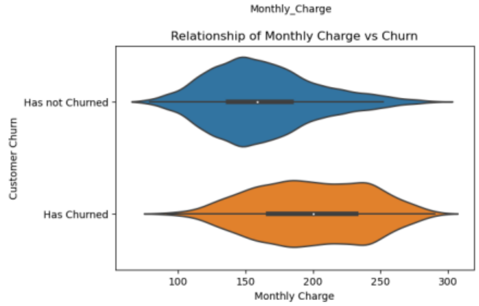
```
# Second plot: Bivariate exploration of Multiple vs Churn
plt.figure(figsize = [14,4])
MC_df = df[["Multiple", "Churn"]].copy()
Multiple_map = {1 : "Multiple Lines", 0: "Single or None"}
Churn_map = {1 : "Has Churned", 0: "Has not Churned"}
MC_df["Multiple"] = MC_df["Multiple"].map(Multiple_map)
MC_df["Churn"] = MC_df["Churn"].map(Churn_map)
mosaic(MC_df, ["Multiple", "Churn"])
plt.suptitle("Relationship of Multiple vs Churn");
```

```
<Figure size 1400x400 with 0 Axes>
```



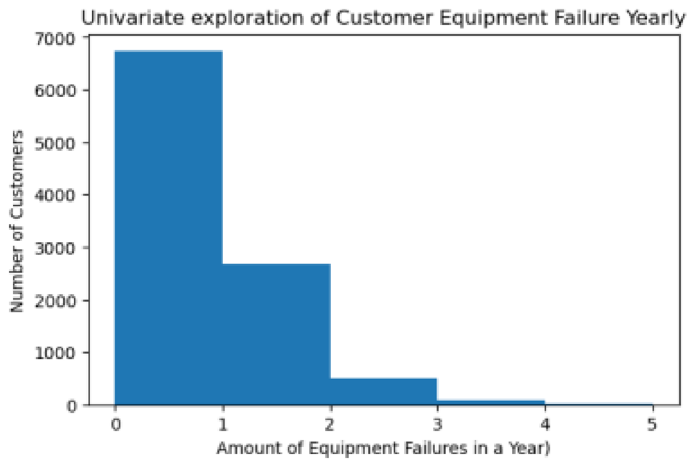

```
# First plot: Univariate exploration of Monthly Charge variable
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title('Univariate Exploration of Monthly Charge per Customer')
bins = np.arange(0, df.Monthly_Charge.max() + 10, 10)
plt.hist(data=model_df, x="Monthly_Charge", bins=bins, alpha=0.7)
plt.xlabel('Monthly_Charge')
plt.ylabel("Number of Customers")
plt.xticks(np.arange(0, df['Monthly_Charge'].max() + 10, 20))
plt.tight_layout()

# Second plot: Univariate exploration of Monthly Charge vs Churn
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 2)
plt.title("Relationship of Monthly Charge vs Churn")
sns.violinplot(data = model_df, x="Monthly_Charge", y="Churn", orient='h')
plt.xlabel("Monthly Charge")
plt.ylabel("Customer Churn")
plt.yticks([0,1], ["Has not Churned", "Has Churned"]);
```



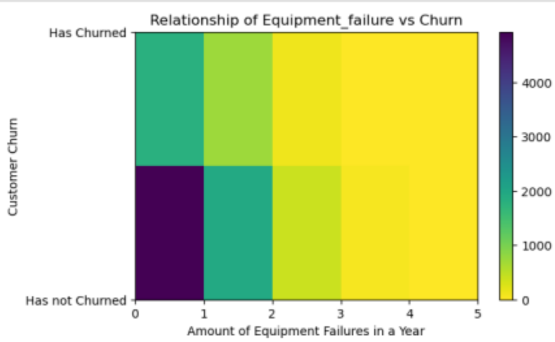


```
# First plot: Univariate exploration of Customer Equipment Failure
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Univariate exploration of Customer Equipment Failure Yearly")
bins = np.arange(0, model_df.Equipment_failure.max())
plt.hist(data=model_df, x="Equipment_failure", bins=bins)
plt.xlabel('Amount of Equipment Failures in a Year)')
plt.ylabel("Number of Customers")
```

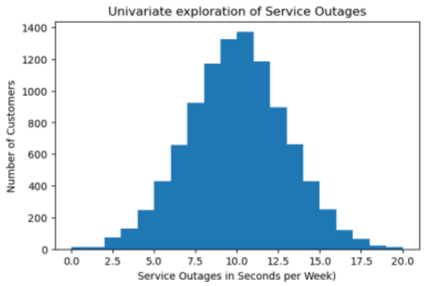
```
8]: Text(0, 0.5, 'Number of Customers')
```

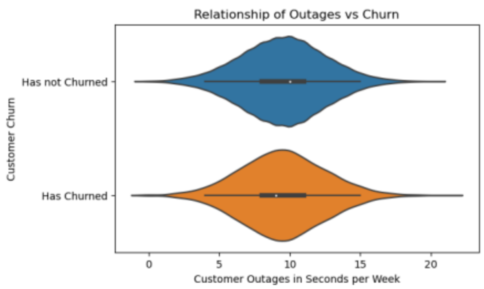



```
# Second plot: Bivariate exploration of Equipment_failure vs Churn
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 2)
plt.title("Relationship of Equipment_failure vs Churn")
bins_y = np.arange(0, 1.25, 0.5)
plt.hist2d(data= model_df, x="Equipment_failure", y="Churn", bins=[bins, bins_y], cmap= "viridis_r")
plt.colorbar()
plt.xlabel("Amount of Equipment Failures in a Year")
plt.ylabel("Customer Churn")
plt.yticks([0,1], ["Has not Churned", "Has Churned"]);
```



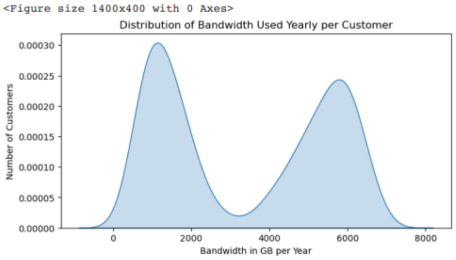
```
# First plot: Univariate exploration of Outage Variable
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title('Univariate exploration of Service Outages')
bins = np.arange(0, df.Outages.max())
plt.hist(data=model_df, x="Outages", bins=bins)
plt.xlabel('Service Outages in Seconds per Week)')
plt.ylabel("Number of Customers")

# Second plot: Bivariate exploration of Outages vs Churn
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 2)
plt.title("Relationship of Outages vs Churn")
sns.violinplot(data = model_df, x="Outages", y="Churn", orient='h')
plt.xlabel("Customer Outages in Seconds per Week")
plt.ylabel("Customer Churn")
plt.yticks([0,1], ["Has not Churned", "Has Churned"]);
```



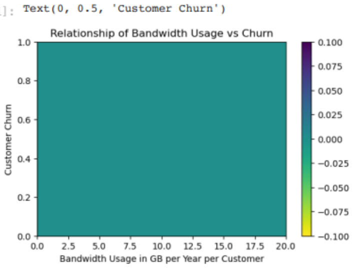



```
# First plot: Univariate exploration of Bandwidth Variable
plt.figure(figsize = [14,4])
plt.figure(figsize=(8, 4))
sns.kdeplot(model_df['Bandwidth_Usage'], fill=True)
plt.title("Distribution of Bandwidth Used Yearly per Customer")
plt.xlabel("Bandwidth in GB per Year")
plt.ylabel("Number of Customers")
plt.show()

# Second plot: Bivariate exploration of Bandwidth vs Churn
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 2)
plt.title("Relationship of Bandwidth Usage vs Churn")
bins_y = np.arange(0, 1.25, 0.5)
plt.hist2d(data= model_df, x="Bandwidth_Usage", y="Churn", bins=[bins, bins_y], cmap= "viridis_r")
plt.colorbar()
plt.xlabel("Bandwidth Usage in GB per Year per Customer")
plt.ylabel("Customer Churn")
```

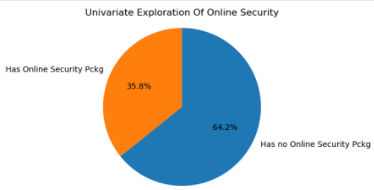
```
<Figure size 1400x400 with 0 Axes>
```

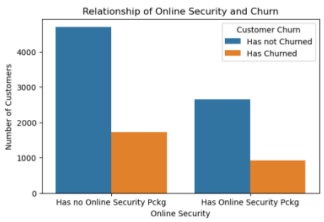



```
[]: Text(0, 0.5, 'Customer Churn')
```

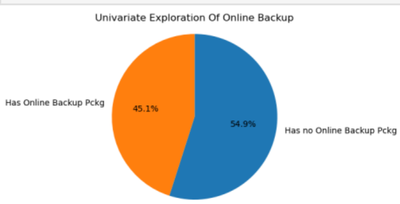


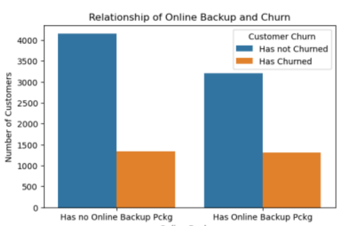
```python
# First plot: Univariate exploration of Online_Security Variable
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Univariate Exploration Of Online Security")
Online_Security_counts = model_df["Online_Security"].value_counts().sort_index()
plt.pie(Online_Security_counts, labels=["Has no Online Security Pckg", "Has Online Security Pckg"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# First plot: Bivariate exploration of Online_Security Vs Churn
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 2)
plt.title("Relationship of Online Security and Churn")
sns.countplot(data = model_df, x="Online_Security", hue="Churn")
plt.legend(title="Customer Churn", labels=["Has not Churned", "Has Churned"])
plt.xlabel("Online Security")
plt.ylabel("Number of Customers")
plt.xticks([0,1,], ["Has no Online Security Pckg", "Has Online Security Pckg"]);
```





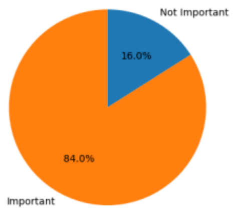

```python
# First plot: Univariate exploration of Online_Backup Variable
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Univariate Exploration Of Online Backup")
Online_Backup_counts = model_df["Online_Backup"].value_counts().sort_index()
plt.pie(Online_Backup_counts, labels=["Has no Online Backup Pckg", "Has Online Backup Pckg"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# Second plot: Bivariate exploration of Online_Backup Vs Churn
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 2)
plt.title("Relationship of Online Backup and Churn")
sns.countplot(data = model_df, x="Online_Backup", hue="Churn")
plt.legend(title="Customer Churn", labels=["Has not Churned", "Has Churned"])
plt.xlabel("Online Backup")
plt.ylabel("Number of Customers")
plt.xticks([0,1,], ["Has no Online Backup Pckg", "Has Online Backup Pckg"]);
```





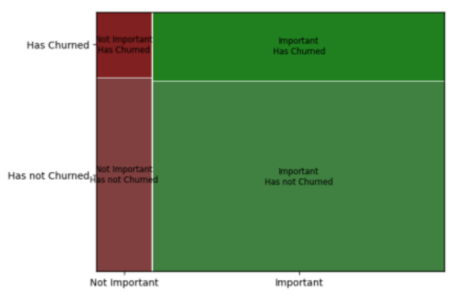
```
# First plot: Univariate exploration of Timely Fixes Variable
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Univariate Exploration Of Timely Fixes Variable")
Timely_Fixes_counts = model_df["Timely_Fixes"].value_counts().sort_index()
plt.pie(Timely_Fixes_counts, labels=["Not Important", "Important"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# Second plot: Bivariate exploration of Timely Fixes Vs Churn
plt.figure(figsize = [14,4])
TFC_df = df[["Timely_Fixes", "Churn"]].copy()
Timely_Fixes_map = {1 : "Important", 0: "Not Important"}
Churn_map = {1 : "Has Churned", 0: "Has not Churned"}
TFC_df["Timely_Fixes"] = TFC_df["Timely_Fixes"].map(Timely_Fixes_map)
TFC_df["Churn"] = TFC_df["Churn"].map(Churn_map)
mosaic(TFC_df, ["Timely_Fixes", "Churn"])
plt.suptitle("Relationship of Timely Fixes vs Churn");
```

Univariate Exploration Of Timely Fixes Variable



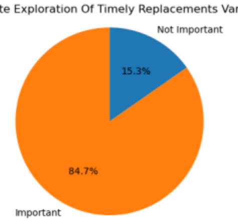
<Figure size 1400x400 with 0 Axes>

Relationship of Timely Fixes vs Churn



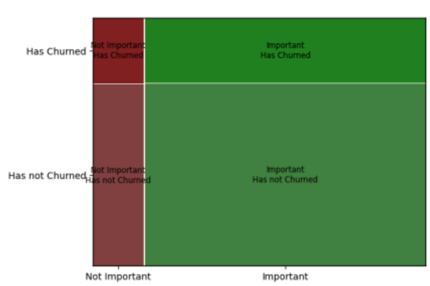

```
# First plot: Univariate exploration of Timely Replacements Variable
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Univariate Exploration Of Timely Replacements Variable")
Timely_Replacements_counts = model_df["Timely_Replacements"].value_counts().sort_index()
plt.pie(Timely_Replacements_counts, labels=["Not Important", "Important"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# Second plot: Bivariate exploration of Timely Replacements Vs Churn
plt.figure(figsize = [14,4])
TRC_df = df[["Timely_Replacements", "Churn"]].copy()
Timely_Replacements_map = {1 : "Important", 0: "Not Important"}
Churn_map = {1 : "Has Churned", 0: "Has not Churned"}
TRC_df["Timely_Replacements"] = TRC_df["Timely_Replacements"].map(Timely_Replacements_map)
TRC_df["Churn"] = TRC_df["Churn"].map(Churn_map)
mosaic(TRC_df, ["Timely_Replacements", "Churn"])
plt.suptitle("Relationship of Timely Replacements vs Churn");
```

Univariate Exploration Of Timely Replacements Variable

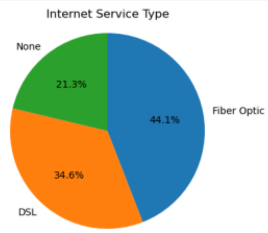


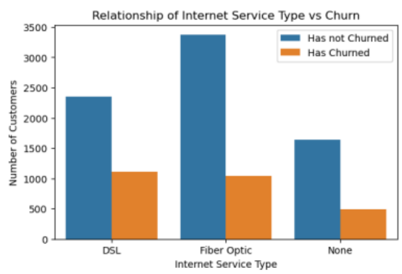
<Figure size 1400x400 with 0 Axes>

Relationship of Timely Replacements vs Churn

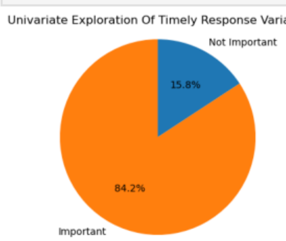


```python
# First plot: Univariate exploration of Internet Service
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Internet Service Type")
Internet_Service_counts = df["Internet_Service"].value_counts()
plt.pie(Internet_Service_counts, labels=Internet_Service_counts.index, autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# Second plot: Bivariate exploration of Internet Service vs Churn
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 2)
plt.title("Relationship of Internet Service Type vs Churn")
sns.countplot(data = df, x="Internet_Service", hue="Churn")
plt.legend(["Has not Churned", "Has Churned"])
plt.xlabel("Internet Service Type")
plt.ylabel("Number of Customers");
```

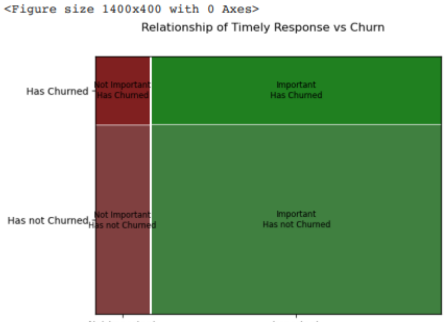




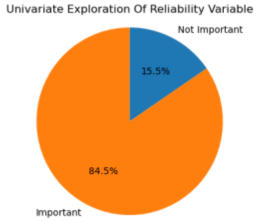

```python
# First plot: Univariate exploration of Timely Response Variable
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Univariate Exploration Of Timely Response Variable")
Timely_Response_counts = model_df["Timely_Response"].value_counts().sort_index()
plt.pie(Timely_Response_counts, labels=["Not Important", "Important"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# Second plot: Bivariate exploration of Timely Response vs Churn
plt.figure(figsize = [14,4])
TRRC_df = df[["Timely_Response", "Churn"]].copy()
Timely_Response_map = {1 : "Important", 0: "Not Important"}
Churn_map = {1 : "Has Churned", 0: "Has not Churned"}
TRRC_df["Timely_Response"] = TRRC_df["Timely_Response"].map(Timely_Response_map)
TRRC_df["Churn"] = TRRC_df["Churn"].map(Churn_map)
mosaic(TRRC_df, ["Timely_Response", "Churn"])
plt.suptitle("Relationship of Timely Response vs Churn");
```



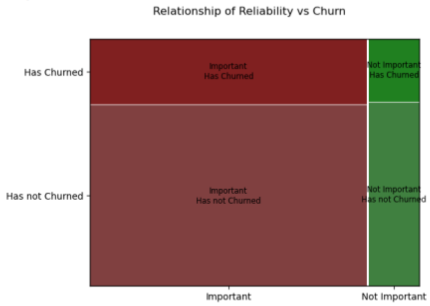
<Figure size 1400x400 with 0 Axes>



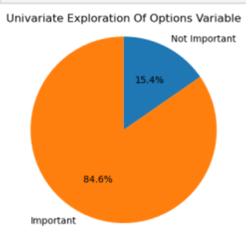
```
# First plot: Univariate exploration of Reliability Variable
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Univariate Exploration Of Reliability Variable")
Reliability_counts = model_df["Reliability"].value_counts().sort_index()
plt.pie(Reliability_counts, labels=["Not Important", "Important"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# Second plot: Bivariate exploration of Reliability vs Churn
plt.figure(figsize = [14,4])
RC_df = df[["Reliability", "Churn"]].copy()
Reliability_map = {1 : "Important", 0: "Not Important"}
Churn_map = {1 : "Has Churned", 0: "Has not Churned"}
RC_df["Reliability"] = RC_df["Reliability"].map(Reliability_map)
RC_df["Churn"] = RC_df["Churn"].map(Churn_map)
mosaic(RC_df, ["Reliability", "Churn"])
plt.suptitle("Relationship of Reliability vs Churn");
```



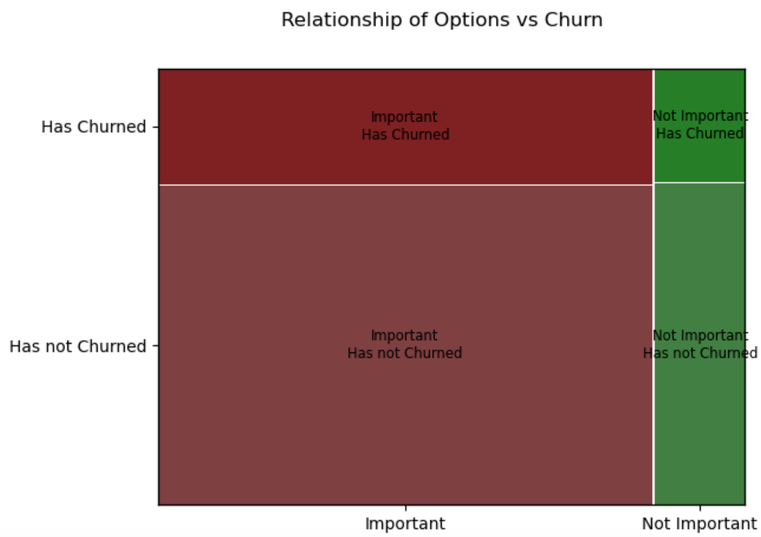

```
<Figure size 1400x400 with 0 Axes>
```



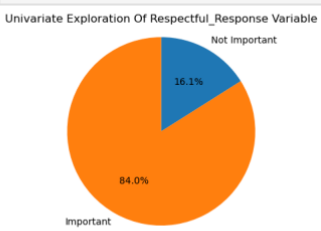

```
# First plot: Univariate exploration of Options Variable
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Univariate Exploration Of Options Variable")
Options_counts = model_df["Options"].value_counts().sort_index()
plt.pie(Options_counts, labels=["Not Important", "Important"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# Second plot: Bivariate exploration of Options vs Churn
plt.figure(figsize = [14,4])
OC_df = df[["Options", "Churn"]].copy()
Options_map = {1 : "Important", 0: "Not Important"}
Churn_map = {1 : "Has Churned", 0: "Has not Churned"}
OC_df["Options"] = OC_df["Options"].map(Options_map)
OC_df["Churn"] = OC_df["Churn"].map(Churn_map)
mosaic(OC_df, ["Options", "Churn"])
plt.suptitle("Relationship of Options vs Churn");
```



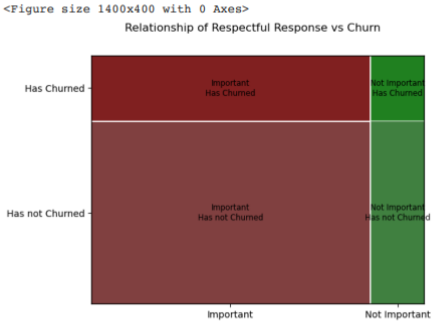

```
<Figure size 1400x400 with 0 Axes>
```



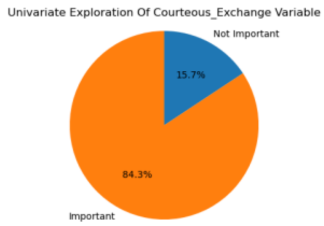

```
# First plot: Univariate exploration of Respectful_Response Variable
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Univariate Exploration Of Respectful_Response Variable")
Respectful_Response_counts = model_df["Respectful_Response"].value_counts().sort_index()
plt.pie(Respectful_Response_counts, labels=["Not Important", "Important"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# Second plot: Bivariate exploration of Respectful Response vs Churn
plt.figure(figsize = [14,4])
RR_df = df[["Respectful_Response", "Churn"]].copy()
Respectful_Response_map = {1 : "Important", 0: "Not Important"}
Churn_map = {1 : "Has Churned", 0: "Has not Churned"}
RR_df["Respectful_Response"] = RR_df["Respectful_Response"].map(Respectful_Response_map)
RR_df["Churn"] = RR_df["Churn"].map(Churn_map)
mosaic(RR_df, ["Respectful_Response", "Churn"])
plt.suptitle("Relationship of Respectful Response vs Churn");
```



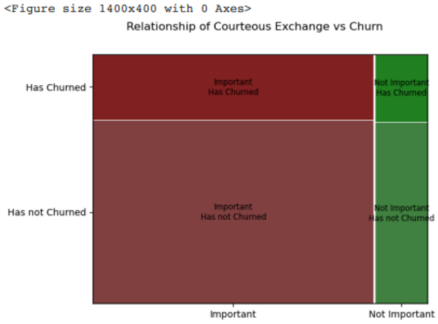

```
<Figure size 1400x400 with 0 Axes>
```



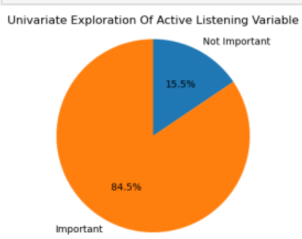
```python
# First plot: Univariate exploration of Courteous_Exchange Variable
plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Univariate Exploration Of Courteous_Exchange Variable")
Courteous_Exchange_counts = model_df["Courteous_Exchange"].value_counts().sort_index()
plt.pie(Courteous_Exchange_counts, labels=["Not Important", "Important"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# Second plot: Bivariate exploration of Courteous Exchange vs Churn
plt.figure(figsize = [14,4])
CEC_df = df[["Courteous_Exchange", "Churn"]].copy()
Courteous_Exchange_map = {1 : "Important", 0: "Not Important"}
Churn_map = {1 : "Has Churned", 0: "Has not Churned"}
CEC_df["Courteous_Exchange"] = CEC_df["Courteous_Exchange"].map(Courteous_Exchange_map)
CEC_df["Churn"] = CEC_df["Churn"].map(Churn_map)
mosaic(CEC_df, ["Courteous_Exchange", "Churn"])
plt.suptitle("Relationship of Courteous Exchange vs Churn");
```



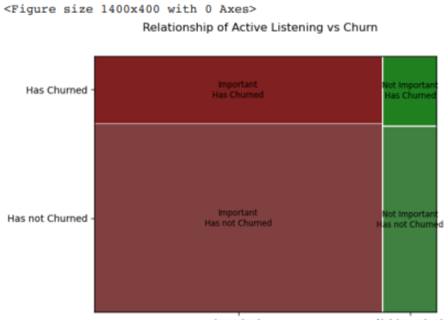

```
<Figure size 1400x400 with 0 Axes>
```




```python
# First plot: Univariate exploration of Active_Listening Variable

plt.figure(figsize = [14,4])
plt.subplot(1, 2, 1)
plt.title("Univariate Exploration Of Active Listening Variable")
Active_Listening_counts = model_df["Active_Listening"].value_counts().sort_index()
plt.pie(Active_Listening_counts, labels=["Not Important", "Important"], autopct='%1.1f%%', startangle=90, counterclock = False)
plt.axis('square');

# Second plot: Bivariate exploration of Active Listening vs Churn
plt.figure(figsize = [14,4])
AL_df = df[["Active_Listening", "Churn"]].copy()
Active_Listening_map = {1 : "Important", 0: "Not Important"}
Churn_map = {1 : "Has Churned", 0: "Has not Churned"}
AL_df["Active_Listening"] = AL_df["Active_Listening"].map(Active_Listening_map)
AL_df["Churn"] = AL_df["Churn"].map(Churn_map)
mosaic(AL_df, ["Active_Listening", "Churn"])
plt.suptitle("Relationship of Active Listening vs Churn");
```




```
<Figure size 1400x400 with 0 Axes>
```



C4. Describe your data transformation goals

When looking at the data frame initially, I noticed that a lot of variables had datatypes of either object or float64, with a few being int64. Transforming all the data types to better suit their data values was necessary prior to completing the linear regression model. Starting with the numeric columns, the data type was converted to “int.” Then, the two variables with categorical data points “contract” and “internet service” were transformed to data type category. Then I was able to map “yes” values as 1 and “no” values as 0 for variables "Multiple", "Online_Security", "Online_Backup" and "Churn".

Values within the survey columns needed to be transformed as well. Having responses being 1 through 8 seems excessive and confusing. To know that “1” is equivalent to “important” and “8” is equivalent to “not important” does not inform me of the values in which a response of 2,3,4,5,6 and 7 represent. In addition, it is likely that values near 1 such as 2, 3 and 4 are closer to “important” than the opposite. The same goes for values 5, 6 and 7 likely to be related to “not important.” As such, I mapped all the values within each of the survey questions to show those responses of 1-4 are given a value of “1”, or important. Survey responses of 5-8 are given a value of “0”, or not important. Grouping the 4 low values and the 4 high values made understanding the data much more manageable.

Since almost all of the variables used for this analysis were now numeric in nature, the next necessary step was to create dummy variables for “contract” and “internet_service.” This step was completed using scripts previously used in assessment 1 with the help of information from Unifying Data Science. (Eubaok, 2022) Then I created a new data frame with all the other variables I needed for the model and concatenated the dummy variables to the new data frame. “Creating these dummy columns, otherwise known as one hot encoding, allows the applicable data points to be accounted for in a numeric sense by also using 1 or 0 to identify the specific categorical value applicable to each customer. One value from each column is omitted, and the remaining variables will have a value of 1 if they are applicable to the customer. If all remaining variables have a value of 0, then the omitted value is applicable.” (Churchill, 2023) The data transformation steps can be seen below:

```
# Convert numeric values to int
df["Income"] = df["Income"].astype(int);
df["Monthly_Charge"] = df["Monthly_Charge"].astype(int);
df["Equipment_failure"] = df["Equipment_failure"].astype(int);
df["Outages"] = df["Outages"].astype(int);
df["Bandwidth_Usage"] = df["Bandwidth_Usage"].astype(int);

# Convert Contract column
df["Contract"] = df["Contract"].astype("category");

# Convert Internet_Service column
df["Internet_Service"] = df["Internet_Service"].astype("category");

# Change all yes/no values to 1 or 0 by mapping
Cat_map = {'Yes': 1, 'No': 0};

# Apply the mapping to applicable columns
convert = ["Multiple" , "Online_Security" , "Online_Backup", "Churn"]
df[convert] = df[convert].replace(Cat_map);


# Create a mapping of survey scores to their ordinal values
Survey_mapping = {
    1: '1',
    2: '1',
    3: '1',
    4: '1',
    5: '0',
    6: '0',
    7: '0',
    8: '0'
}

# Apply the mapping to create new binary columns for each survey score
survey_columns = ["Timely_Response", "Timely_Fixes", "Timely_Replacements",
                  "Reliability", "Options", "Respectful_Response",
                  "Courteous_Exchange", "Active_Listening"]

for column in survey_columns:
    df[column] = df[column].map(Survey_mapping).astype(int)

# Create dummy variables for applicable columns and new dataframe
Dummy_Variables = ["Internet_Service", "Contract"]
dummy_dfs = []

for column in Dummy_Variables:
    dummy_df = pd.get_dummies(data=df[column], prefix=column, drop_first=True)
    dummy_dfs.append(dummy_df)

# Create new dataframe with variables used in analysis
model_df = df[[ "Bandwidth_Usage" , "Outages" , "Equipment_failure" , "Monthly_Charge" , "Income" ,
      "Multiple", "Online_Security", "Online_Backup", "Churn" , "Timely_Response", "Timely_Fixes", "Timely_Replacements",
               "Reliability", "Options", "Respectful_Response",
           "Courteous_Exchange", "Active_Listening"]]

# Concatenate the dummy variables/df with the original df
model_df = pd.concat([model_df] + dummy_dfs, axis=1)
```

C5. Provide the prepared data set as a CSV file

Attached to the submission of this assessment the CSV file model_df.csv can be found.

Part IV: Model Comparison and Analysis

D1. Construct an initial logistic regression model

The initial linear regression model was created along with the AIC (Akaike information criterion) for later analysis steps. (Zach, 2021) Both the model and AIC value can be seen here:

```
# Create initial regressions results
y = model_df.Churn
X = model_df[["Income" , "Monthly_Charge" , "Bandwidth_Usage" , "Multiple" , "Online_Security" , "Online_Backup" ,
              "Outages" , "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
"Internet_Service_None" , "Internet_Service_Fiber Optic" , "Timely_Response" ,
"Timely_Fixes" , "Timely_Replacements" ,  "Reliability" ,
"Options" , "Respectful_Response" , "Courteous_Exchange" , "Active_Listening"]].assign(const=1)
l_model=sm.Logit(y,X)
result=l_model.fit()
print(result.summary())

#add constant to predictor variables
x = sm.add_constant(X)

#fit regression model
red_model = sm.OLS(y, X).fit()

#view AIC of model
print(red_model.aic)
```

```
Optimization terminated successfully.
         Current function value: 0.232564
         Iterations 9
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9979
Method:                           MLE   Df Model:                           20
Date:                Tue, 29 Aug 2023   Pseudo R-squ.:                  0.5978
Time:                        21:28:44   Log-Likelihood:                -2325.6
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Income                        3.011e-07   1.31e-06      0.229      0.819   -2.27e-06    2.88e-06
Monthly_Charge                   0.0646      0.002     38.759      0.000       0.061       0.068
Bandwidth_Usage                 -0.0013   3.35e-05    -39.693      0.000      -0.001      -0.001
Multiple                        -0.4648      0.081     -5.769      0.000      -0.623      -0.307
Online_Security                 -0.2181      0.078     -2.809      0.005      -0.370      -0.066
Online_Backup                   -0.5848      0.079     -7.446      0.000      -0.739      -0.431
Outages                          0.0027      0.012      0.216      0.829      -0.022       0.027
Equipment_failure               -0.0315      0.059     -0.537      0.591      -0.146       0.083
Contract_One year               -3.1820      0.122    -25.983      0.000      -3.422      -2.942
Contract_Two Year               -3.2427      0.120    -27.117      0.000      -3.477      -3.008
Internet_Service_None           -1.0769      0.106    -10.154      0.000      -1.285      -0.869
Internet_Service_Fiber Optic    -3.0820      0.108    -28.511      0.000      -3.294      -2.870
Timely_Response                  0.0023      0.117      0.020      0.984      -0.227       0.232
Timely_Fixes                     0.0814      0.116      0.702      0.483      -0.146       0.309
Timely_Replacements             -0.1017      0.114     -0.890      0.374      -0.326       0.122
Reliability                      0.0306      0.105      0.293      0.770      -0.174       0.235
Options                         -0.1098      0.108     -1.019      0.308      -0.321       0.101
Respectful_Response              0.1464      0.109      1.346      0.178      -0.067       0.359
Courteous_Exchange              -0.1149      0.108     -1.067      0.286      -0.326       0.096
Active_Listening                -0.0397      0.105     -0.379      0.705      -0.245       0.166
const                           -5.8875      0.309    -19.023      0.000      -6.494      -5.281
================================================================================================
5501.977024331241
```

D2. Justify a statistically based feature selection procedure

For the previous regression model, I had used backward stepwise elimination and checked the VIF (variance inflation factor) to rule out statistically insignificant variables and high multicollinearity. (Zach, 2020) Given the number of variables in this analysis I felt that this feature selection procedure would be beneficial for the linear regression model as well. Picking out each variable individually with high multicollinearity one by one based off the highest value first and moving down the list until all variables with a VIF value of 5 or greater takes care of any multicollinearity. At that point, the model is reduced and ready to be assessed for P-values for the stepwise backward elimination. Any variables with a p-value of less than or equal to 0.05 have no statistical significance and were removed to create the final reduced model with only statistically significant variables.

The following columns were removed during these processes:

- Monthly_Charge for VIF: 19.186020
- Outages for VIF: 9.033140
- Timely_Response for VIF: 8.846529
- Timely_Replacements for VIF: 7.239841

- Respectful_Response for VIF: 6.646223
- Courteous_Exchange for VIF: 6.137363
- Active_Listening for VIF: 5.812230
- Timely_Fixes for VIF: 5.111696
- Reliability for p-value: 0.847
- Online_Security for p-value: 0.813
- Income for p-value: 0.687
- Options for p-value: 0.414
- Equipment_failure for p-value: 0.396

D3. Reduced logistic regression model and the output for each feature selection procedure

Eight variables were removed due to high multicollinearity and five were removed for their p-values. All other variables had a VIF value of < 5 and p-value of < 0.05 This process can be seen here:

```
# Check for high multicollinearity (VIF > 5) among variables
X = model_df[["Income" , "Monthly_Charge" , "Bandwidth_Usage" , "Multiple" , "Online_Security" , "Online_Backup" ,
              "Outages" , "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic" , "Timely_Response" ,
        "Timely_Fixes" , "Timely_Replacements" ,  "Reliability" ,
        "Options" , "Respectful_Response" , "Courteous_Exchange" , "Active_Listening"]]
vif_df = pd.DataFrame()
vif_df["Variable"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                        Variable        VIF
0                         Income   2.895511
1                 Monthly_Charge  19.186020
2                Bandwidth_Usage   3.365857
3                       Multiple   2.224782
4                Online_Security   1.556622
5                  Online_Backup   1.995495
6                        Outages   9.567035
7              Equipment_failure   1.387620
8              Contract_One year   1.378658
9              Contract_Two Year   1.438461
10         Internet_Service_None   1.584750
11  Internet_Service_Fiber Optic   2.423338
12               Timely_Response   8.867568
13                  Timely_Fixes   8.125815
14           Timely_Replacements   7.904159
15                   Reliability   6.161140
16                       Options   5.786415
17           Respectful_Response   7.017446
18            Courteous_Exchange   6.801167
19              Active_Listening   6.616474
```

```
# Remove Variable "Monthly_Charge" due to VIF value, re-check for high multicollinearity
X = model_df[["Income" , "Bandwidth_Usage" , "Multiple" , "Online_Security" , "Online_Backup" ,
              "Outages" , "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic" , "Timely_Response" ,
        "Timely_Fixes" , "Timely_Replacements" ,  "Reliability" ,
        "Options" , "Respectful_Response" , "Courteous_Exchange" , "Active_Listening"]]
vif_df = pd.DataFrame()
vif_df["Variable"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                        Variable       VIF
0                         Income  2.863991
1                Bandwidth_Usage  3.266075
2                       Multiple  1.833019
3                Online_Security  1.543876
4                  Online_Backup  1.799638
5                        Outages  9.033140
6              Equipment_failure  1.386276
7              Contract_One year  1.373897
8              Contract_Two Year  1.435405
9          Internet_Service_None  1.583216
10  Internet_Service_Fiber Optic  2.199079
11               Timely_Response  8.857365
12                  Timely_Fixes  8.105404
13           Timely_Replacements  7.855134
14                   Reliability  6.003845
15                       Options  5.501106
16           Respectful_Response  6.989842
17            Courteous_Exchange  6.733788
18              Active_Listening  6.550022
```

```
# Remove Variable "Outages" due to VIF value, re-check for high multicollinearity
X = model_df[["Income" , "Bandwidth_Usage" , "Multiple" , "Online_Security" , "Online_Backup" ,
              "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic" , "Timely_Response" ,
        "Timely_Fixes" , "Timely_Replacements" ,  "Reliability" ,
        "Options" , "Respectful_Response" , "Courteous_Exchange" , "Active_Listening"]]
vif_df = pd.DataFrame()
vif_df["Variable"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                        Variable       VIF
0                         Income  2.839638
1                Bandwidth_Usage  3.216464
2                       Multiple  1.824240
3                Online_Security  1.540497
4                  Online_Backup  1.793782
5              Equipment_failure  1.384216
6              Contract_One year  1.370713
7              Contract_Two Year  1.429236
8          Internet_Service_None  1.568680
9   Internet_Service_Fiber Optic  2.172701
10               Timely_Response  8.846529
11                  Timely_Fixes  8.072952
12           Timely_Replacements  7.803313
13                   Reliability  5.847306
14                       Options  5.150194
15           Respectful_Response  6.937704
16            Courteous_Exchange  6.648857
17              Active_Listening  6.489027
```

```
# Remove Variable "Timely_Response" due to VIF value, re-check for high multicollinearity
X = model_df[["Income" , "Bandwidth_Usage" , "Multiple" , "Online_Security" , "Online_Backup" ,
            "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic" ,
        "Timely_Fixes" , "Timely_Replacements" ,  "Reliability" ,
        "Options" , "Respectful_Response" , "Courteous_Exchange" , "Active_Listening"]]
vif_df = pd.DataFrame()
vif_df["Variable"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                        Variable       VIF
0                         Income  2.839485
1                Bandwidth_Usage  3.212220
2                       Multiple  1.824219
3                Online_Security  1.540020
4                  Online_Backup  1.793665
5              Equipment_failure  1.384091
6              Contract_One year  1.370434
7              Contract_Two Year  1.429070
8          Internet_Service_None  1.566800
9   Internet_Service_Fiber Optic  2.172690
10                  Timely_Fixes  7.094515
11           Timely_Replacements  7.239841
12                   Reliability  5.846096
13                       Options  5.116974
14           Respectful_Response  6.808913
15            Courteous_Exchange  6.561810
16              Active_Listening  6.440214
```

```
# Remove Variable "Timely_Replacements" due to VIF value, re-check for high multicollinearity
X = model_df[["Income" , "Bandwidth_Usage" , "Multiple" , "Online_Security" , "Online_Backup" ,
            "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic" ,
        "Timely_Fixes" , "Reliability" ,
        "Options" , "Respectful_Response" , "Courteous_Exchange" , "Active_Listening"]]
vif_df = pd.DataFrame()
vif_df["Variable"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                        Variable       VIF
0                         Income  2.830748
1                Bandwidth_Usage  3.202154
2                       Multiple  1.823448
3                Online_Security  1.539603
4                  Online_Backup  1.790741
5              Equipment_failure  1.382094
6              Contract_One year  1.370060
7              Contract_Two Year  1.427612
8          Internet_Service_None  1.566029
9   Internet_Service_Fiber Optic  2.166915
10                  Timely_Fixes  6.336905
11                   Reliability  5.832094
12                       Options  5.025520
13           Respectful_Response  6.646223
14            Courteous_Exchange  6.480772
15              Active_Listening  6.342818
```

```
# Remove Variable "Respectful_Response" due to VIF value, re-check for high multicollinearity
X = model_df[["Income" , "Bandwidth_Usage" , "Multiple" , "Online_Security" , "Online_Backup" ,
              "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic" ,
        "Timely_Fixes" , "Reliability" ,
        "Options" , "Courteous_Exchange" , "Active_Listening"]]
vif_df = pd.DataFrame()
vif_df["Variable"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                        Variable       VIF
0                         Income  2.821743
1                Bandwidth_Usage  3.188792
2                       Multiple  1.822211
3                Online_Security  1.538041
4                  Online_Backup  1.789205
5              Equipment_failure  1.381489
6              Contract_One year  1.369877
7              Contract_Two Year  1.426394
8          Internet_Service_None  1.560774
9   Internet_Service_Fiber Optic  2.163231
10                  Timely_Fixes  6.024222
11                   Reliability  5.625622
12                       Options  5.018493
13            Courteous_Exchange  6.137363
14              Active_Listening  6.131239
```

```
# Remove Variable "Courteous_Exchange" due to VIF value, re-check for high multicollinearity
X = model_df[["Income" , "Bandwidth_Usage" , "Multiple" , "Online_Security" , "Online_Backup" ,
              "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic" ,
        "Timely_Fixes" , "Reliability" ,
        "Options" , "Active_Listening"]]
vif_df = pd.DataFrame()
vif_df["Variable"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                        Variable       VIF
0                         Income  2.810056
1                Bandwidth_Usage  3.163428
2                       Multiple  1.820025
3                Online_Security  1.533732
4                  Online_Backup  1.786595
5              Equipment_failure  1.379885
6              Contract_One year  1.368670
7              Contract_Two Year  1.424830
8          Internet_Service_None  1.555777
9   Internet_Service_Fiber Optic  2.152233
10                  Timely_Fixes  5.603701
11                   Reliability  5.366301
12                       Options  4.990573
13              Active_Listening  5.812230
```

```
# Remove Variable "Active_Listening" due to VIF value, re-check for high multicollinearity
X = model_df[["Income" , "Bandwidth_Usage" , "Multiple" , "Online_Security" , "Online_Backup" ,
              "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic" ,
        "Timely_Fixes" , "Reliability" ,
        "Options"]]
vif_df = pd.DataFrame()
vif_df["Variable"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                        Variable       VIF
0                         Income  2.787425
1                Bandwidth_Usage  3.116284
2                       Multiple  1.814172
3                Online_Security  1.530891
4                  Online_Backup  1.779628
5              Equipment_failure  1.378053
6              Contract_One year  1.364231
7              Contract_Two Year  1.419761
8          Internet_Service_None  1.538276
9   Internet_Service_Fiber Optic  2.129476
10                  Timely_Fixes  5.111696
11                   Reliability  4.995837
12                       Options  4.914814
```

```
# Remove Variable "Timely_Fixes" due to VIF value, re-check for high multicollinearity
X = model_df[["Income" , "Bandwidth_Usage" , "Multiple" , "Online_Security" , "Online_Backup" ,
              "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic" ,
         "Reliability" ,
        "Options"]]
vif_df = pd.DataFrame()
vif_df["Variable"] = X.columns

vif_df["VIF"] = [variance_inflation_factor(X.values, i)
for i in range(len(X.columns))]

print(vif_df)
```

```
                        Variable       VIF
0                         Income  2.746179
1                Bandwidth_Usage  3.057655
2                       Multiple  1.800600
3                Online_Security  1.526195
4                  Online_Backup  1.772383
5              Equipment_failure  1.372860
6              Contract_One year  1.356560
7              Contract_Two Year  1.413212
8          Internet_Service_None  1.518118
9   Internet_Service_Fiber Optic  2.091885
10                   Reliability  4.625861
11                       Options  4.535348
```

```
# Begin backward elimination by checking regression results for P-values > 0.05
y = model_df.Churn
X = model_df[["Income" , "Bandwidth_Usage" , "Multiple" , "Online_Security" , "Online_Backup" ,
                "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
          "Internet_Service_None" , "Internet_Service_Fiber Optic" ,
           "Reliability" , "Options"]].assign(const=1)
l_model01=sm.Logit(y,X)
result=l_model01.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.392833
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9987
Method:                           MLE   Df Model:                           12
Date:                Tue, 29 Aug 2023   Pseudo R-squ.:                  0.3206
Time:                        21:25:13   Log-Likelihood:                -3928.3
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Income                        3.961e-07      1e-06      0.396      0.692   -1.56e-06    2.36e-06
Bandwidth_Usage                 -0.0007   1.71e-05    -41.151      0.000      -0.001      -0.001
Multiple                         0.9670      0.058     16.692      0.000       0.853       1.080
Online_Security                 -0.0138      0.059     -0.235      0.814      -0.129       0.102
Online_Backup                    0.4878      0.057      8.557      0.000       0.376       0.600
Equipment_failure               -0.0384      0.045     -0.858      0.391      -0.126       0.049
Contract_One year               -1.7430      0.079    -21.970      0.000      -1.899      -1.588
Contract_Two Year               -1.8824      0.078    -24.143      0.000      -2.035      -1.730
Internet_Service_None           -1.0336      0.080    -12.992      0.000      -1.189      -0.878
Internet_Service_Fiber Optic    -1.0190      0.065    -15.562      0.000      -1.147      -0.891
Reliability                      0.0153      0.079      0.193      0.847      -0.140       0.170
Options                         -0.0623      0.079     -0.785      0.432      -0.218       0.093
const                            1.5757      0.137     11.498      0.000       1.307       1.844
================================================================================================
```

```
# Continue backward elimination after removing "Reliability" for p-value: 0.847
y = model_df.Churn
X = model_df[["Income" , "Bandwidth_Usage" , "Multiple" , "Online_Security" , "Online_Backup" ,
                "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
          "Internet_Service_None" , "Internet_Service_Fiber Optic" , "Options"]].assign(const=1)
l_model02=sm.Logit(y,X)
result=l_model02.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.392835
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9988
Method:                           MLE   Df Model:                           11
Date:                Tue, 29 Aug 2023   Pseudo R-squ.:                  0.3206
Time:                        21:25:14   Log-Likelihood:                -3928.4
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Income                        3.985e-07      1e-06      0.399      0.690   -1.56e-06    2.36e-06
Bandwidth_Usage                 -0.0007   1.71e-05    -41.152      0.000      -0.001      -0.001
Multiple                         0.9673      0.058     16.703      0.000       0.854       1.081
Online_Security                 -0.0140      0.059     -0.237      0.813      -0.130       0.102
Online_Backup                    0.4880      0.057      8.561      0.000       0.376       0.600
Equipment_failure               -0.0383      0.045     -0.856      0.392      -0.126       0.049
Contract_One year               -1.7430      0.079    -21.970      0.000      -1.899      -1.588
Contract_Two Year               -1.8824      0.078    -24.143      0.000      -2.035      -1.730
Internet_Service_None           -1.0337      0.080    -12.995      0.000      -1.190      -0.878
Internet_Service_Fiber Optic    -1.0191      0.065    -15.566      0.000      -1.147      -0.891
Options                         -0.0645      0.079     -0.821      0.412      -0.218       0.089
const                            1.5903      0.114     13.920      0.000       1.366       1.814
================================================================================================
```

```
# Continue backward elimination after removing "Online_Security" for p-value: 0.813
y = model_df.Churn
X = model_df[["Income" , "Bandwidth_Usage" , "Multiple" , "Online_Backup" ,
              "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic" , "Options"]].assign(const=1)
l_model03=sm.Logit(y,X)
result=l_model03.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.392838
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9989
Method:                           MLE   Df Model:                           10
Date:                Tue, 29 Aug 2023   Pseudo R-squ.:                  0.3206
Time:                        21:25:15   Log-Likelihood:                -3928.4
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Income                        4.033e-07      1e-06      0.403      0.687   -1.56e-06    2.36e-06
Bandwidth_Usage                 -0.0007   1.71e-05    -41.155      0.000      -0.001      -0.001
Multiple                         0.9672      0.058     16.703      0.000       0.854       1.081
Online_Backup                    0.4878      0.057      8.559      0.000       0.376       0.600
Equipment_failure               -0.0380      0.045     -0.851      0.395      -0.126       0.050
Contract_One year               -1.7432      0.079    -21.972      0.000      -1.899      -1.588
Contract_Two Year               -1.8827      0.078    -24.150      0.000      -2.035      -1.730
Internet_Service_None           -1.0335      0.080    -12.994      0.000      -1.189      -0.878
Internet_Service_Fiber Optic    -1.0191      0.065    -15.566      0.000      -1.147      -0.891
Options                         -0.0643      0.079     -0.819      0.413      -0.218       0.090
const                            1.5851      0.112     14.140      0.000       1.365       1.805
================================================================================================
```

```
# Continue backward elimination after removing "Income" for p-value: 0.687
y = model_df.Churn
X = model_df[["Bandwidth_Usage" , "Multiple" , "Online_Backup" ,
              "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic" , "Options"]].assign(const=1)
l_model04=sm.Logit(y,X)
result=l_model04.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.392846
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9990
Method:                           MLE   Df Model:                            9
Date:                Tue, 29 Aug 2023   Pseudo R-squ.:                  0.3206
Time:                        21:25:16   Log-Likelihood:                -3928.5
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Bandwidth_Usage                 -0.0007   1.71e-05    -41.155      0.000      -0.001      -0.001
Multiple                         0.9672      0.058     16.702      0.000       0.854       1.081
Online_Backup                    0.4877      0.057      8.558      0.000       0.376       0.599
Equipment_failure               -0.0380      0.045     -0.850      0.396      -0.126       0.050
Contract_One year               -1.7430      0.079    -21.972      0.000      -1.899      -1.588
Contract_Two Year               -1.8831      0.078    -24.157      0.000      -2.036      -1.730
Internet_Service_None           -1.0341      0.080    -13.003      0.000      -1.190      -0.878
Internet_Service_Fiber Optic    -1.0197      0.065    -15.579      0.000      -1.148      -0.891
Options                         -0.0642      0.079     -0.817      0.414      -0.218       0.090
const                            1.6016      0.104     15.340      0.000       1.397       1.806
================================================================================================
```

```
# Continue backward elimination after removing "Options" for p-value: 0.414
y = model_df.Churn
X = model_df[["Bandwidth_Usage" , "Multiple" , "Online_Backup" ,
              "Equipment_failure" , "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic"]].assign(const=1)
l_model05=sm.Logit(y,X)
result=l_model05.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.392879
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9991
Method:                           MLE   Df Model:                            8
Date:                Tue, 29 Aug 2023   Pseudo R-squ.:                  0.3205
Time:                        21:25:17   Log-Likelihood:                -3928.8
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Bandwidth_Usage                 -0.0007   1.71e-05    -41.151      0.000      -0.001      -0.001
Multiple                         0.9672      0.058     16.703      0.000       0.854       1.081
Online_Backup                    0.4885      0.057      8.572      0.000       0.377       0.600
Equipment_failure               -0.0380      0.045     -0.849      0.396      -0.126       0.050
Contract_One year               -1.7428      0.079    -21.972      0.000      -1.898      -1.587
Contract_Two Year               -1.8819      0.078    -24.149      0.000      -2.035      -1.729
Internet_Service_None           -1.0336      0.080    -12.998      0.000      -1.189      -0.878
Internet_Service_Fiber Optic    -1.0202      0.065    -15.587      0.000      -1.148      -0.892
const                            1.5463      0.079     19.495      0.000       1.391       1.702
================================================================================================
```

```
# Continue backward elimination after removing "Equipment_failure" for p-value: 0.396
y = model_df.Churn
X = model_df[["Bandwidth_Usage" , "Multiple" , "Online_Backup" ,
              "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic"]].assign(const=1)
l_model06=sm.Logit(y,X)
result=l_model06.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.392916
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9992
Method:                           MLE   Df Model:                            7
Date:                Tue, 29 Aug 2023   Pseudo R-squ.:                  0.3205
Time:                        21:25:19   Log-Likelihood:                -3929.2
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Bandwidth_Usage                 -0.0007   1.71e-05    -41.161      0.000      -0.001      -0.001
Multiple                         0.9672      0.058     16.704      0.000       0.854       1.081
Online_Backup                    0.4893      0.057      8.587      0.000       0.378       0.601
Contract_One year               -1.7436      0.079    -21.984      0.000      -1.899      -1.588
Contract_Two Year               -1.8820      0.078    -24.151      0.000      -2.035      -1.729
Internet_Service_None           -1.0336      0.080    -12.999      0.000      -1.189      -0.878
Internet_Service_Fiber Optic    -1.0202      0.065    -15.588      0.000      -1.148      -0.892
const                            1.5315      0.077     19.800      0.000       1.380       1.683
================================================================================================
```

```
# Create the Reduced model
y = model_df.Churn
X = model_df[["Bandwidth_Usage" , "Multiple" , "Online_Backup" ,
              "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic"]].assign(const=1)
red_model=sm.Logit(y,X)
red_result=red_model.fit()
print(red_result.summary())


#add constant to predictor variables
x = sm.add_constant(X)

#fit regression model
red_model = sm.OLS(y, X).fit()

#view AIC of model
print(red_model.aic)
```

```
Optimization terminated successfully.
         Current function value: 0.392916
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9992
Method:                           MLE   Df Model:                            7
Date:                Tue, 29 Aug 2023   Pseudo R-squ.:                  0.3205
Time:                        21:25:20   Log-Likelihood:                -3929.2
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Bandwidth_Usage                 -0.0007   1.71e-05    -41.161      0.000      -0.001      -0.001
Multiple                         0.9672      0.058     16.704      0.000       0.854       1.081
Online_Backup                    0.4893      0.057      8.587      0.000       0.378       0.601
Contract_One year               -1.7436      0.079    -21.984      0.000      -1.899      -1.588
Contract_Two Year               -1.8820      0.078    -24.151      0.000      -2.035      -1.729
Internet_Service_None           -1.0336      0.080    -12.999      0.000      -1.189      -0.878
Internet_Service_Fiber Optic    -1.0202      0.065    -15.588      0.000      -1.148      -0.892
const                            1.5315      0.077     19.800      0.000       1.380       1.683
================================================================================================
8366.229530974775
```

E1. Explain your data analysis process

Overall, there were 21 variables in the initial model. In the reduced model there are only 8. Therefore, the results found in the reduced model, as well as its relevance, were certainly likely to change. To compare the models, I looked at the AIC (Akaike's Information Criteria). In addition, I compared the coefficients and Pseudo R squared values in each model.

The AIC "penalizes the errors made in case a new variable is added to the regression equation. The model with the lowest AIC offers the best fit." (Middleton, 2023) Comparing the changes in coefficient values and their odds ratios shows the difference in how much the likelihood of churn is for a change of one unit in each independent variable. Lastly, the Pseudo R squared is another way to determine which model has a better fit by looking for the model with the higher value between 0-1.

After evaluating these measures and the coefficients, the next step was to complete and review the confusion matrix and accuracy calculations for the reduced model and initial model to compare.

E2. Provide the output and *all* calculations of the analysis you performed

```
# Create the Reduced model
y = model_df.Churn
X = model_df[["Bandwidth_Usage" , "Multiple" , "Online_Backup" ,
              "Contract_One year" , "Contract_Two Year" ,
        "Internet_Service_None" , "Internet_Service_Fiber Optic"]].assign(const=1)
red_model=sm.Logit(y,X)
red_result=red_model.fit()
print(red_result.summary())


#add constant to predictor variables
x = sm.add_constant(X)

#fit regression model
red_model = sm.OLS(y, X).fit()

#view AIC of model
print(red_model.aic)
```

```
Optimization terminated successfully.
         Current function value: 0.392916
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9992
Method:                           MLE   Df Model:                            7
Date:                Tue, 29 Aug 2023   Pseudo R-squ.:                  0.3205
Time:                        21:25:20   Log-Likelihood:                -3929.2
converged:                       True   LL-Null:                       -5782.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Bandwidth_Usage                 -0.0007   1.71e-05    -41.161      0.000      -0.001      -0.001
Multiple                         0.9672      0.058     16.704      0.000       0.854       1.081
Online_Backup                    0.4893      0.057      8.587      0.000       0.378       0.601
Contract_One year               -1.7436      0.079    -21.984      0.000      -1.899      -1.588
Contract_Two Year               -1.8820      0.078    -24.151      0.000      -2.035      -1.729
Internet_Service_None           -1.0336      0.080    -12.999      0.000      -1.189      -0.878
Internet_Service_Fiber Optic    -1.0202      0.065    -15.588      0.000      -1.148      -0.892
const                            1.5315      0.077     19.800      0.000       1.380       1.683
================================================================================================
8366.229530974775
```

In the image above the reduced model shows the remaining variables along with their coefficient and p-values, these are described in the table below. The regression results also show the Pseudo R squared: 0.3205, log-likelihood: -3929.2, LL-Null: -5782.2 and LLR p-value:0.000. The remaining numeric value shown in the image is the AIC value of 8366.2295.

Column Name	Coefficient	P-value
Bandwidth usage	-0.0007	0.000
Multiple	0.9672	0.000
Online backup	0.4893	0.000
One year contract	-1.7436	0.000
Two-year contract	-1.8820	0.000
No internet services	-1.0336	0.000
Fiber optic internet service	-1.0202	0.000
Churn (dependent variable)	1.5315	0.000

Information needed to understand, create, and evaluate the confusion matrix and accuracy calculation was obtained from Statology. (Zach, 2021) The results can be seen in the image here:

```
# Create the confusion matrix for reduced model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
C_matrix = confusion_matrix(y_test, y_pred)
print('Accuracy of logistic regression: {:.2f}'.format(logreg.score(X_test, y_test)))
print("Confusion Matrix:")
print(C_matrix)

# Find accuracy, recision, recall and F1 score for reduced model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy :", accuracy)
precision = precision_score(y_test, y_pred)
print("Precision :", precision)
recall = recall_score(y_test, y_pred)
print("Recall :", recall)
F1_score = f1_score(y_test, y_pred)
print("F1-score :", F1_score)
```

```
Accuracy of logistic regression: 0.81
Confusion Matrix:
[[1301  155]
 [ 222  322]]
Accuracy : 0.8115
Precision : 0.6750524109014675
Recall : 0.5919117647058824
F1-score : 0.6307541625857004
```

In the confusion matrix the values 1301, 155, 222 and 322 are seen. These indicate that the logistic regression model has made the following predictions:

- 1,301 instances correctly classified as negative.
- 155 instances incorrectly classified as positive (false positives)
- 222 instances incorrectly classified as negative (false negatives)
- 322 instances correctly classified as positive.

Below the confusion matrix are the values for the model's accuracy, precision, recall and f1-score. The accuracy value indicates that the model correctly predicted 81% of the total instances. The precision value shows that of all positives predicted, 67.5% of them are truly positive. Recall, otherwise known as sensitivity, measures the accuracy of predicting positives by comparing true positives and false negatives. In this regard, the model is accurate 59.19% of the time. F1-score is the "harmonic mean" of precision and recall. It evaluates false positives and false negatives and is used for imbalanced datasets. The higher the F1 score, the better the balance is between precision and recall. In this case it is 0.6307 (2023, Kumar)

```
# Create the confusion matrix for initial model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
logreg = LogisticRegression(max_iter=1000)
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
C_matrix = confusion_matrix(y_test, y_pred)
print('Accuracy of logistic regression: {:.2f}'.format(logreg.score(X_test, y_test)))
print("Confusion Matrix:")
print(C_matrix)

# Find accuracy, recision, recall and F1 score for initial model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy :", accuracy)
precision = precision_score(y_test, y_pred)
print("Precision :", precision)
recall = recall_score(y_test, y_pred)
print("Recall :", recall)
F1_score = f1_score(y_test, y_pred)
print("F1-score :", F1_score)
```

```
Accuracy of logistic regression: 0.87
Confusion Matrix:
[[1360   96]
 [ 166  378]]
Accuracy : 0.869
Precision : 0.7974683544303798
Recall : 0.6948529411764706
F1-score : 0.7426326129666011
```

In the confusion matrix for the initial model the values 1360, 96, 166 and 378 are seen. These indicate that the logistic regression model has made the following predictions:

- 1,360 instances correctly classified as negative.
- 96 instances incorrectly classified as positive (false positives)
- 166 instances incorrectly classified as negative (false negatives)
- 378 instances correctly classified as positive.

Below the confusion matrix are the values for the model's accuracy, precision, recall and f1-score. The precision value shows that of all positives predicted, 79.7% of them are truly positive. Recall, otherwise known as sensitivity, measures the accuracy of predicting positives by comparing true positives and false negatives. In this regard, the model is accurate 69.485% of the time. F1-score is the "harmonic mean" of precision and recall. It evaluates false positives and false negatives and is used for imbalanced datasets. The higher the F1 score, the better the balance is between precision and recall. In this case it is 0.7426 (2023, Kumar)

E3. Provide an executable error-free copy of the code used

The executable script file associated with this analysis is attached to this submission.

Part V: Data Summary and Implications

F1. Discuss the results

The regression equation for the reduced model:

log(1−P(Churn=1)P(Churn=1)) = 1.5315 + -0.0007 (Bandwidth) + 0.9672 (Multiple) + 0.4893 (Online Backup) -1.7436 (One year contract) -1.8820 (Two-year contract) -1.0336 (No internet service) -1.0202 (Fiber Optic)

In the table below are the remaining independent variables in the logistic regression model and their coefficient values. Assessment of the coefficients was done through instruction from Python For Data Science. (n.d.)

Variable	Coefficient
Bandwidth usage	-0.0007
Multiple	0.9672
Online backup	0.4893
One year contract	-1.7436
Two-year contract	-1.8820
No internet services	-1.0336
Fiber optic internet service	-1.0202

All other factors held constant:

- For each one unit of increase in bandwidth usage the log-odds of churn decrease by 0.0007. Suggesting that customers who utilize more bandwidth in GB per year are slightly less likely to churn.
- Customers with multiple phone lines have log-odds of churn 0.9672 higher than customers without multiple lines. Suggesting that likelihood of churn is significantly higher for customers who have multiple phone lines.
- The log-odds of churn for customers with online backup is 0.4893 higher than customers without it. Suggesting that customers who utilize online backup are more likely to churn.
- Customers with a one-year contract have log-odds of churn 1.7436 lower than customers with other contract lengths. This suggests that customers having a one-year contract significantly reduces the likelihood of churn.
- Customers with a two-year contract have log-odds of churn 1.8820 lower than customers with other contract lengths. This suggests that customers having a two-year contract significantly reduces the likelihood of churn, more so than customers with a one-year contract.
- Customers with no internet service package have log-odds of churn 1.0336 lower than customers with other internet service types. This suggests that customers having no internet service package significantly reduces the likelihood of churn.
- Customers with the fiber optic internet service package have log-odds of churn 1.0202 lower than customers with other internet service types. This suggests that customers having the fiber optic internet service package significantly reduces the likelihood of churn.

Although the model accurately predicted 81% of total instances, the model overall is not reliable. The AIC value for the reduced model is 8366.23, and the Pseudo R squared is 0.3205. When comparing this to the initial model, both values indicate that the initial model is a better fit. The AIC for the initial model is 5501.98 and the Pseudo R squared is 0.5978.

Another problem with the model is that two of the independent variables are related to the length of contract. They both have similar statistical outputs. One-year contract has a coefficient value of -1.7436. A two-year contract has a coefficient value of -1.8820. The company marketing for one of these would contradict the relevance of marketing for the other. Having more information regarding phone service in comparison to no internet service would be more appropriate prior to making changes related to the statistical significance of no internet service in this case. Overall, the reduced model is not statistically and practically insignificant.

The data analysis's greatest limitation is that there is not much data within the dataset. Information for only 10,000 customers is available within one month to about 6 years. The reduced model is statistically and practically insignificant and would not be reliable to inform the business of any changes they could make to better retain customers. Having more data over a longer time would give better, more usable results for the business.

F2. Recommend a course of action

Given the absence of reliability within the residual model, it would be recommended that the business does not move forward with any business decisions based off these statistics. If the presently available dataset is the only data to work on, it would be best to reassess the research question. Making changes to the question or scope of the project may render a reliable model in which the business can utilize.

If there is a chance that more data is available for analysis, I would recommend reassessing the current question with the larger dataset. Preferably with data related to customer information for up to at least 10 years. Getting more detailed information of the exact responses for the survey questions, and not just the details of value 1 and 8, could also be much more beneficial to assess the survey questions' importance to the research question. Overall, more information within and about the data in general would be the best course of action.

Part VI: Demonstration

G. Panopto video

The Panopto video recorded for this assessment can be found in the corresponding folder for this course.

H. List the web sources used to acquire data or segments of third-party code

Churchill, Briana. (2023, Aug 22). *Performance Assessment: Predictive Modeling Task 1.* Assignment for MS Data Analytics Course D208. Western Governors University.

Eubank, N. (2022). *Using and Interpreting Indicator (Dummy) Variables.* Unifying Data Science. <https://www.unifyingdatascience.org/html/interpreting_indicator_vars.html>

Statsmodels.graphics.mosaicplot.mosaic. (n.d.). Retrieved from https://www.statsmodels.org/devel/generated/statsmodels.graphics.mosaicplot.mosaic.html

Zach. (2021, May 20). *How to Calculate AIC of Regression Models in Python.* Statology. https://www.statology.org/aic-in-python/#:~:text=To%20calculate%20the%20AIC%20of,value%20for%20a%20given%20model

Zach. (2021, Sep. 1). *How to Create a Confusion Matrix in Python.* Statology https://www.statology.org/confusion-matrix-python/

I. Acknowledge sources

"Interpret the key results for Fit Binary Logistic Model." Minitab 21 Support. https://support.minitab.com/en-us/minitab/21/help-and-how-to/statistical-modeling/regression/how-to/fit-binary-logistic-model/interpret-the-results/key-results/

Kumar, A. (2023, Mar 17). *Accuracy, Precision, Recall & F1-Score – Python Examples.* https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/

Middleton, K. (2023, July 13). "*Getting Started With D208*" Part II. Western Governors University. Pages 13-14.

"Python For Data Science." Python for Data Science, LLC.

https://www.pythonfordatascience.org/logistic-regression-python/

Zach. (2021, May 19). *How to Interpret an Odds Ratio Less Than 1*. Statology. https://www.statology.org/interpret-odds-ratio-less-than-1/

Zach. (2020, Oct. 13). *The 6 Assumptions of Logistic Regression (With Examples).* Statology. https://www.statology.org/assumptions-of-logistic-regression/