

D207 Mattinson (2nd Submission)

```
#####  
      STUDENT INFORMATION  
#####  
      Student | Mike Mattinson  
      Student ID | 001980761  
      Class | D207 Exploratory Data Analysis  
      Dataset | Churn  
      Submission | 2nd Submission  
      School | WGU  
      Instructor | Dr. Sewell  
      Today is July 31, 2021
```

This is what a code snippet looks like:

```
# imports  
import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd  
import datetime  
from mm_lib import *
```

This is what terminal output looks like:

```
#####  
      DATAFRAME (DF)  
#####  
      Customer_id      City Churn  MonthlyCharge  Tenure  
0      K409198  Point Baker    No      172.455519    6.795513  
1      S120509  West Branch   Yes      242.632554    1.156681  
2      K191035    Yamhill    No      159.947583   15.754144  
3      D90850    Del Mar     No      119.956840   17.087227  
(10000, 50)
```

Note. Changes from 1st submission are identified by vertical line in left margin and dark blue colored font.

Table of Contents

Part A Research Question & Benefits	5
Remove Unwanted Data.....	6
Numerical Data.....	7
Categorical Data.....	8
Part B Analysis, code and output	9
Part C Univariate Statistics.....	14
Descriptive Statistics	14
Count Plot – Churn (Cat)	16
Count Plot – InternetService (Cat).....	17
Scatter Plot – MonthlyCharge (Cont).....	19
Scatter Plot – Income (Cont)	20
Box Plot – Normalized Problem Data (Cat)	21
Box Plot – Normalized Demographic Data (Integer, Cont)	22
Sample of different plots – Income (Cont)	23
Part D Bivariate Statistics.....	24
Joint Plot – Income (Cont) vs MonthlyCharge (Cont)	25
PaymentMethod_bivariate_countplot	29
Tablet_bivariate_countplot.....	30
Gender_bivariate_countplot.....	31
Port_modem_bivariate_countplot.....	32
Techie_bivariate_countplot.....	33
InternetService_bivariate_stacked bar	39
Part E Summary.....	40
Part F Video	41
Part G References.....	42
Part H Acknowledge Sources.....	44
Part I Code	45
Part J Output	55
Part K Video Script.....	62

List of Tables

Table 1 CHI-SQUARE DISTRIBUTION TABLE	12
---	----

List of Figures

Figure 1 Churn Cat_countplot (.PNG)	16
Figure 2 InternetService Cat_countplot (.PNG)	17
Figure 3 MonthlyCharge Cont_scatterplot (.PNG)	19
Figure 4 Income Cont_scatterplot (.PNG)	20
Figure 5 Problem Data_boxplot (.PNG) - not used	21
Figure 6 Demographics_boxplot (.PNG) - not used	22
Figure 7 Boxplot - Income (Cont)	23
Figure 8 Histogram plot - Income (Cont)	23
Figure 9 Scatter plot - Income (Cont)	23
Figure 10 Birvariate - Income vs. MonthlyCharge (.PNG)	25
Figure 11 InternetService_bivariate_countplot	27
Figure 12 TechSupport_bivariate_countplot	28
Figure 13 PaymentMethod_bivariate_countplot	29
Figure 14 Tablet_bivariate_countplot	30
Figure 15 Gender_bivariate_countplot	31
Figure 16 Port_modem_bivariate_countplot	32
Figure 17 Techie_bivariate_countplot	33
Figure 18 MonthlyCharge_bivariate_stacked histogram	35
Figure 19 Bandwidth_GB_Year_bivariate_stacked histogram	36
Figure 20 Gender_bivariate_stacked bar.png	38
Figure 21 InternetService_bivariate_stacked bar.png	39

Part A Research Question & Benefits

Part A1. Is the predictor variable 'MonthlyCharge' dependent of the target variable 'Churn'? Is there confidence (at the 99% significance level) that 'MonthlyCharge' can be used to predict 'Churn'?

Part A2. Stakeholders will benefit from knowing the answer to this question. Future market research can make changes and recommendations regarding 'MonthlyCharge' where customers retain services longer.

Part A3. The company provided a dataset of 10,000 customer records. 'Churn' is the categorical variable to indicate ('Yes' or 'No') whether the customer has terminated service within the last month. This will be our target variable. In addition, there are a number of other categorical and numerical variables that can be analyzed as predictor variables. This analysis will consider the following:

Tenure – Numerical – Number of months the customer has stayed with the provider.

MonthlyCharge – Numerical (our primary predictor for part A1 and the following hypothesis testing to follow. It is the amount charged to the customer monthly.

Bandwidth_GB_Year – Numerical – Average amount of data used, in GB, in a year by the customer.

Outage_sec_perweek – Numerical - Average number of seconds per week of system outage s in the customer's neighborhood

Income – Numerical - : Annual income of customer as reported at time of sign up

Gender – Categorical - Customer self identification as male, female, or nonbinary

InternetService – Categorical - Customer's internet service provider (DSL, fiber optic, None)

Remove Unwanted Data

Some of the original data was removed:

```
# remove unwanted columns
unwanted_columns = ["UID", "Interaction", "Lat", "Lng"]

for uc in unwanted_columns:
    if uc in df.columns:
        df.drop(columns=uc, inplace=True)
        print("{} column removed.".format(uc))

# show remaining columns
print("Remaining columns: \n{}".format(df.columns))
```

```
#####
      REMOVE UNWANTED COLUMNS
#####
[UID] column removed.
[Interaction] column removed.
[Lat] column removed.
[Lng] column removed.
Remaining columns:
Index(['CaseOrder', 'Customer_id', 'City', 'State', 'County',
      'Zip',
      'Population', 'Area', 'TimeZone', 'Job', 'Children',
      'Age', 'Income',
      'Marital', 'Gender', 'Churn', 'Outage_sec_perweek',
      'Email', 'Contacts',
      'Yearly equip_failure', 'Techie', 'Contract',
      'Port_modem', 'Tablet',
      'InternetService', 'Phone', 'Multiple',
      'OnlineSecurity',
      'OnlineBackup', 'DeviceProtection', 'TechSupport',
      'StreamingTV',
      'StreamingMovies', 'PaperlessBilling',
      'PaymentMethod', 'Tenure',
      'MonthlyCharge', 'Bandwidth_GB_Year', 'Item1',
      'Item2', 'Item3',
      'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

Numerical Data

Here is the list of available continuous data:

```
print(heading_toString("CONTINUOUS DATA"))
print(df.select_dtypes(include="float").info())
```

```
#####
          CONTINUOUS DATA
#####
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 5 columns):
#   Column                      Non-Null Count  Dtype
---  ---
0   Income                      10000 non-null  float64
1   Outage_sec_perweek          10000 non-null  float64
2   Tenure                      10000 non-null  float64
3   MonthlyCharge                10000 non-null  float64
4   Bandwidth_GB_Year           10000 non-null  float64
dtypes: float64(5)
memory usage: 390.8 KB
None
```

Here is a list of available integer data:

```
print(heading_toString("INTEGER DATA"))
print(df.select_dtypes(include="integer").info())
```

```
#####
          INTEGER DATA
#####
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 16 columns):
#   Column                      Non-Null Count  Dtype
---  ---
0   CaseOrder                   10000 non-null  int64
1   Zip                         10000 non-null  int64
2   Population                  10000 non-null  int64
3   Children                    10000 non-null  int64
4   Age                         10000 non-null  int64
5   Email                       10000 non-null  int64
6   Contacts                    10000 non-null  int64
7   Yearly_equip_failure        10000 non-null  int64
8   Item1                       10000 non-null  int64
9   Item2                       10000 non-null  int64
10  Item3                       10000 non-null  int64
11  Item4                       10000 non-null  int64
12  Item5                       10000 non-null  int64
13  Item6                       10000 non-null  int64
14  Item7                       10000 non-null  int64
15  Item8                       10000 non-null  int64
dtypes: int64(16)
memory usage: 1.2 MB
None
```

Categorical Data

Here is a list of available categorical data:

```
print(heading_toString("CATEGORICAL DATA"))
print(df.select_dtypes(include="object").info())
```

```
#####
      CATEGORICAL DATA
#####
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 25 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   Customer_id           10000 non-null  object
1   City                  10000 non-null  object
2   State                 10000 non-null  object
3   County               10000 non-null  object
4   Area                 10000 non-null  object
5   TimeZone             10000 non-null  object
6   Job                  10000 non-null  object
7   Marital              10000 non-null  object
8   Gender               10000 non-null  object
9   Churn                10000 non-null  object
10  Techie               10000 non-null  object
11  Contract             10000 non-null  object
12  Port_modem           10000 non-null  object
13  Tablet               10000 non-null  object
14  InternetService      10000 non-null  object
15  Phone               10000 non-null  object
16  Multiple             10000 non-null  object
17  OnlineSecurity       10000 non-null  object
18  OnlineBackup         10000 non-null  object
19  DeviceProtection     10000 non-null  object
20  TechSupport          10000 non-null  object
21  StreamingTV          10000 non-null  object
22  StreamingMovies      10000 non-null  object
23  PaperlessBilling     10000 non-null  object
24  PaymentMethod        10000 non-null  object
dtypes: object(25)
memory usage: 1.9+ MB
None
```


Part B Analysis, code and output

Part B1. All the analysis is conducted using Python programming language using Microsoft's Visual Studio Code IDE.

Part B2. The complete .PY file is attached. In addition, selected 'Code Snippets' and 'Terminal Output' will be included and described in this document

Part B3. This analysis will use the '**Chi-square, Independence Test**' The test is suited to the analysis of two categorical variables.

Here is the procedure I used to conduct the Chi-square **independence** test. I wrote this as a procedure so that I can reused and standardize all my testing. The assessment asks to complete one analysis test, but because it was a method, I was able to quickly run and look at other predictor variables as well. All the procedures I used were consolidated into a library file, mm_lib.py, which is attached. Here is the function that calculates and outputs the chi-square test:

```
def chi_square_analysis(target,predictor,bins,prob,df):
    ''' Calculate and display results of chi-square independence test

    Arg:
        target - target variable
        predictor - predictor variable
        bins - number of bins, if numerical, 0 if categorical
        prob - probability
        df - dataframe

    Return: outputs analysis to terminal.

    ...
    import pandas as pd
    from scipy.stats import chi2_contingency
    from scipy.stats import chi2
    if bins == 0:
        # variable is categorical
        contingency= pd.crosstab(df[target], df[predictor])
    else:
        # variable is numerical
        mc_groups =pd.cut(df[predictor], bins=bins)
        contingency= pd.crosstab(df[target], mc_groups)

    # print out the contingency table
    print(contingency.T)

    stat, p, dof, expected = chi2_contingency(contingency)
    alpha = 1 - prob # significance level
    critical = chi2.ppf(prob, dof)

    # interpret test-statistic
    if abs(stat) >= critical:
        print('At an alpha level of {}, the critical value is {}'.format(
            round(alpha,3),round(critical,3)))
        print('chi^2 = {} > {}, therefore, variables are dependent (r
            eject H0).'.format(round(stat,3),round(critical,3)))
        #print('Also, we will need to perform additional analysis to
            determine extent of the relationship.')
    else:
        print('chi^2 = {} <= {}, therefore, variables are independent
            (fail to reject H0).'.format(round(stat,3),round(critical,3)))

    print(' p-value: {:>8.3f}'.format(p))
    print('      dof: {:>8d}'.format(dof))
    print('      alpha: {:>8.3f}'.format(alpha))
    print('      stat: {:>8.3f}'.format(stat))
    print('critical: {:>8.3f}'.format(critical))
```

Here is the snippet where the procedure was called:

```
# CHI-SQUARE TEST
target = 'Churn'
prob = 0.999
predictor = 'MonthlyCharge' # numerical
print(heading_toString("PART B - CHI-
SQUARE INDEPENDENCE TEST - " + target + " vs. " + predictor))
bins = 6

# use library function to perform test
chi_square_analysis(target,predictor,bins,prob,df)

# use library function to print out distribution table
print(heading_toString("CHI-SQUARE DISTRIBUTION TABLE"))
print(chi_square_dist_table(0,0))
```

The null hypothesis is that 'Churn' and 'MonthlyCharge' are independent of each other, the null hypothesis is defined as H_0 .

Null Hypothesis (H_0) – variables are independent

Alternate Hypothesis (H_1) – variables are not independent

Here is the terminal output for the test:

```
#####
#####
PART B - CHI-SQUARE INDEPENDENCE TEST - Churn vs.
MonthlyCharge
#####
#####
Churn          No  Yes
MonthlyCharge
(79.769, 115.009] 755  35
(115.009, 150.039] 2477 338
(150.039, 185.07] 2356 693
(185.07, 220.1] 1009 608
(220.1, 255.13] 579 715
(255.13, 290.16] 174 261
At an alpha level of 0.001, the critical value is 20.515.
chi^2 = 1425.642 > 20.515, therefore, variables are dependent
(reject H0).
p-value: 0.000
dof: 5
alpha: 0.001
stat: 1425.642
critical: 20.515
```

A couple of notes for this test. 'MonthlyCharge' is a numerical variable, so I used **bins=6** to break the numerical values into groups in order to conduct the test. The p-value is extremely low, lower than the alpha of 0.001, and the calculated statistic was extremely high compared to the critical value. Both indicate strong support that the two variables are dependent. The analysis indicates to '**reject null hypothesis**'. Also, further testing and research will be needed to determine the full extent of the relationship.

The test critical value, χ^2 , was determined as a function of degree of freedom (dof) and probability (prob). I used the chi2 method in the scipy.stats package to calculate the value, but I also wanted to see the table for reference, so I made a library function to create a distribution table, here is the Chi-square lower-tail distribution table from terminal output using my function:

Table 1 CHI-SQUARE DISTRIBUTION TABLE

***** CHI-SQUARE DISTRIBUTION TABLE *****					
Lower-tail critical values of chi-Square distribution:					
dof	0.100	0.050	0.025	0.010	0.001
1	2.71	3.84	5.02	6.63	10.83
2	4.61	5.99	7.38	9.21	13.82
3	6.25	7.81	9.35	11.34	16.27
4	7.78	9.49	11.14	13.28	18.47
5	9.24	11.07	12.83	15.09	20.52
6	10.64	12.59	14.45	16.81	22.46
7	12.02	14.07	16.01	18.48	24.32
8	13.36	15.51	17.53	20.09	26.12
9	14.68	16.92	19.02	21.67	27.88
10	15.99	18.31	20.48	23.21	29.59
11	17.28	19.68	21.92	24.72	31.26
12	18.55	21.03	23.34	26.22	32.91
13	19.81	22.36	24.74	27.69	34.53
14	21.06	23.68	26.12	29.14	36.12
15	22.31	25.00	27.49	30.58	37.70
16	23.54	26.30	28.85	32.00	39.25
17	24.77	27.59	30.19	33.41	40.79
18	25.99	28.87	31.53	34.81	42.31
19	27.20	30.14	32.85	36.19	43.82

Table was generated using the following code:

```
def chi_square_dist_table(dof,prob):
    '''create chi square distribution table

    Args:
        dof - degree of freedom
        prob - significance level (alpha)

    Returns:
        A string representation of the chi-square
        lower-left distribution table.

    ...
    from scipy.stats import chi2
    prob_range = [ .10, 0.05, 0.025, 0.01, 0.001]
    dof_range = range(1,20)
    output = 'Lower-tail critical values of chi-
    Square distribution:\n'.format()

    # header row
    output += '{:>4}'.format('dof') # blank space for first col
    for p in prob_range:
        output += '{:>8.3f}'.format(p)
```

```
output += '\n' + '-' * 47

# each row
for d in dof_range:
    output += '\n{:4d}'.format(d)
    for p in prob_range:
        temp = chi2.isf(p, d)
        if d == dof and p == prob:
            output += '{:>8.2f}'.format(temp)
        else:
            output += '{:>8.2f}'.format(temp)

return output
```

Part C Univariate Statistics

Part C. Identify the distribution of two continuous variables and two categorical variables using univariate statistics from your cleaned and prepared data. The plots were created and then saved as .PNG files.

Descriptive Statistics

To begin, it is helpful to show traditional descriptive statistics for the numerical data, this includes the center of the data and the range. Here is the terminal output showing the descriptive statistics of the numerical (float) variables:

```
#####
DESCRIPTIVE STATS
#####
      Income  Outage_sec_perweek    Tenure  MonthlyCharge
Bandwidth_GB_Year
count    10000.00         10000.00  10000.00         10000.00
10000.00
mean      39806.93             10.00     34.53          172.62
3392.34
std       28199.92              2.98     26.44           42.94
2185.29
min        348.67              0.10      1.00           79.98
155.51
25%       19224.72              8.02      7.92          139.98
1236.47
50%       33170.60             10.02     35.43          167.48
3279.54
75%       53246.17             11.97     61.48          200.73
5586.14
max       258900.70            21.21     72.00          290.16
7158.98
```

```
print(df.select_dtypes(include="float").describe().round(2))
```

The next two plots are countplots of univariate categorical variables:

- Fig_1_Churn_categorical_countplot.png
- Fig_2_InternetService_categorical_countplot.png

Here is the code:

```
# visualization of selected categorical data
univariate_categorical = {
    "1": "Churn",
    "2": "InternetService",
}

for key in univariate_categorical:
    fig, ax = plt.subplots()
    c = univariate_categorical[key]
    sns.countplot(df[c])
    plt.xlabel(c)
    plt.ylabel('Count')
    count_figures = key
    #title = 'Hello world'
    sub_title = '{}_{}_{}'.format(c, 'categorical', 'countplot')
    title = '{}'.format(c)
    fname = 'Fig_{}_{}_{}'.format(str(count_figures), sub_title, "png")
    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"facecolor": "orange", "alpha": 0.5, "pad": 5})
    fig.tight_layout()

    # save file in the course figure's folder
    plt.savefig(os.path.join(folder, fname))
    plt.close()
    print('figure saved at: {}'.format(fname))
```

Count Plot – Churn (Cat)

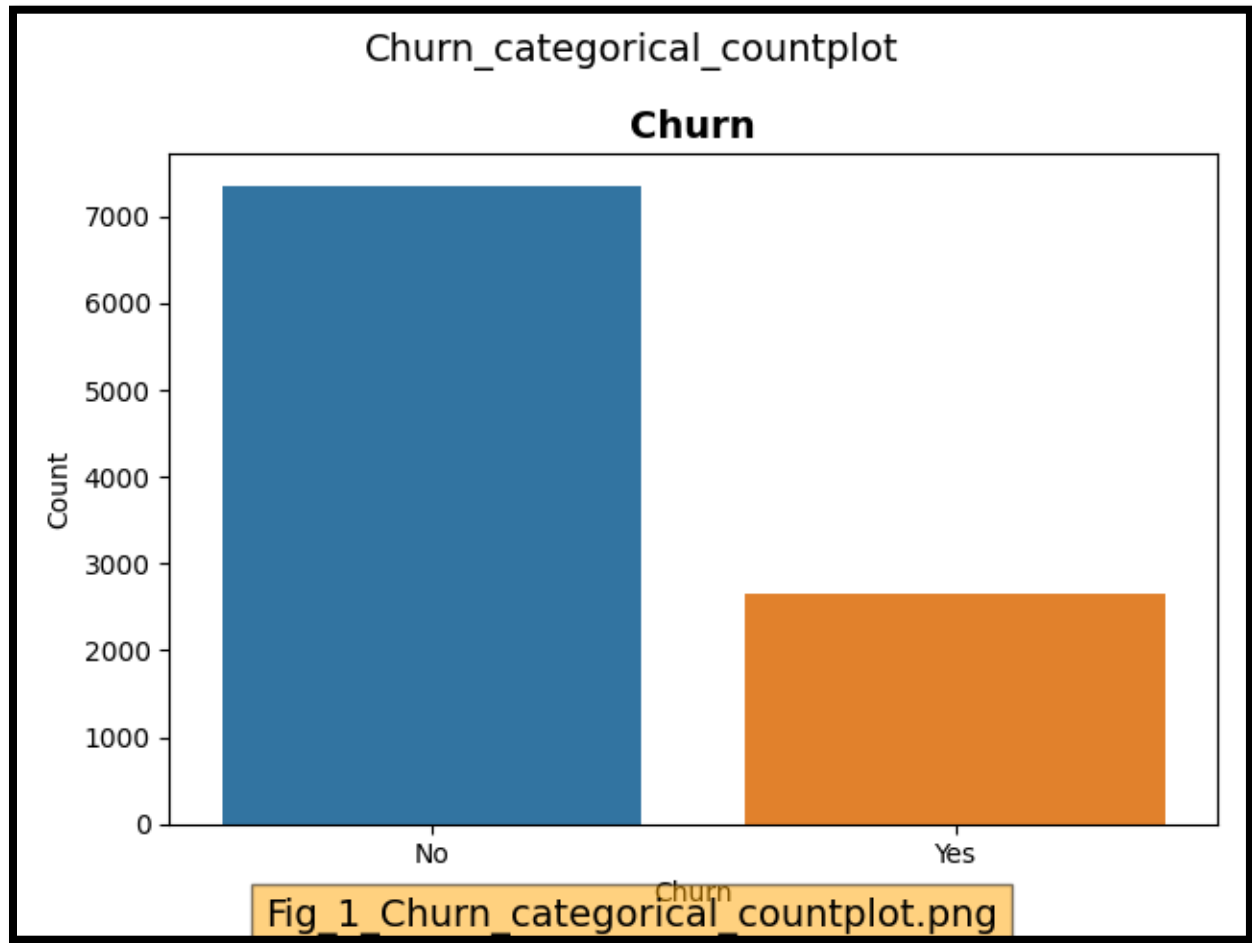


Figure 1 Churn Cat_countplot (.PNG)

Count Plot – InternetService (Cat)

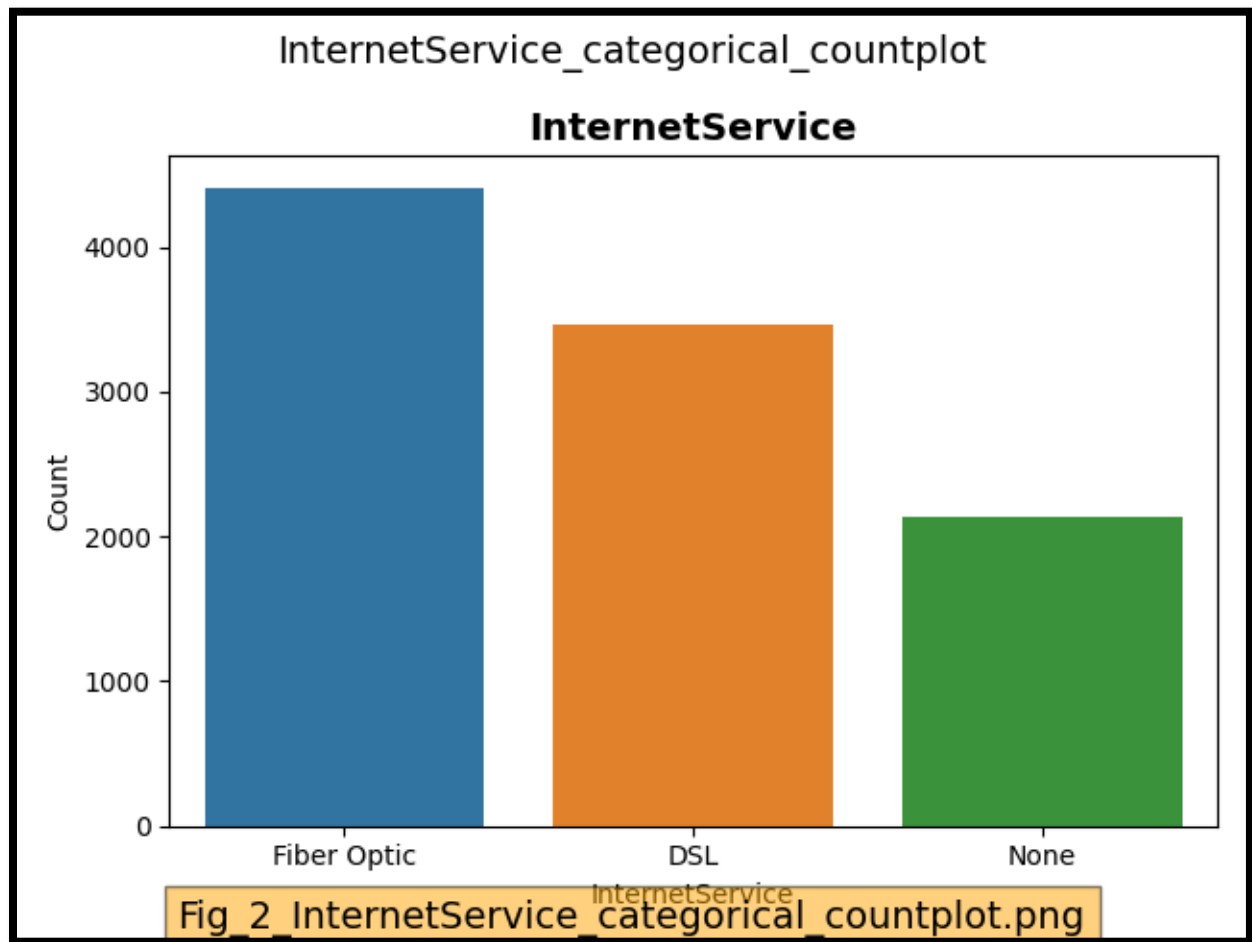


Figure 2 InternetService Cat_countplot (.PNG)

The next two plots are univariate continuous:

- Fig_3_MonthlyCharge_continuous_scatterplot.png
- Fig_4_Income_continuous_scatterplot.png

Here is the code:

```
# visualization of selected univariate continuous data
univariate_continuous = {
    "3": "MonthlyCharge",
    "4": "Income",
}

for key in univariate_continuous:
    fig, ax = plt.subplots()
    c = univariate_continuous[key]
    plt.scatter(df.index, df[c])
    plt.xlabel(c)
    plt.ylabel('Count')
    count_figures = key
    #title = 'Hello world'
    sub_title = '{}_{}_{}'.format(c, 'continuous', 'scatterplot')
    title = '{}'.format(c)
    fname = 'Fig_{}_{}_{}'.format(str(count_figures), sub_title, "png")
    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"facecolor": "orange", "alpha": 0.5, "pad": 5})
    fig.tight_layout()

    # save file in the course figure's folder
    plt.savefig(os.path.join(figure_folder, fname))
    plt.close()
    print('figure saved at: {}'.format(fname))
```

Scatter Plot – MonthlyCharge (Cont)

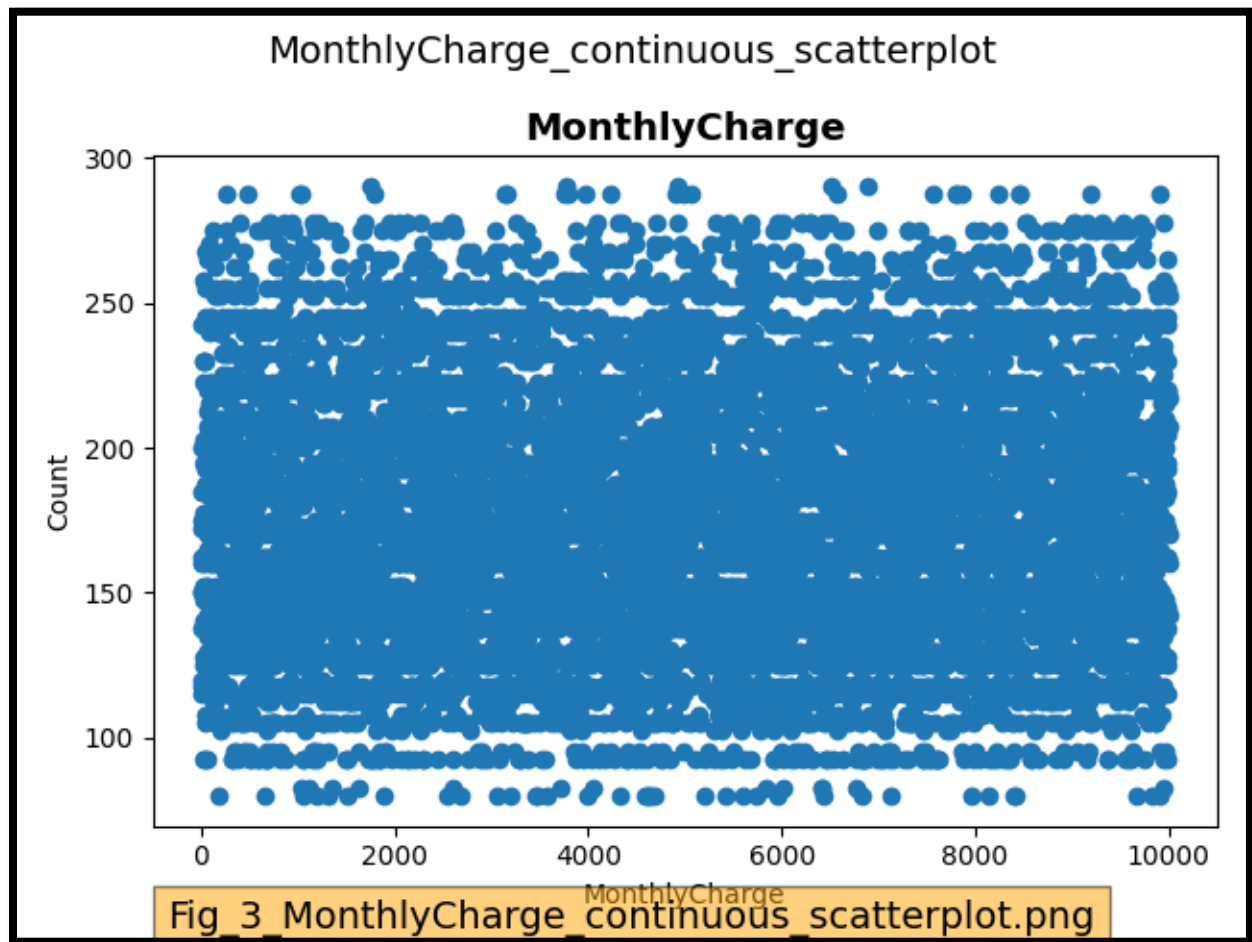


Figure 3 MonthlyCharge Cont_scatterplot (.PNG)

Scatter Plot – Income (Cont)

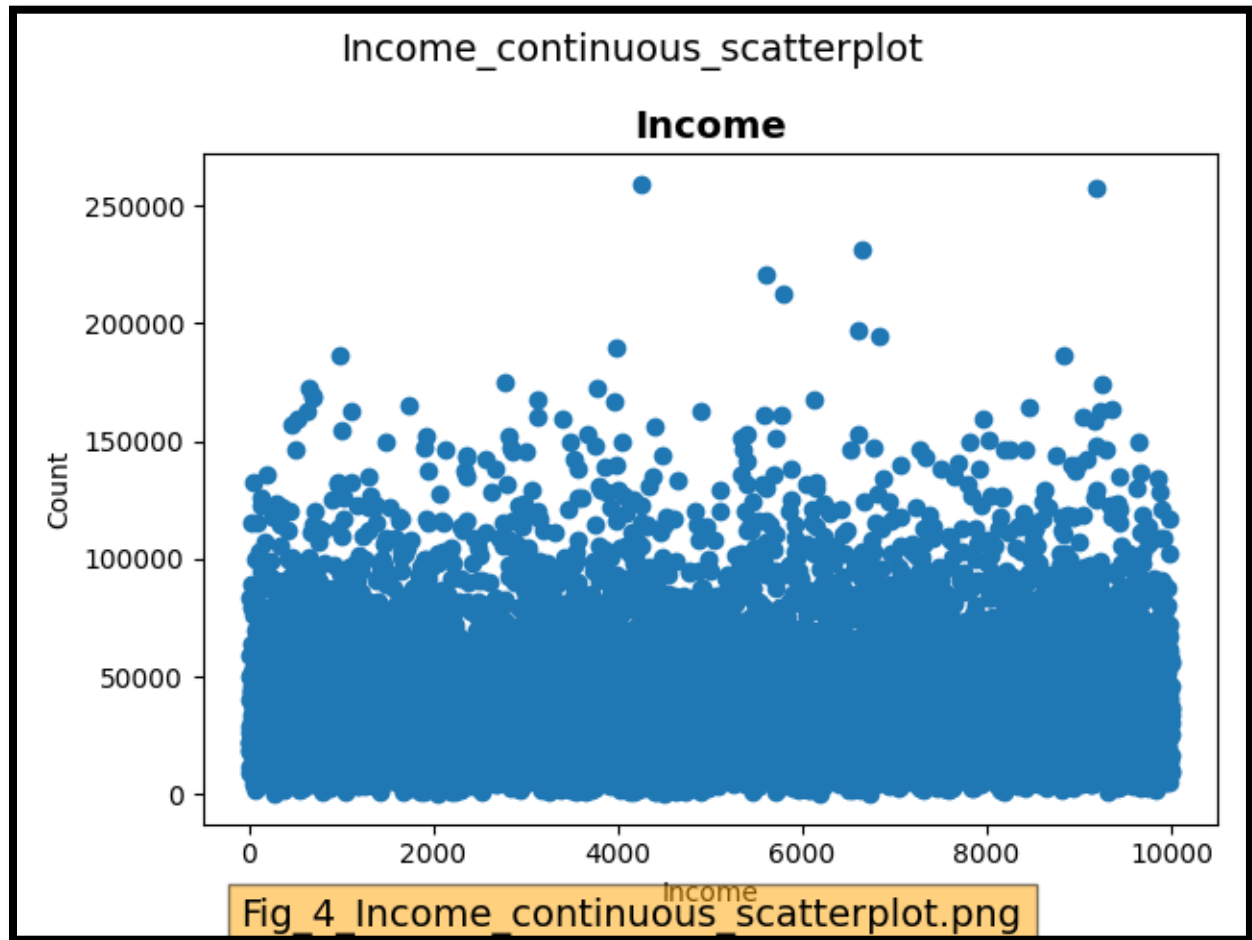


Figure 4 Income Cont_scatterplot (.PNG)

Box Plot – Normalized Problem Data (Cat)

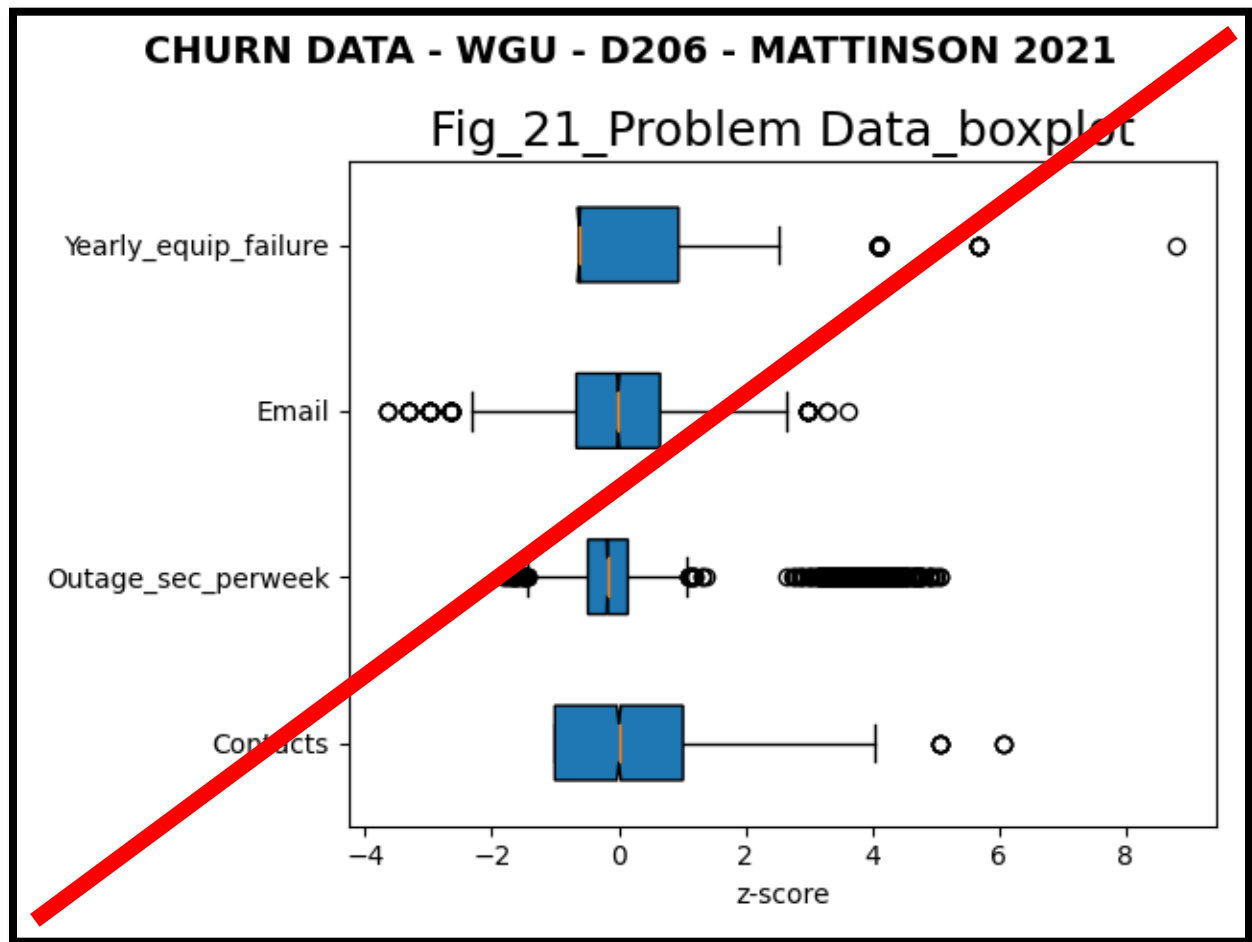


Figure 5 Problem Data_boxplot (.PNG) - not used

Figure notes: This was created using a list of related, problem, variables:

1. Yearly equip_failure (Cont)
2. Email (Integer)
3. Outage_sec_perweek (Integer)
4. Contacts (Integer)

Box Plot – Normalized Demographic Data (Integer, Cont)

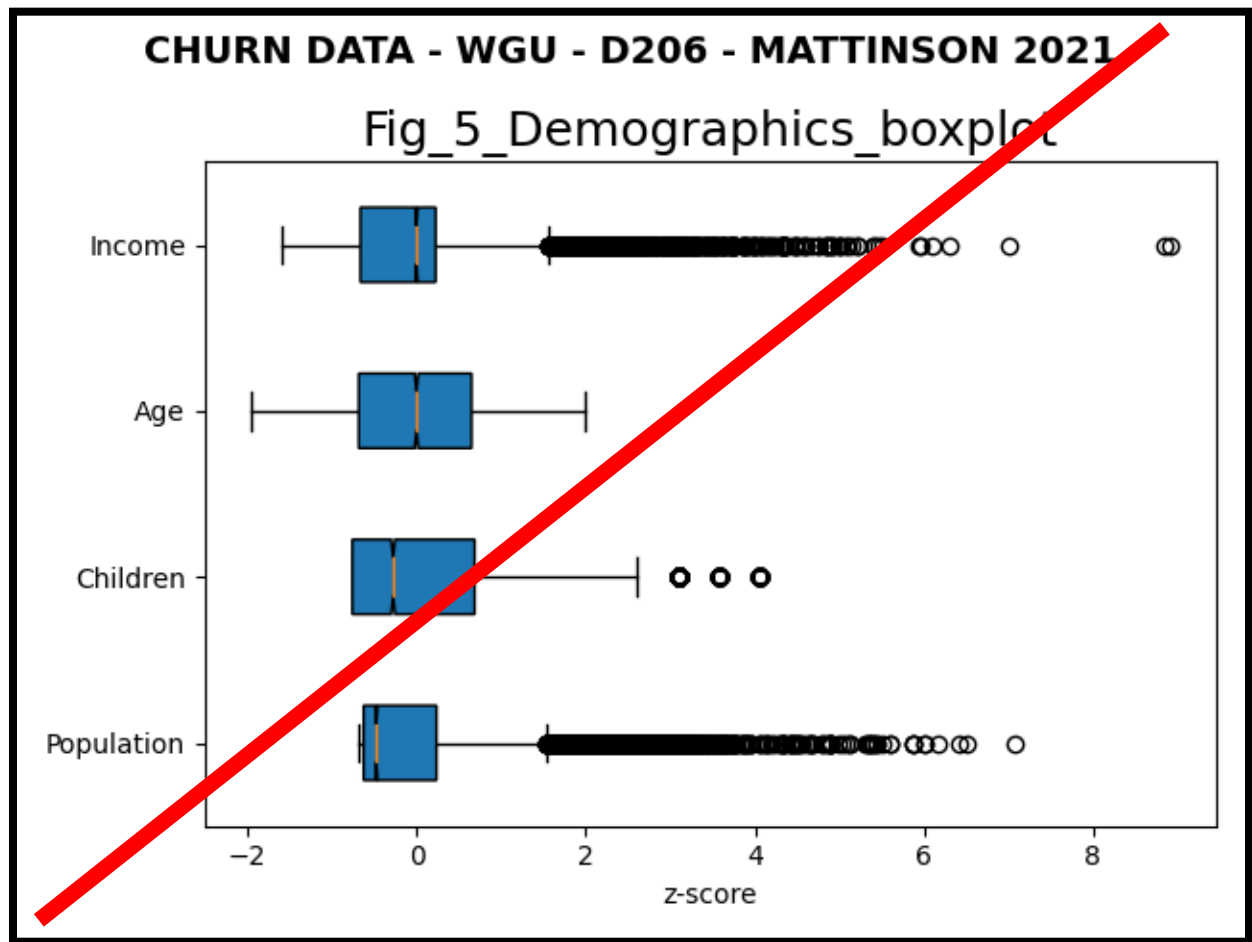
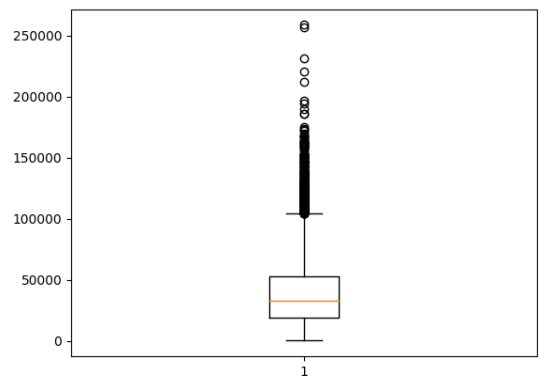
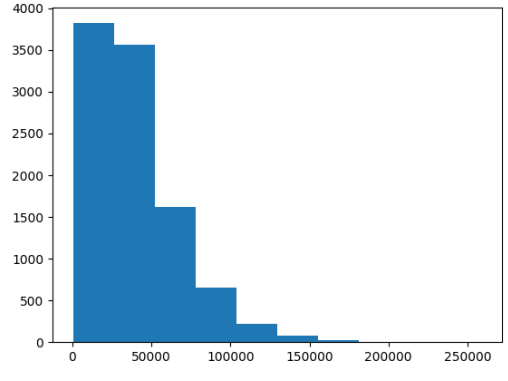
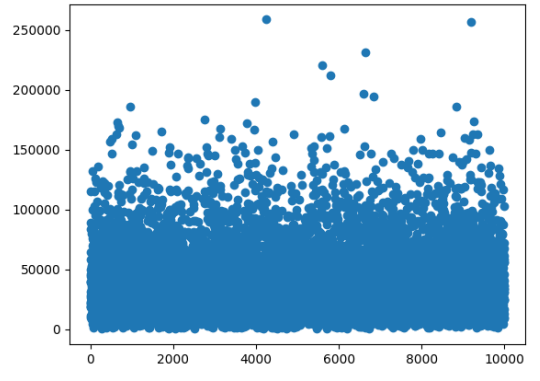


Figure 6 Demographics_boxplot (.PNG) - not used

Figure notes: This was created using a list of related, demographic, variables:

5. Income (Cont)
6. Age (Integer)
7. Children (Integer)
8. Population (Cont)

Sample of different plots – Income (Cont)

 <p>A boxplot showing the distribution of income. The y-axis ranges from 0 to 250,000. The box is centered around 30,000, with a median line at approximately 35,000. The whiskers extend from 0 to 100,000. There are several outliers represented by open circles, with the highest outlier at 250,000.</p> <p><i>Figure 7 Boxplot - Income (Cont)</i></p>	 <p>A histogram showing the frequency of income. The x-axis ranges from 0 to 250,000, and the y-axis ranges from 0 to 4,000. The distribution is right-skewed, with the highest frequency (around 3,800) in the first bin (0-50,000) and a long tail extending to the right.</p> <p><i>Figure 8 Histogram plot - Income (Cont)</i></p>
<p>BOX PLOT</p>  <p>A scatter plot showing the distribution of income. The x-axis ranges from 0 to 10,000, and the y-axis ranges from 0 to 250,000. The plot shows a dense cloud of blue points, with a few outliers at higher income levels.</p> <p><i>Figure 9 Scatter plot - Income (Cont)</i></p> <p>SCATTER PLOT</p>	<p>HISTOGRAM</p>

```
plt.scatter(df.index,df['Income'])
plt.hist(df['Income'])
plt.boxplot(df['Income'])
```

Part D Bivariate Statistics

Part D. Identify the distribution of two continuous variables and two categorical variables using bivariate statistics from your cleaned and prepared data. [Here is a list of all bivariate plots created:](#)

- Fig_10_Bivariate Income vs MonthlyCharge_Jointplot
- Fig_11_InternetService_bivariate_countplot.png
- Fig_12_TechSupport_bivariate_countplot.png
- Fig_13_PaymentMethod_bivariate_countplot.png
- Fig_14_Tablet_bivariate_countplot.png
- Fig_15_Gender_bivariate_countplot.png
- Fig_16_Port_modem_bivariate_countplot.png
- Fig_17_Techie_bivariate_countplot.png
- Fig_18_MonthlyCharge_bivariate_stacked histogram.png
- Fig_19_Bandwidth_GB_Year_bivariate_stacked histogram.png
- Fig_20_Tenure_bivariate_stacked histogram.png
- Fig_21_Outage_sec_perweek_bivariate_stacked histogram.png
- Fig_22_Income_bivariate_stacked histogram.png
- Fig_23_Gender_bivariate_stacked bar.png
- Fig_24_InternetService_bivariate_stacked bar.png

Joint Plot – Income (Cont) vs MonthlyCharge (Cont)

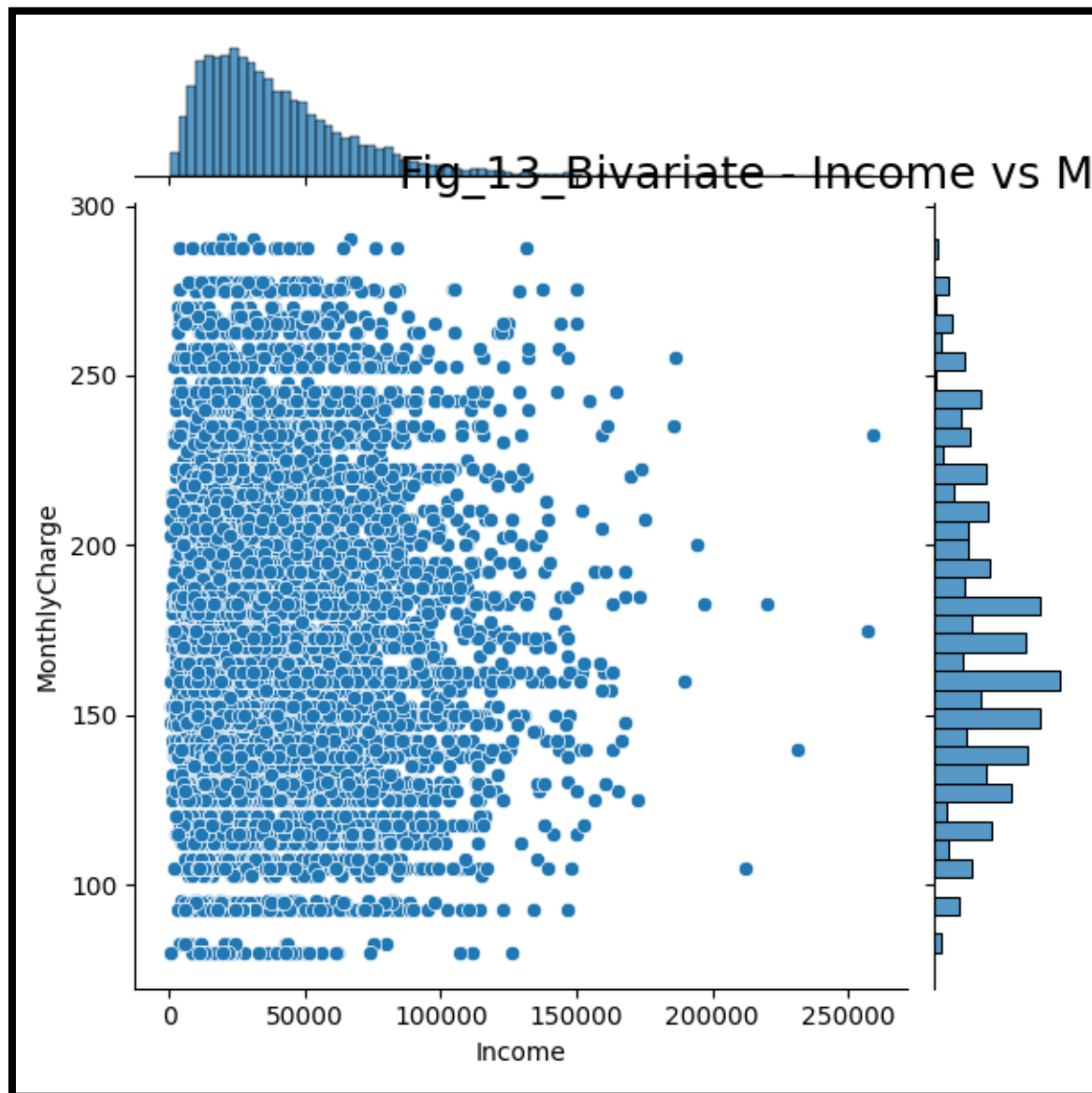


Figure 10 Birvariate - Income vs. MonthlyCharge (.PNG)

Figure notes. This plot allows you to see both variables as scatter plot, but also the individual distribution above or to the side of the data.

Ref: <https://mlcourse.ai/articles/topic2-visual-data-analysis-in-python/#2.-Univariate-visualization>

The next bivariate plots were countplots of two categorical variables:

- Fig_11_InternetService_bivariate_countplot.png
- Fig_12_TechSupport_bivariate_countplot.png
- Fig_13_PaymentMethod_bivariate_countplot.png
- Fig_14_Tablet_bivariate_countplot.png
- Fig_15_Gender_bivariate_countplot.png
- Fig_16_Port_modem_bivariate_countplot.png
- Fig_17_Techie_bivariate_countplot.png

Here is the code snippet that created the plots:

```
# visualization of selected bivariate data
bivariate = {
    '11': 'InternetService',
    '12': 'TechSupport',
    '13': 'PaymentMethod',
    '14': 'Tablet',
    '15': 'Gender',
    '16': 'Port_modem',
    '17': 'Techie',
}

for key in bivariate:
    fig, ax = plt.subplots()
    c = bivariate[key]
    sns.countplot(x='Churn', hue=c, data=df, palette='hls')
    plt.xlabel('{} vs. {}'.format(c, 'Churn'))
    plt.ylabel('Count')
    count_figures = key
    sub_title = '{}_{}_{}'.format(c, 'bivariate', 'countplot')
    title = '{} vs. {}'.format(c, 'Churn')
    fname = 'Fig_{}_{}_{}'.format(str(count_figures), sub_title, "png")
    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"facecolor": "orange", "alpha": 0.5, "pad": 5})
    fig.tight_layout()

    # save file in the course figure's folder
    plt.savefig(os.path.join(folder, fname))
    plt.close()
    print('figure saved at: {}'.format(fname))
```

InternetService_bivariate_countplot

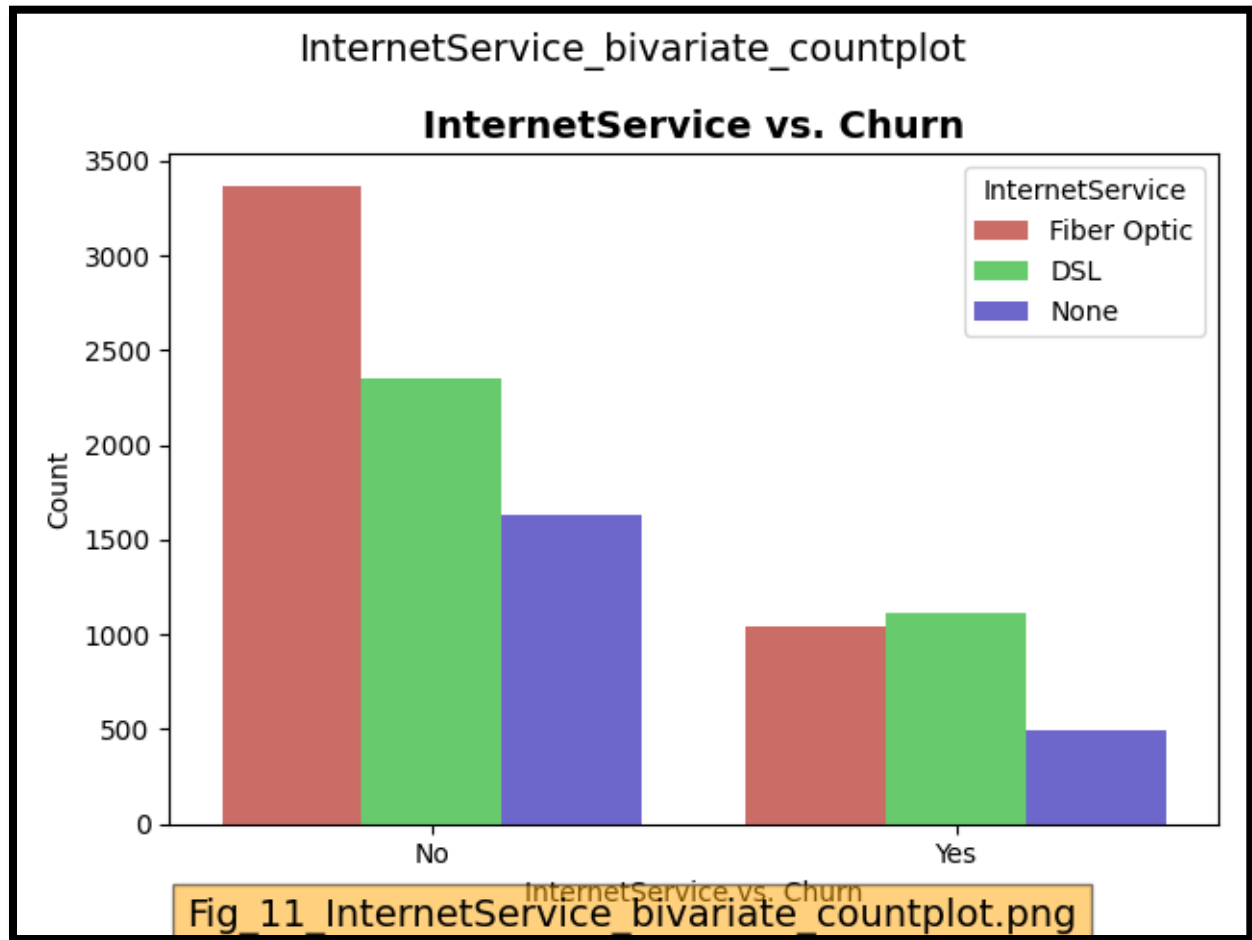


Figure 11 InternetService_bivariate_countplot

TechSupport_bivariate_countplot

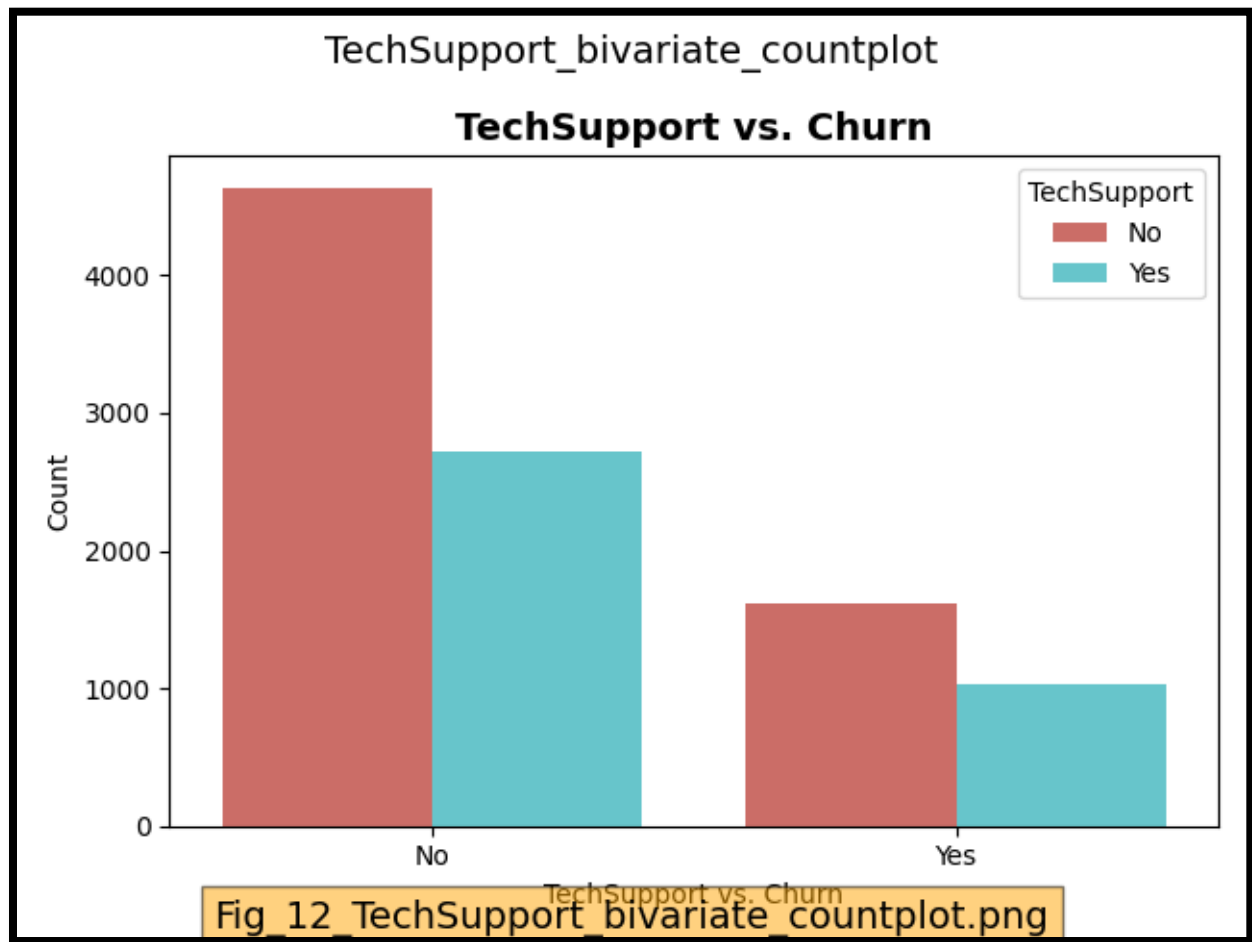


Figure 12 TechSupport_bivariate_countplot

PaymentMethod_bivariate_countplot

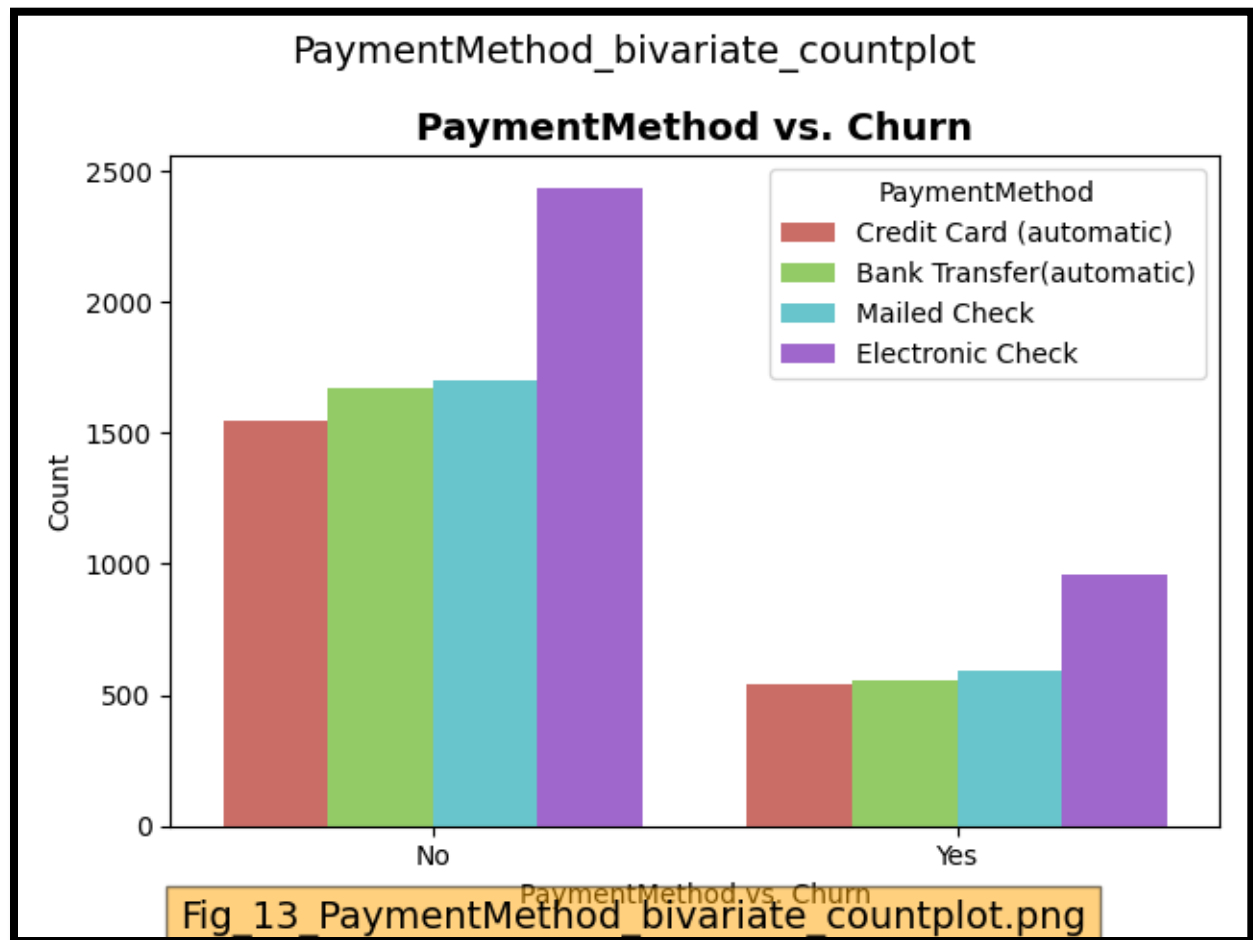


Figure 13 PaymentMethod_bivariate_countplot

Tablet_bivariate_countplot

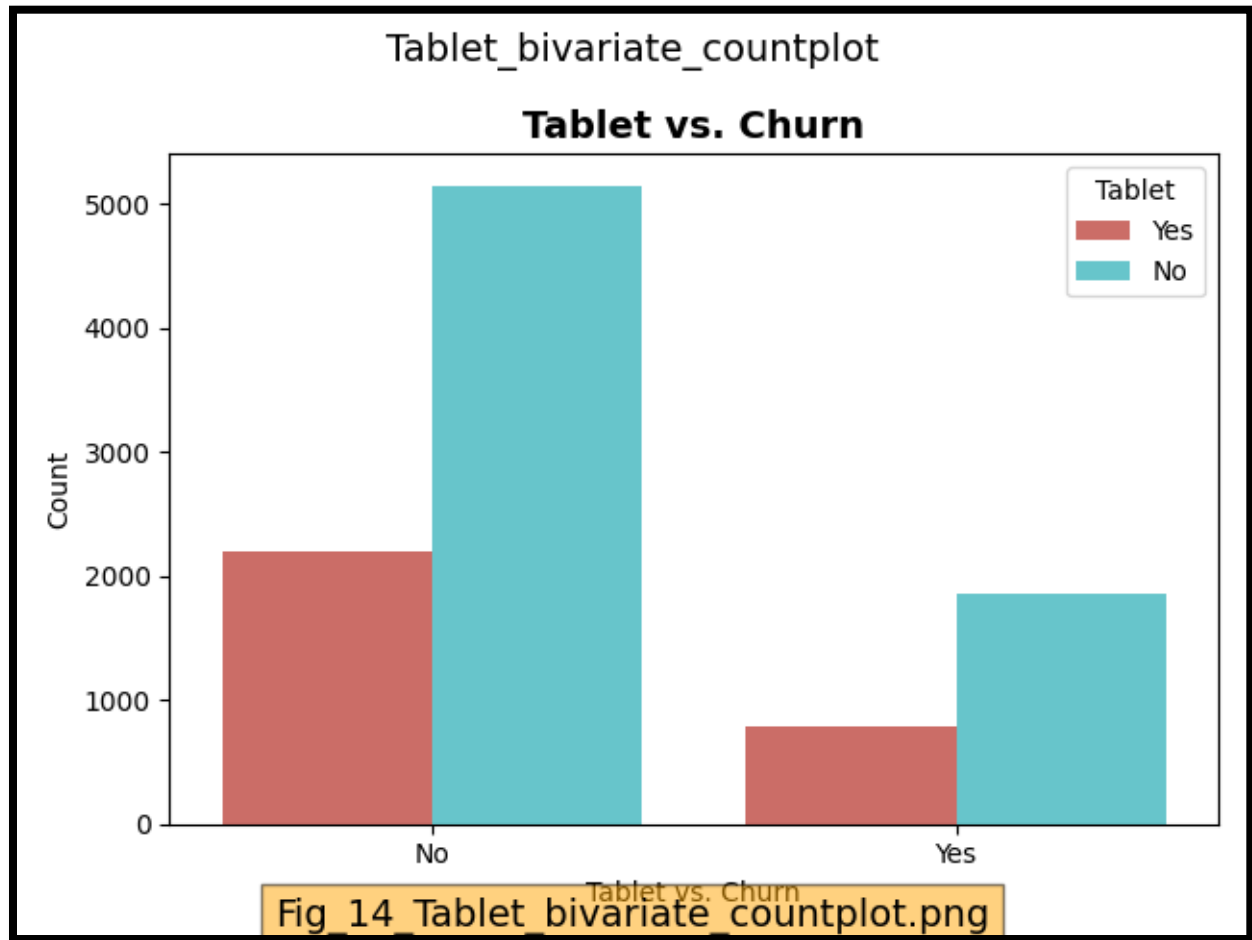


Figure 14 Tablet_bivariate_countplot

Gender_bivariate_countplot

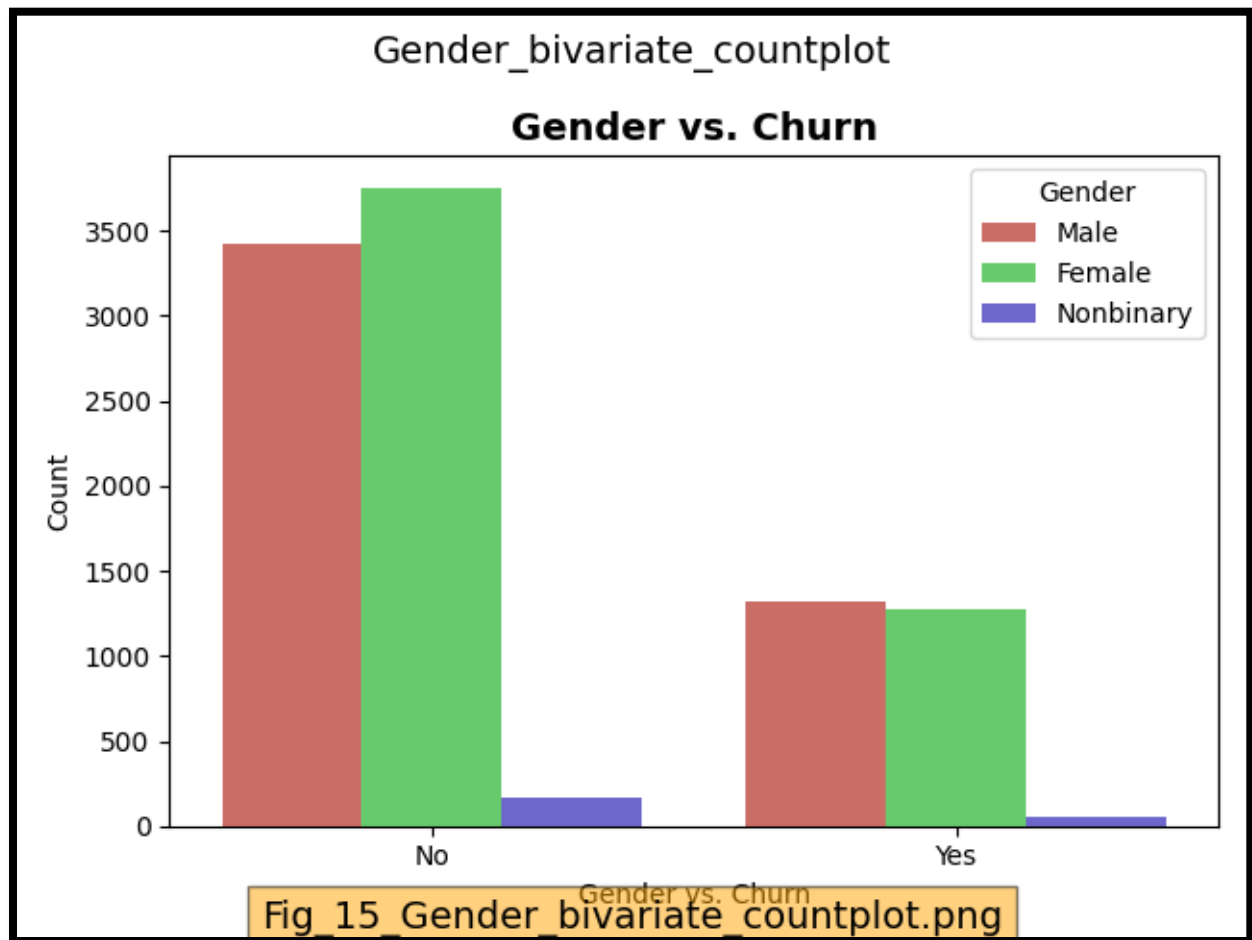


Figure 15 Gender_bivariate_countplot

Port_modem_bivariate_countplot

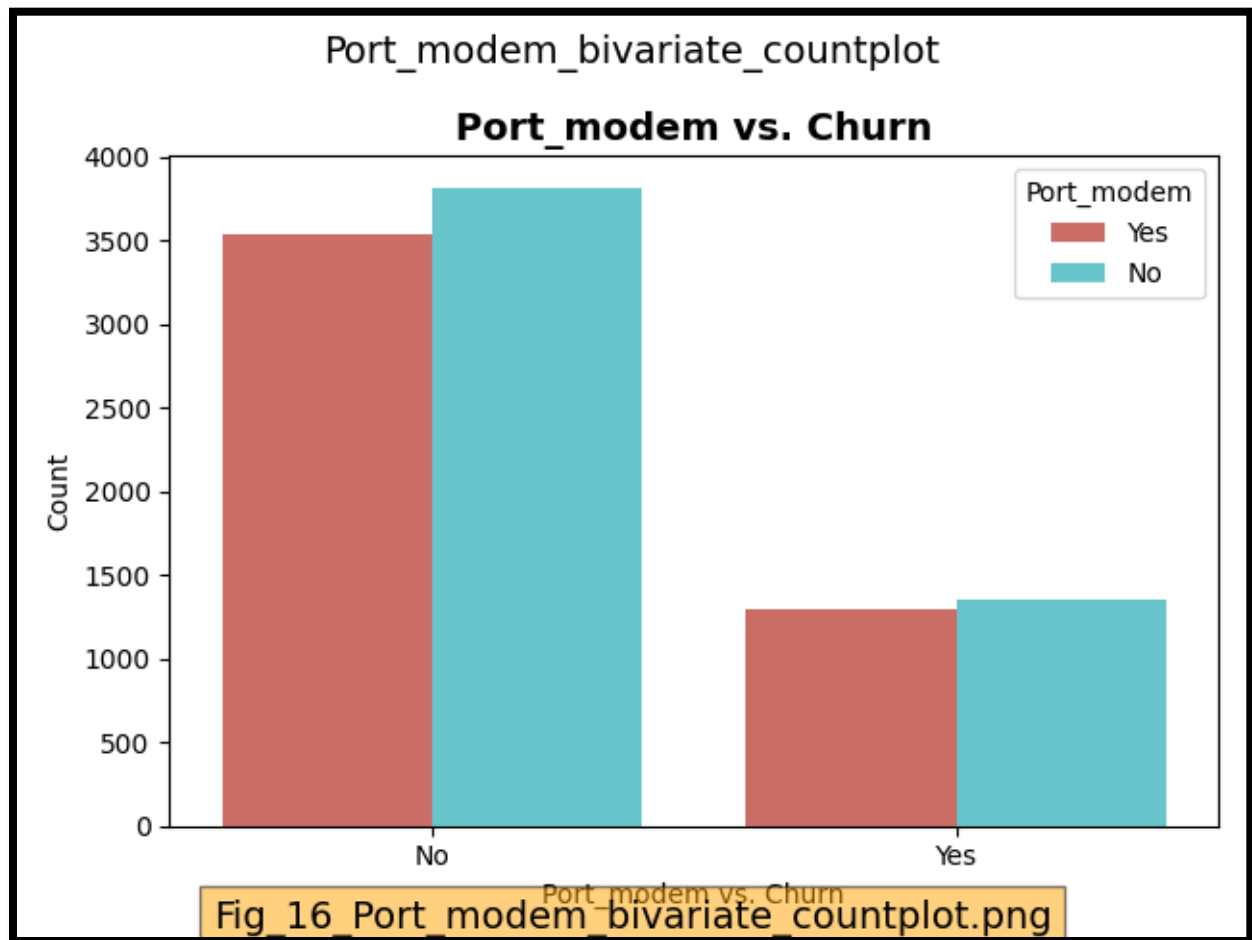


Figure 16 Port_modem_bivariate_countplot

Techie_bivariate_countplot

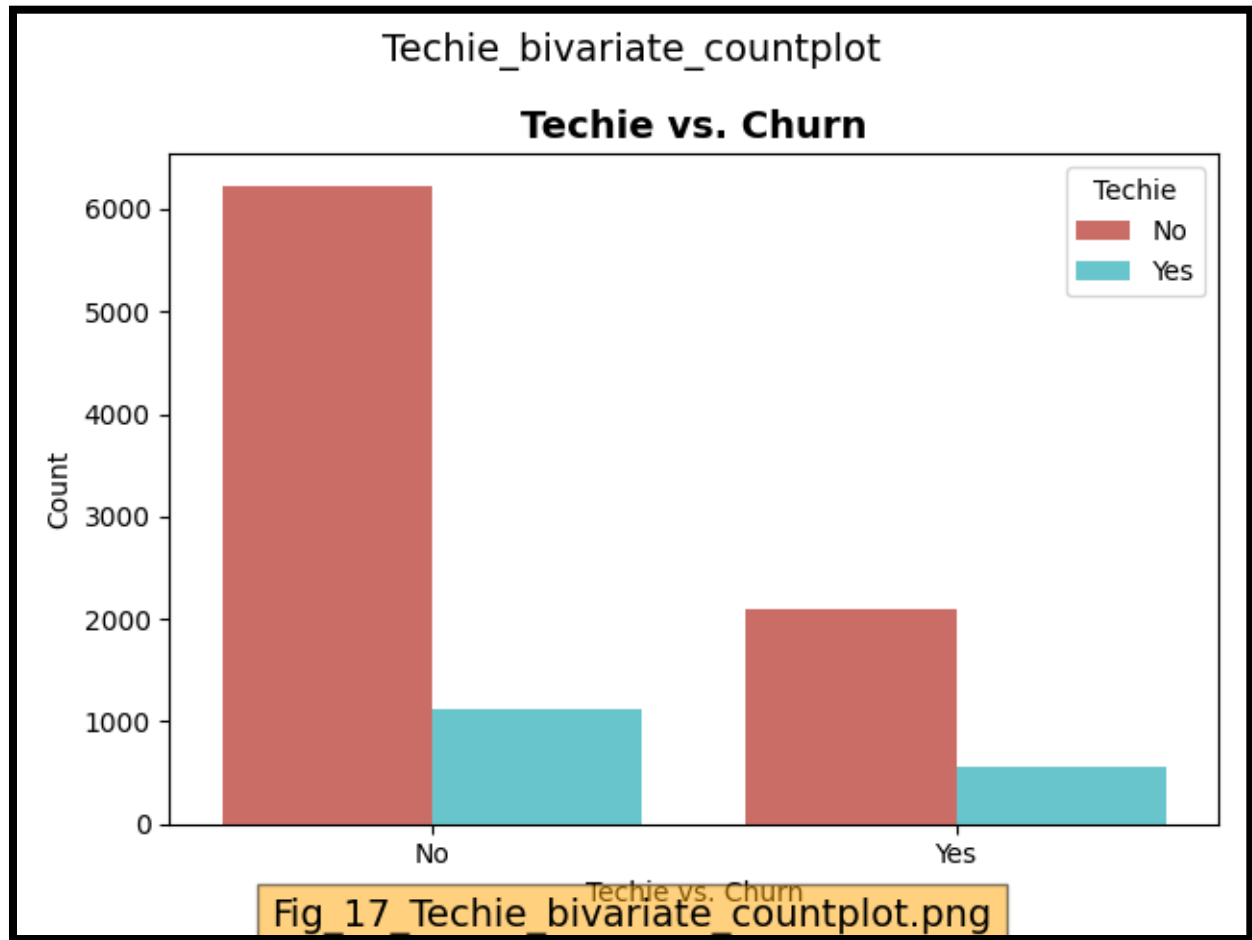


Figure 17 Techie_bivariate_countplot

The next plots are bivariate plots of continuous variables overlayed with the target 'Churn' (categorical) variable:

- Fig_18_MonthlyCharge_bivariate_stacked histogram.png
- Fig_19_Bandwidth_GB_Year_bivariate_stacked histogram.png

Here is the code:

```
# visualization of selected continous data
bivariate = {
    '18': 'MonthlyCharge',
    '19': 'Bandwidth_GB_Year',
    '20': 'Tenure',
    '21': 'Outage_sec_perweek',
    '22': 'Income',
}

for key in bivariate:
    c = bivariate[key]
    target = 'Churn'

    # print crosstab table associated with figure
    print(heading_toString(c + " crosstab"))
    print(pd.crosstab(pd.cut(df[c], bins = 6), df[target], margins=True))

    df_yes = df[df.Churn == "Yes"][c]
    df_no = df[df.Churn == "No"][c]
    yes_mean = df_yes.mean()
    no_mean = df_no.mean()
    fig, ax = plt.subplots()
    _, bins, _ = ax.hist([df_yes, df_no], bins=6, stacked=True)
    ax.legend(["Churn - Yes", "Churn - No"])
    ymin, ymax = ax.get_ylim()
    xmin, xmax = ax.get_xlim()
    ax.axvline(yes_mean, color="blue", lw=2) # yes mean
    ax.axvline(no_mean, color="orangered", lw=2) # no mean
    ax.text(
        (xmax - xmin) / 2,
        (ymax - ymin) / 2,
        "Delta:\n" + str(round(abs(yes_mean - no_mean), 2)),
        bbox={"facecolor": "white"},
    )
    plt.xlabel('{} vs. {}'.format(c, 'Churn'))
    plt.ylabel('Count')
    count_figures = key
    sub_title = '{}_{}_{}'.format(c, 'bivariate', 'stacked histogram')
    title = '{} vs. {}'.format(c, 'Churn')
    fname = 'Fig_{}_{}.{}'.format(str(count_figures), sub_title, "png")

    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"facecolor": "orange", "alpha": 0.5, "pad": 5})
    fig.tight_layout()

    # save file in the course figure's folder
    plt.savefig(os.path.join(figure_folder, fname))
    plt.close()
    print('figure saved at: {}'.format(fname))
```

MonthlyCharge_bivariate_stacked histogram

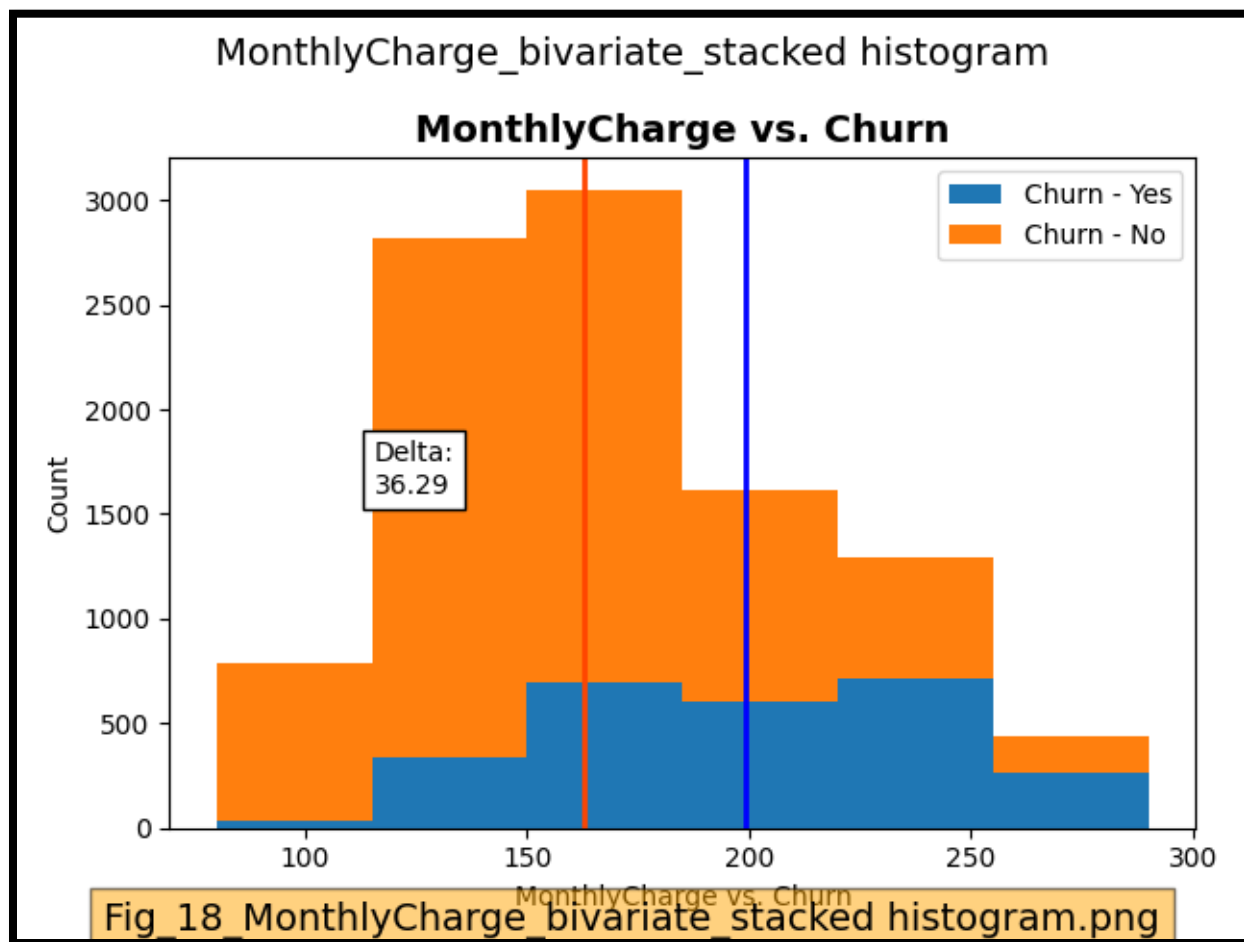


Figure 18 MonthlyCharge_bivariate_stacked histogram

Figure notes: the figure shows a blue line (Median of 'Churn' Yes) and an orange line (Median of 'Churn' No) to show a visualization of the difference between the two sub-populations. The difference between the two means is the Delta.

Here is the data that goes with this plot:

```
#####
MonthlyCharge crosstab
#####
Churn      No    Yes    All
MonthlyCharge
(79.769, 115.009]    755    35    790
(115.009, 150.039]  2477   338   2815
(150.039, 185.07]   2356   693   3049
(185.07, 220.1]     1009   608   1617
(220.1, 255.13]     579   715   1294
(255.13, 290.16]    174   261    435
All                 7350  2650  10000
```

Bandwidth_GB_Year_bivariate_stacked histogram

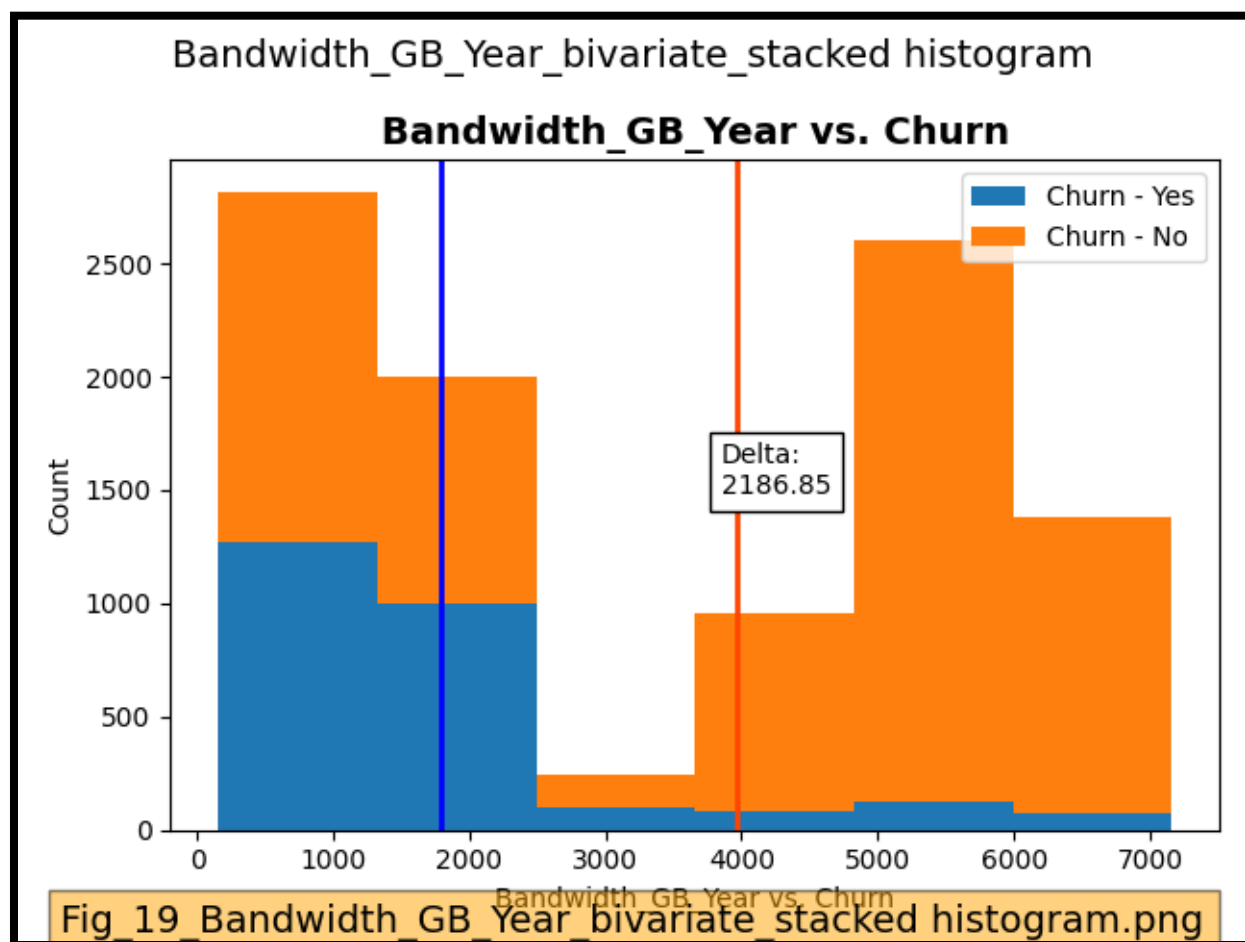


Figure 19 Bandwidth_GB_Year_bivariate_stacked histogram

Figure notes: the figure shows a blue line (Median of 'Churn' Yes) and an orange line (Median of 'Churn' No) to show a visualization of the difference between the two sub-populations. The difference between the two means is the Delta.

Here is the data that goes with this plot:

```
#####
      Bandwidth_GB_Year crosstab
#####
Churn      No      Yes      All
Bandwidth_GB_Year
(148.503, 1322.753]    1544    1273    2817
(1322.753, 2489.998]    1000     999    1999
(2489.998, 3657.244]     147     97     244
(3657.244, 4824.49]      874     82     956
(4824.49, 5991.736]    2476    125    2601
(5991.736, 7158.982]    1309     74    1383
All                   7350    2650   10000
```

The next plots are stacked bar plots of two categorical values:

- Fig_23_Gender_bivariate_stacked bar.png
- Fig_23_Gender_bivariate_stacked bar.png

Here is the code:

```
# visualization of selected bivariate data
bivariate = {
    "23": "Gender",
    "24": "InternetService",
}

for key in bivariate:
    c = bivariate[key]
    target = 'Churn'

    # print crosstab table associated with figure
    print(heading_toString(c + " crosstab"))
    print(pd.crosstab(df[c], df[target], margins=True))

    y = DataFrame({"count": df.groupby([c, target]).size()}).reset_index()

    """ each dataframe will look like this:
    Phone Churn  count
    0    No    No   1351
    1    No    Yes    521
    2    Yes   No   5999
    3    Yes   Yes   2129
    """

    x = y[c].unique()

    fig, ax = plt.subplots()
    no = y[y[target] == "No"]
    yes = y[y[target] == "Yes"]

    ax.barh(
        x,
        yes["count"],
        height=0.75,
        color="lightgreen",
        label="Churn Yes",
        left=no["count"],
    )
    ax.barh(x, no["count"], height=0.75, color="darkgreen", label="Churn No")
    ax.legend(["Churn - Yes", "Churn - No"])
    plt.xlabel('{} vs. {}'.format(c, 'Churn'))
    plt.ylabel('Count')
    count_figures = key
    sub_title = '{}_{}_{}'.format(c, 'bivariate', 'stacked bar')
    title = '{} vs. {}'.format(c, 'Churn')
    fname = 'Fig_{}_{}_{}'.format(str(count_figures), sub_title, "png")

    plt.title(title, fontsize=14, fontweight="bold")
    fig.suptitle(sub_title, fontsize=14)
    plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14, bbox={"facecolor": "orange", "alpha": 0.5, "pad": 5})
    fig.tight_layout()
```

```
# save file in the course figure's folder
plt.savefig(os.path.join(ffigure_folder, fname))
plt.close()
print('figure saved at: [{}].format(fname))
```

Gender_bivariate_stacked bar

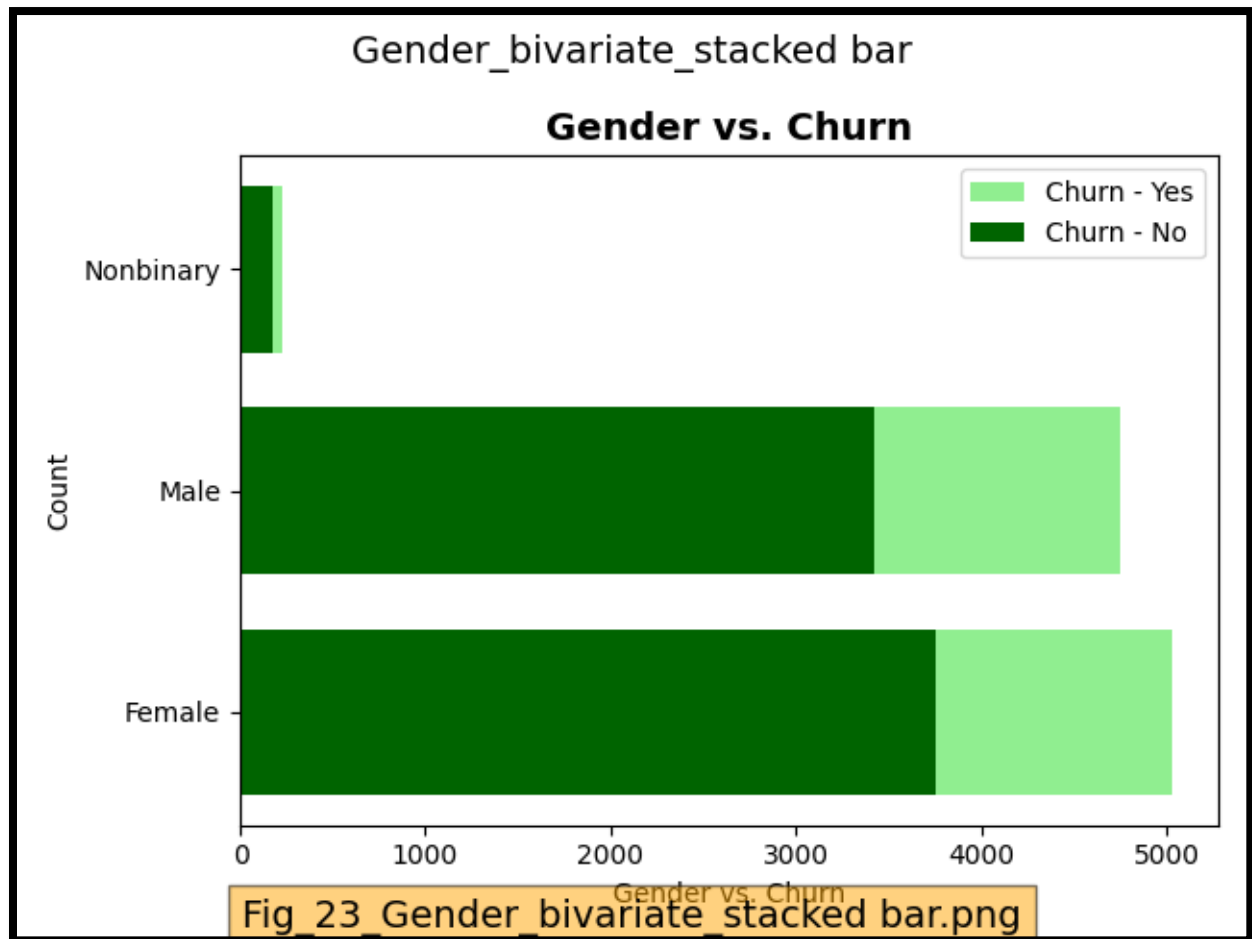


Figure 20 Gender_bivariate_stacked bar.png

Here is the data table for this plot:

```
#####
Gender crosstab
#####
Churn      No   Yes   All
Gender
Female     3753 1272 5025
Male       3425 1319 4744
Nonbinary   172   59   231
All        7350 2650 10000
```

InternetService_bivariate_stacked bar

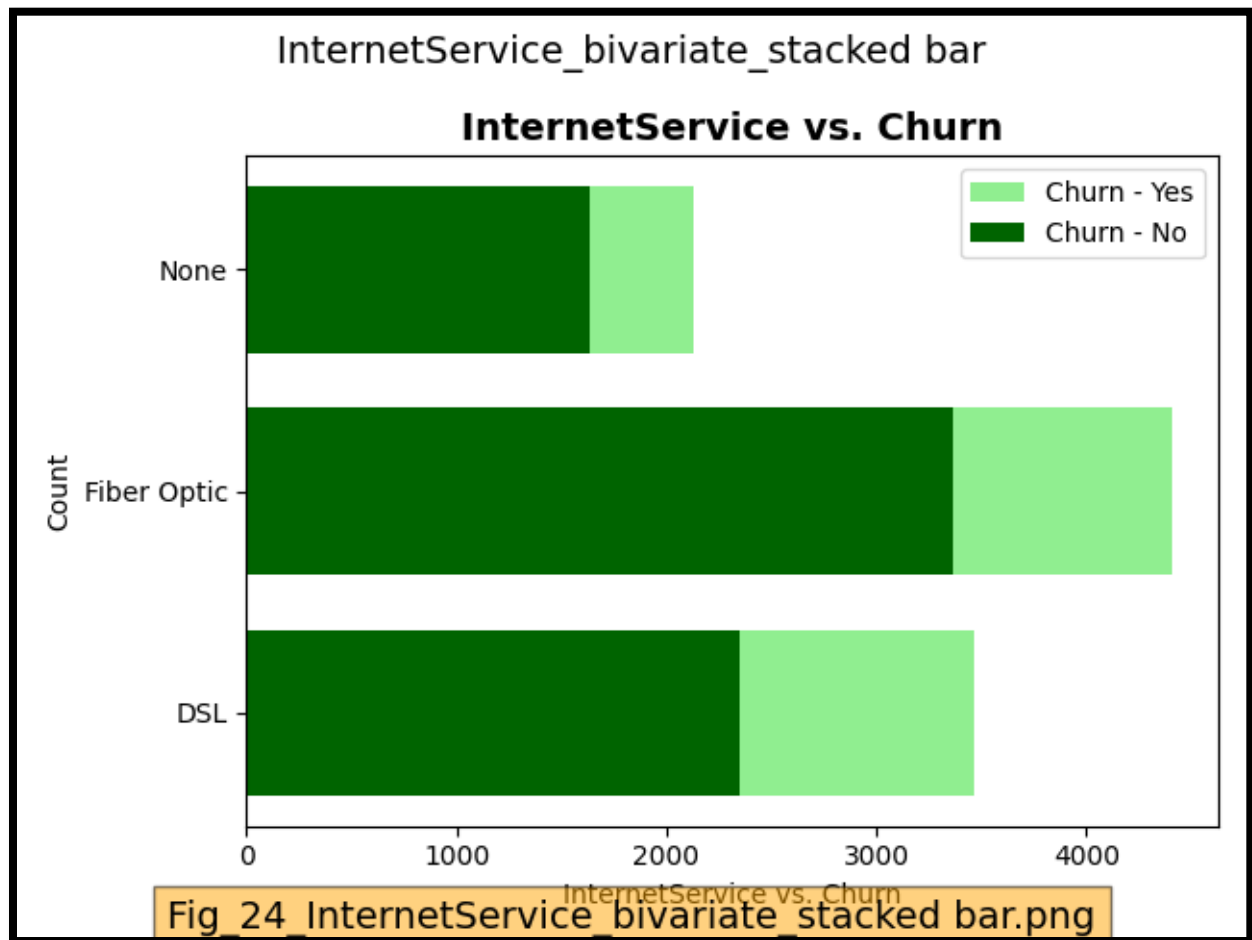


Figure 21 InternetService_bivariate_stacked bar.png

Here is the data table for this plot:

```
#####
InternetService crosstab
#####
Churn      No  Yes  All
InternetService
DSL        2349 1114 3463
Fiber Optic 3368 1040 4408
None       1633  496 2129
All        7350 2650 10000
```

Part E Summary

Part E1. Discuss the results of the test

The Chi-square hypothesis test was a very strong evidence of a relationship between 'Churn' and 'MonthlyCharge'.

Part E2. Discuss the limitations of your data analysis

- The test results do not explain the relationship, but only indicate possibility of relationship or association, further research is required.

Part E3. Recommend a course of action based on your results.

Recommend continued research to find extent of the relationship. Recommend organization consider modifying monthly charge rates for customers.

Part F Video

Part F – Submit video presentation. Video created and .MP4 file is attached to submission.

Part G References

Part G – References

Reference the web sources used to acquire segments of third-party code to support the analysis.

Brown, J. (2018, June 15). *A Gentle Introduction to the Chi-Squared Test for Machine Learning*. Retrieved from Machine Learning Mastery.com:
<https://machinelearningmastery.com/chi-squared-test-for-machine-learning/>

Bruce, P., Bruce, A., & Gedeck, P. (2020). *Practical Statistics for Data Scientists*. O'Reilly Media Inc.

Chi Squared Table. (2021, July 31). Retrieved from Statistics How To:
<https://www.statisticshowto.com/tables/chi-squared-table-right-tail/>

Contingency Tables. (2021, July 31). Retrieved from statsmodels.org:
https://www.statsmodels.org/stable/contingency_tables.html

Griffiths, D. (2009). *Head First Statistics*. O'Reilly Media Inc.

Hamel, G. (2021, July 31). *Python for Data 25: Chi-Squared Tests*. Retrieved from kaggle.com: <https://www.kaggle.com/hamelg/python-for-data-25-chi-squared-tests>

Larose, C. D., & Larose, D. T. (2019). *Data Science Using Python and R*. Wiley.

scipy.stats.chisquare. (2021, July 31). Retrieved from Scipy.org:
<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chisquare.html>

Sullivan, L. (2021, July 31). *Hypothesis Testing - Chi Squared Test*. Retrieved from sphweb.bumc.be.edu: https://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_HypothesisTesting-ChiSquare/BS704_HypothesisTesting-ChiSquare_print.html

(MAT21) <https://matplotlib.org/>

(MAT21) <https://math.meta.stackexchange.com/questions/21841/how-to-type-greater-than-or-equal-to-symbols?noredirect=1>

(NUM21) <https://numpy.org/>

(PAN21) <https://pandas.pydata.org/docs/pandas.pdf>

(PCA21) https://sebastianraschka.com/Articles/2014_pca_step_by_step.html

- (PCA21) <https://pub.towardsai.net/principal-component-analysis-pca-with-python-examplestutorial-67a917bae9aa>
- (PCA21) <https://www.districtdatalabs.com/principal-component-analysis-with-python>
- (PCA21) <https://www.edureka.co/blog/principal-component-analysis/>
- (PCA21) <https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-componentanalysis.html>
- (PCA21) <https://www.geeksforgeeks.org/principal-component-analysis-with-python/>
- (PCA21) <https://stackoverflow.com/questions/13224362/principal-component-analysis-pca-inpython12>
- (PCA21) <https://www.machinelearningplus.com/machine-learning/principal-componentsanalysis-pca-better-explained/>
- (SCI21) <https://www.scipy.org/>
- (STO21) Stoltenber, S. (2021, April 26). Styling a Jupyter Notebook. Retrieved from https://skelouse.github.io/styling_a_jupyter_notebook
- (TUK77) Tukey, J. W. (1977). Exploratory Data Analysis. Addison-Wesley Publishing Company. Retrieved from http://www.ru.ac.bd/wp-content/uploads/sites/25/2019/03/102_05_01_Tukey-Exploratory-Data-Analysis-1977.pdf
- (WIL19) Wilke, C. O. (2019). Fundamental of Data Visualization: A Primer on Making Informative and Compelling Figures. O'Reilly Media Inc.
- (WGU21) WGU.edu (2021, Jun 1). Data Files and Associated Dictionary Files. Retrieved from <https://access.wgu.edu/ASP3/aap/content/kgj47f8gj49f8du49d3k.html>

Part H Acknowledge Sources

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

Part I Code

The following is a complete list of all Python code:

```

1  '''
2  #####
3  IMPORT REQUIRED PACKAGES
4  ##### '''
5
6  # imports
7  import matplotlib.pyplot as plt
8  import numpy as np
9  import pandas as pd
10 from pandas.core.frame import DataFrame
11 import scipy.stats as stats
12 import seaborn as sns
13 from sklearn import preprocessing
14 from sklearn.decomposition import PCA
15 from scipy.stats import chi2_contingency
16 from scipy.stats import chi2
17 from IPython.display import Latex
18 import sys
19 import os
20 import datetime
21 from mm_lib import *
22
23 # remove the ellipsis from terminal pandas tables
24 pd.set_option("display.width", None)
25 np.set_printoptions(threshold=np.inf)
26
27 # reconfigure sys.stdout to utf-8
28 sys.stdout.reconfigure(encoding="utf-8")
29
30 '''
31 #####
32 STUDENT INFORMATION
33 ##### '''
34
35 # print student information
36 print(heading_toString("STUDENT INFORMATION"))
37 stud_info = {
38     "Student": "Mike Mattinson",
39     "Student ID": "001980761",
40     "Class": "D207 Exploratory Data Analysis",
41     "Dataset": "Churn",
42     "Submission": "2nd Submission",
43     "School": "WGU",
44     "Instructor": "Dr. Sewell"
45 }
46 for key in stud_info:
47     print('{:>14} | {}'.format(key, stud_info[key]))
48
49 print("Today is {}".format(datetime.date.today().strftime(
50     "%B %d, %Y")))
51
52 # print python environment
53 print(heading_toString("PYTHON ENVIRONMENT"))
54 print("Version: {} located at {}".format(sys.version, sys.
55     executable))
56
57 # global variables
58 count_tables = 0 # keep track of the number of tables gen
59     erated

```

```

57_   count_figures = 0 # keep track of the number of figures g
      enerated
58_   course = "d207"
59_   fig_title_fontsize = 18
60_   target = "Churn"
61_   title_str = "CHURN DATA - WGU - D207 - MATTINSON 2021"
62_
63_   # check and create folder to hold all figures
64_   figure_folder='figures\\' + course + '\\'
65_   if not os.path.exists(figure_folder):
66_       os.makedirs(figure_folder)
67_
68_   '''
69_   #####
70_       SETUP DATAFRAME
71_   ##### '''
72_
73_   # create dataframe
74_   print(heading_toString("DATAFRAME (DF)"))
75_   df = pd.read_csv("data\churn_clean.csv")
76_   print(df[["Customer_id", "City", "Churn", "MonthlyCharge",
      "Tenure"]].head(4))
77_   print(df.shape)
78_
79_   # remove unwanted columns
80_   print(heading_toString("REMOVE UNWANTED COLUMNS"))
81_
82_   # remove unwanted columns
83_   unwanted_columns = ["UID", "Interaction", "Lat", "Lng"]
84_
85_   for uc in unwanted_columns:
86_       if uc in df.columns:
87_           df.drop(columns=uc, inplace=True)
88_           print("[{}] column removed.".format(uc))
89_
90_   # show remaining columns
91_   print("Remaining columns: \n{}".format(df.columns))
92_
93_   print(heading_toString("CONTINUOUS DATA"))
94_   print(df.select_dtypes(include="float").info())
95_
96_   print(heading_toString("INTEGER DATA"))
97_   print(df.select_dtypes(include="integer").info())
98_
99_   print(heading_toString("CATEGORICAL DATA"))
100_  print(df.select_dtypes(include="object").info())
101_
102_  # output dataframe as .csv table
103_  print(heading_toString("SAVE CLEAN DATAFRAME"))
104_  count_tables += 1
105_  table_title = "churn_clean_data"
106_  fname = (
107_      "tables\\" + course + "\\" + "Tab_" + str(count_tables
108_  ) + "_" + table_title + ".csv"
109_  )
110_  print("table saved at: {}".format(fname))
111_  df.to_csv(fname, index=False, header=True)
112_
113_  '''
114_  #####
115_      PART B - CHI-SQUARE TEST
116_  ##### '''
117_  # CHI-SQUARE TEST

```

```

118__ target = 'Churn'
119__ prob = 0.999
120__ predictor = 'MonthlyCharge' # numerical
121__ print(heading_toString("PART B - CHI-
SQUARE INDEPENDENCE TEST - " + target + " vs. " + predictor))
122__ bins = 6
123__
124__ # use library function to perform test
125__ chi_square_analysis(target,predictor,bins,prob,df)
126__
127__ # use library function to print out distribution table
128__ print(heading_toString("CHI-SQUARE DISTRIBUTION TABLE"))
129__ print(chi_square_dist_table(0,0))
130__
131__ '''
132__ #####
133__ PART C - UNIVARIATE CATEOGRICAL
134__ ##### '''
135__
136__ print(heading_toString("PLOT UNIVARIATE CATEGORICAL"))
137__ # box plot of Churn (cat)
138__ #sns.countplot(df['InternetService'])
139__ #plt.show()
140__
141__ # visualization of selected categorical data
142__ univariate_categorical = {
143__     "1": "Churn",
144__     "2": "InternetService",
145__ }
146__
147__ for key in univariate_categorical:
148__     fig, ax = plt.subplots()
149__     c = univariate_categorical[key]
150__     sns.countplot(df[c])
151__     plt.xlabel(c)
152__     plt.ylabel('Count')
153__     count_figures = key
154__     #title = 'Hello world'
155__     sub_title = '{}_{}_{}'.format(c,'categorical','countpl
ot')
156__     title = '{}'.format(c)
157__     fname = 'Fig_{}_{}.{}'.format(str(count_figures), sub_
title, "png")
158__     plt.title(title, fontsize=14, fontweight="bold")
159__     fig.suptitle(sub_title, fontsize=14)
160__     plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14
, bbox={"facecolor":"orange", "alpha":0.5, "pad":5})
161__     fig.tight_layout()
162__
163__     # save file in the course figure's folder
164__     plt.savefig(os.path.join(folder, fname))
165__     plt.close()
166__     print('figure saved at: {}'.format(fname))
167__
168__ '''
169__ #####
170__ PART C - UNIVARIATE CONTINUOUS
171__ ##### '''
172__
173__ print(heading_toString("DESCRIPTIVE STATS"))
174__ # print descriptive stats for the dataframe
175__ print(df.select_dtypes(include="float").describe().round(2
))
176__

```

```

177_ print(heading_toString("PLOT UNIVARIATE CONTINUOUS"))
178_
179_ # visualization of selected univariate continuous data
180_ univariate_continuous = {
181_     "3": "MonthlyCharge",
182_     "4": "Income",
183_ }
184_
185_ for key in univariate_continuous:
186_     fig, ax = plt.subplots()
187_     c = univariate_continuous[key]
188_     plt.scatter(df.index, df[c])
189_     plt.xlabel(c)
190_     plt.ylabel('Count')
191_     count_figures = key
192_     #title = 'Hello world'
193_     sub_title = '{}_{}_{}'.format(c, 'continuous', 'scatterp
lot')
194_     title = '{}'.format(c)
195_     fname = 'Fig_{}_{}_{}'.format(str(count_figures), sub_
title, "png")
196_     plt.title(title, fontsize=14, fontweight="bold")
197_     fig.suptitle(sub_title, fontsize=14)
198_     plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14
, bbox={ "facecolor": "orange", "alpha": 0.5, "pad": 5 })
199_     fig.tight_layout()
200_
201_     # save file in the course figure's folder
202_     plt.savefig(os.path.join(figure_folder, fname))
203_     plt.close()
204_     print('figure saved at: [{}]' .format(fname))
205_
206_ '''
207_ #####
208_ PART D - BIVARIATE
209_ ##### '''
210_ print(heading_toString("PLOT BIVARIATE COUNTPLOT"))
211_ # visualization of selected bivariate data
212_ bivariate = {
213_     '11': 'InternetService',
214_     '12': 'TechSupport',
215_     '13': 'PaymentMethod',
216_     '14': 'Tablet',
217_     '15': 'Gender',
218_     '16': 'Port_modem',
219_     '17': 'Techie',
220_ }
221_
222_ for key in bivariate:
223_     fig, ax = plt.subplots()
224_     c = bivariate[key]
225_     sns.countplot(x='Churn', hue=c, data=df, palette='hls'
)
226_     plt.xlabel('{} vs. {}'.format(c, 'Churn'))
227_     plt.ylabel('Count')
228_     count_figures = key
229_     sub_title = '{}_{}_{}'.format(c, 'bivariate', 'countplot
')
230_     title = '{} vs. {}'.format(c, 'Churn')
231_     fname = 'Fig_{}_{}_{}'.format(str(count_figures), sub_
title, "png")
232_     plt.title(title, fontsize=14, fontweight="bold")

```



```

233_         fig.suptitle(sub_title, fontsize=14)
234_         plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14
, bbox={ "facecolor": "orange", "alpha": 0.5, "pad": 5 })
235_         fig.tight_layout()
236_
237_         # save file in the course figure's folder
238_         plt.savefig(os.path.join(ffigure_folder, fname))
239_         plt.close()
240_         print('figure saved at: [{}].format(fname))
241_
242_     print(heading_toString("PLOT BIVARIATE STACKED HISTOGRAM")
)
243_     # visualization of selected continous data
244_     bivariate = {
245_         '18': 'MonthlyCharge',
246_         '19': 'Bandwidth_GB_Year',
247_         '20': 'Tenure',
248_         '21': 'Outage_sec_perweek',
249_         '22': 'Income',
250_     }
251_
252_     for key in bivariate:
253_         c = bivariate[key]
254_         target = 'Churn'
255_
256_         # print crosstab table associated with figure
257_         print(heading_toString(c + " crosstab"))
258_         print(pd.crosstab(pd.cut(df[c], bins = 6), df[target],
margins=True))
259_
260_         df_yes = df[df.Churn == "Yes"][c]
261_         df_no = df[df.Churn == "No"][c]
262_         yes_mean = df_yes.mean()
263_         no_mean = df_no.mean()
264_         fig, ax = plt.subplots()
265_         _, bins, _ = ax.hist([df_yes, df_no], bins=6, stacked=
True)
266_         ax.legend(["Churn - Yes", "Churn - No"])
267_         ymin, ymax = ax.get_ylim()
268_         xmin, xmax = ax.get_xlim()
269_         ax.axvline(yes_mean, color="blue", lw=2) # yes mean
270_         ax.axvline(no_mean, color="orangered", lw=2) # no mea
n
271_         ax.text(
272_             (xmax - xmin) / 2,
273_             (ymax - ymin) / 2,
274_             "Delta:\n" + str(round(abs(yes_mean - no_mean), 2)
),
275_             bbox={ "facecolor": "white",
276_             }
277_         plt.xlabel('{} vs. {}'.format(c, 'Churn'))
278_         plt.ylabel('Count')
279_         count_figures = key
280_         sub_title = '{}_{}_{}'.format(c, 'bivariate', 'stacked h
istogram')
281_         title = '{} vs. {}'.format(c, 'Churn')
282_         fname = 'Fig_{}_{}_{}'.format(str(count_figures), sub_
title, "png")
283_         plt.title(title, fontsize=14, fontweight="bold")
284_         fig.suptitle(sub_title, fontsize=14)
285_         plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14
, bbox={ "facecolor": "orange", "alpha": 0.5, "pad": 5 })
286_         fig.tight_layout()
287_

```

```

288_     # save file in the course figure's folder
289_     plt.savefig(os.path.join(ffigure_folder, fname))
290_     plt.close()
291_     print('figure saved at: [{}].format(fname))
292_
293_     print(heading_toString("PLOT BIVARIATE STACKED BAR"))
294_     # visualization of selected bivariate data
295_     bivariate = {
296_         "23": "Gender",
297_         "24": "InternetService",
298_     }
299_
300_     for key in bivariate:
301_         c = bivariate[key]
302_         target = 'Churn'
303_
304_         # print crosstab table associated with figure
305_         print(heading_toString(c + " crosstab"))
306_         print(pd.crosstab(df[c], df[target], margins=True))
307_
308_         y = DataFrame({"count": df.groupby([c, target]).size()
309_         }).reset_index()
310_
311_         """ each dataframe will look like this:
312_         Phone Churn count
313_         0    No    No    1351
314_         1    No    Yes    521
315_         2    Yes   No    5999
316_         3    Yes   Yes    2129
317_         """
318_
319_         x = y[c].unique()
320_
321_         fig, ax = plt.subplots()
322_         no = y[y[target] == "No"]
323_         yes = y[y[target] == "Yes"]
324_
325_         ax.barh(
326_             x,
327_             yes["count"],
328_             height=0.75,
329_             color="lightgreen",
330_             label="Churn Yes",
331_             left=no["count"],
332_         )
333_         ax.barh(x, no["count"], height=0.75, color="darkgreen",
334_         , label="Churn No")
335_         ax.legend(["Churn - Yes", "Churn - No"])
336_         plt.xlabel('{} vs. {}'.format(c, 'Churn'))
337_         plt.ylabel('Count')
338_         count_figures = key
339_         sub_title = '{}_{}_{}'.format(c, 'bivariate', 'stacked bar')
340_         title = '{} vs. {}'.format(c, 'Churn')
341_         fname = 'Fig_{}_{}_{}'.format(str(count_figures), sub_
342_         title, "png")
343_         plt.title(title, fontsize=14, fontweight="bold")
344_         fig.suptitle(sub_title, fontsize=14)
345_         plt.figtext(0.5, 0.01, fname, ha="center", fontsize=14
346_         , bbox={"facecolor": "orange", "alpha": 0.5, "pad": 5})
347_         fig.tight_layout()
348_
349_         # save file in the course figure's folder

```

```

346__ plt.savefig(os.path.join(folder, fname))
347__ plt.close()
348__ print('figure saved at: {}'.format(fname))

```

Here is the mm_lib.py file:

```

1__ # consolidate all of my data analytics functions
2__ # to be reused in the diff courses.
3__ # Author: Mike Mattinson (MM)
4__ # Date: June 29, 2021
5__ #
6__ # To use, place following import in each course file:
7__ # from mm import *
8__
9__ def chi_square_analysis(target, predictor, bins, prob, df):
10__     """ Calculate and display results of chi-
11__         square independence test
12__
13__         Arg:
14__         target - target variable
15__         predictor - predictor variable
16__         bins - number of bins, if numerical, 0 if categorical
17__         prob - probability
18__         df - dataframe
19__
20__         Return: outputs analysis to terminal.
21__         ...
22__         import pandas as pd
23__         from scipy.stats import chi2_contingency
24__         from scipy.stats import chi2
25__         if bins == 0:
26__             # variable is categorical
27__             contingency = pd.crosstab(df[target], df[predictor])
28__         else:
29__             # variable is numerical
30__             mc_groups = pd.cut(df[predictor], bins=bins)
31__             contingency = pd.crosstab(df[target], mc_groups)
32__
33__         # print out the contingency table
34__         print(contingency.T)
35__
36__         stat, p, dof, expected = chi2_contingency(contingency)
37__         alpha = 1 - prob # significance level
38__         critical = chi2.ppf(prob, dof)
39__
40__         # interpret test-statistic
41__         if abs(stat) >= critical:
42__             print('At an alpha level of {}, the critical value
43__                 is {}.format(round(alpha,3),round(critical,3)))
44__             print('chi^2 = {} > {}, therefore, variables are d
45__                 ependent (reject H0).format(round(stat,3),round(critical,3))
46__             )
47__             #print('Also, we will need to perform additional a
48__                 nalysis to determine extent of the relationship.')
49__         else:
50__             print('chi^2 = {} <= {}, therefore, variables are
51__                 independent (fail to reject H0).format(round(stat,3),round(c
52__                 ritical,3)))
53__
54__

```

```

48_         print(' p-value: {:>8.3f}'.format(p))
49_         print('      dof: {:>8d}'.format(dof))
50_         print('    alpha: {:>8.3f}'.format(alpha))
51_         print('    stat: {:>8.3f}'.format(stat))
52_         print('critical: {:>8.3f}'.format(critical))
53_
54_     def chi_square_dist_table(dof,prob):
55_         '''create chi square distribution table
56_
57_         Args:
58_             dof - degree of freedom
59_             prob - significance level (alpha)
60_
61_         Returns:
62_             A string representation of the chi-square
63_             lower-left distribution table.
64_
65_         ...
66_         from scipy.stats import chi2
67_         prob_range = [ .10, 0.05, 0.025, 0.01, 0.001]
68_         dof_range = range(1,20)
69_         output = 'Lower-tail critical values of chi-
Square distribution:\n'.format()
70_
71_         # header row
72_         output += '{:>4}'.format('dof') # blank space for first col
73_         for p in prob_range:
74_             output += '{:>8.3f}'.format(p)
75_             output += '\n' + '-' * 47
76_
77_         # each row
78_         for d in dof_range:
79_             output += '\n{:4d}'.format(d)
80_             for p in prob_range:
81_                 temp = chi2.isf(p, d)
82_                 if d == dof and p == prob:
83_                     output += '{:>8.2f}'.format(temp)
84_                 else:
85_                     output += '{:>8.2f}'.format(temp)
86_
87_         return output
88_
89_     # heading_to_str
90_     def heading_toString(heading):
91_         """Create a heading string.
92_
93_         Args:
94_             heading: heading title.
95_
96_         Returns:
97_             The formatted heading as string.
98_
99_         """
100_         how_wide = len(heading) + 20
101_         underline_str = "#" * how_wide
102_
103_         # unicode support
104_         # must use sys.stdout.reconfigure(encoding="utf-8")
105_         overline_char = "#" # "\u203e"
106_
107_         overline_str = overline_char * how_wide
108_         temp = "\n\n{}".format(underline_str)

```

```

109_     temp += "\n\t{}".format(heading)
110_     temp += "\n{}".format(overline_str)
111_     return temp
112_
113_     # mitigate missing data
114_     def mitigate_missing_data(source, column, new_value, inplace=True):
115_         """Replace missing data with new_value and print message.
116_
117_         Args:
118_             source: source dataframe
119_             column: column to be adjusted.
120_             new_value: value to be inserted
121_             inplace (Default: True): change dataframe inplace
122_
123_         Returns:
124_             The success/failure message.
125_
126_         """
127_         try:
128_             source[column].fillna(new_value, inplace=inplace)
129_             print("{} missing data set to {} successfully.".format(column, new_value))
130_
131_         except:
132_             print("{} unable changes.".format(column))
133_
134_     def custom_lineplot(ax, x, y, error, xlims, ylims, color="red"):
135_         """Customized line plot with error bars."""
136_
137_         ax.errorbar(
138_             x,
139_             y,
140_             yerr=error,
141_             color=color,
142_             ls="--",
143_             marker="o",
144_             capsize=5,
145_             capthick=1,
146_             ecolor="black",
147_         )
148_
149_         ax.set_xlim(xlims)
150_         ax.set_ylim(ylims)
151_
152_         return ax
153_
154_     def custom_scatterplot(ax, x, y, error, xlims, ylims, color="green", markerscale=100):
155_         """Customized scatter plot where marker size is proportional to error measure."""
156_
157_         markersize = error * markerscale
158_
159_         ax.scatter(x, y, color=color, marker="o", s=markersize, alpha=0.5)
160_
161_         ax.set_xlim(xlims)
162_         ax.set_ylim(ylims)
163_
164_         return ax
165_

```

```

166__ import numpy as np
167__
168__ def custom_barchart(
169__     ax, x, y, error, xlims, ylims, error_kw, color="lightb
170__     lue", width=0.75
171__ ):
172__     """Customized bar chart with positive error bars only.
173__
174__     error = [np.zeros(len(error)), error]
175__
176__     ax.bar(
177__         x, y, color=color, width=width, yerr=error, error_
178__         kw=error_kw, align="center"
179__     )
180__
181__     ax.set_xlim(xlims)
182__     ax.set_ylim(ylims)
183__
184__     return ax
185__
186__ def custom_boxplot(ax, x, y, xlims, ylims, mediancolor="ma
187__     genta"):
188__     """Customized boxplot with solid black lines for box,
189__     whiskers, caps, and outliers."""
190__
191__     medianprops = {"color": mediancolor, "linewidth": 2}
192__     boxprops = {"color": "black", "linestyle": "-"}
193__     whiskerprops = {"color": "black", "linestyle": "-"}
194__     capprops = {"color": "black", "linestyle": "-"}
195__     flierprops = {"color": "black", "marker": "x"}
196__
197__     ax.boxplot(
198__         y,
199__         positions=x,
200__         medianprops=medianprops,
201__         boxprops=boxprops,
202__         whiskerprops=whiskerprops,
203__         capprops=capprops,
204__         flierprops=flierprops,
205__     )
206__
207__     ax.set_xlim(xlims)
208__     ax.set_ylim(ylims)
209__
210__     return ax
211__
212__ def fig_title_toString(title, fig_count, fig_type):
213__     return "Fig_" + str(fig_count) + "_" + title + "_" + f
214__     ig_type
215__
216__ def table_title_toString(title, table_count):
217__     return "Tab_" + str(table_count) + "_" + title
218__
219__ def fig_fname_toString(folder, title, filetype):
220__     return folder + title + "." + filetype
221__
222__ def table_fname_toString(course, title, filetype):
223__     return "tables\\" + course + "\\" + title + "." + file
224__     type
225__
226__ def numeric_data_toBoxplot():
227__     return True

```

Part J Output

The following is a complete list of all terminal output:

```
#####
      STUDENT INFORMATION
#####
      Student | Mike Mattinson
      Student ID | 001980761
      Class | D207 Exploratory Data Analysis
      Dataset | Churn
      Submission | 2nd Submission
      School | WGU
      Instructor | Dr. Sewell
      Today is August 03, 2021

#####
      PYTHON ENVIRONMENT
#####
      Version: 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21)
      [MSC v.1929 64 bit (AMD64)] located at P:\workspace-
      wgu\wgu_local_py\wgu_venu\Scripts\python.exe

#####
      DATAFRAME (DF)
#####
      Customer_id      City Churn  MonthlyCharge      Tenure
0      K409198  Point Baker    No      172.455519      6.795513
1      S120509  West Branch   Yes      242.632554      1.156681
2      K191035    Yamhill     No      159.947583      15.754144
3      D90850    Del Mar      No      119.956840      17.087227
(10000, 50)

#####
      REMOVE UNWANTED COLUMNS
#####
      [UID] column removed.
      [Interaction] column removed.
      [Lat] column removed.
      [Lng] column removed.
      Remaining columns:
      Index(['CaseOrder', 'Customer_id', 'City', 'State', 'County',
      'Zip',
      'Population', 'Area', 'TimeZone', 'Job', 'Children',
      'Age', 'Income',
      'Marital', 'Gender', 'Churn', 'Outage_sec_perweek',
      'Email', 'Contacts',
      'Yearly equip_failure', 'Techie', 'Contract',
      'Port_modem', 'Tablet',
      'InternetService', 'Phone', 'Multiple',
      'OnlineSecurity',
      'OnlineBackup', 'DeviceProtection', 'TechSupport',
      'StreamingTV',
      'StreamingMovies', 'PaperlessBilling',
      'PaymentMethod', 'Tenure',
      'MonthlyCharge', 'Bandwidth_GB_Year', 'Item1',
      'Item2', 'Item3',
      'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

```
#####
CONTINUOUS DATA
#####
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 5 columns):
#   Column                      Non-Null Count  Dtype
---  ---
0   Income                      10000 non-null  float64
1   Outage_sec_perweek          10000 non-null  float64
2   Tenure                      10000 non-null  float64
3   MonthlyCharge               10000 non-null  float64
4   Bandwidth_GB_Year           10000 non-null  float64
dtypes: float64(5)
memory usage: 390.8 KB
None
```

```
#####
INTEGER DATA
#####
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 16 columns):
#   Column                      Non-Null Count  Dtype
---  ---
0   CaseOrder                   10000 non-null  int64
1   Zip                         10000 non-null  int64
2   Population                  10000 non-null  int64
3   Children                    10000 non-null  int64
4   Age                         10000 non-null  int64
5   Email                       10000 non-null  int64
6   Contacts                    10000 non-null  int64
7   Yearly_equip_failure        10000 non-null  int64
8   Item1                       10000 non-null  int64
9   Item2                       10000 non-null  int64
10  Item3                       10000 non-null  int64
11  Item4                       10000 non-null  int64
12  Item5                       10000 non-null  int64
13  Item6                       10000 non-null  int64
14  Item7                       10000 non-null  int64
15  Item8                       10000 non-null  int64
dtypes: int64(16)
memory usage: 1.2 MB
None
```

```
#####
CATEGORICAL DATA
#####
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 25 columns):
#   Column                      Non-Null Count  Dtype
---  ---
0   Customer_id                 10000 non-null  object
1   City                        10000 non-null  object
2   State                       10000 non-null  object
3   County                     10000 non-null  object
4   Area                       10000 non-null  object
```



```

5   TimeZone          10000 non-null object
6   Job                10000 non-null object
7   Marital            10000 non-null object
8   Gender             10000 non-null object
9   Churn              10000 non-null object
10  Techie             10000 non-null object
11  Contract           10000 non-null object
12  Port_modem         10000 non-null object
13  Tablet             10000 non-null object
14  InternetService    10000 non-null object
15  Phone              10000 non-null object
16  Multiple           10000 non-null object
17  OnlineSecurity     10000 non-null object
18  OnlineBackup       10000 non-null object
19  DeviceProtection   10000 non-null object
20  TechSupport        10000 non-null object
21  StreamingTV        10000 non-null object
22  StreamingMovies    10000 non-null object
23  PaperlessBilling   10000 non-null object
24  PaymentMethod      10000 non-null object

```

```

dtypes: object(25)
memory usage: 1.9+ MB
None

```

```
#####
```

```
SAVE CLEAN DATAFRAME
```

```
#####
```

```
table saved at: tables\d207\Tab_1_churn_clean_data.csv
```

```
#####
```

```
#####
```

```
PART B - CHI-SQUARE INDEPENDENCE TEST - Churn vs.
```

```
MonthlyCharge
```

```
#####
```

```
#####
```

```
Churn                No   Yes
```

```
MonthlyCharge
```

```
(79.769, 115.009]      755   35
```

```
(115.009, 150.039]    2477  338
```

```
(150.039, 185.07]     2356  693
```

```
(185.07, 220.1]       1009  608
```

```
(220.1, 255.13]       579   715
```

```
(255.13, 290.16]      174   261
```

```
At an alpha level of 0.001, the critical value is 20.515.
```

```
chi^2 = 1425.642 > 20.515, therefore, variables are dependent (reject H0).
```

```
p-value: 0.000
```

```
dof: 5
```

```
alpha: 0.001
```

```
stat: 1425.642
```

```
critical: 20.515
```

```
#####
```

```
CHI-SQUARE DISTRIBUTION TABLE
```

```
#####
```

```
Lower-tail critical values of chi-Square distribution:
```

```
dof   0.100   0.050   0.025   0.010   0.001
```

```
-----
```

```
1      2.71    3.84    5.02    6.63   10.83
```

2	4.61	5.99	7.38	9.21	13.82
3	6.25	7.81	9.35	11.34	16.27
4	7.78	9.49	11.14	13.28	18.47
5	9.24	11.07	12.83	15.09	20.52
6	10.64	12.59	14.45	16.81	22.46
7	12.02	14.07	16.01	18.48	24.32
8	13.36	15.51	17.53	20.09	26.12
9	14.68	16.92	19.02	21.67	27.88
10	15.99	18.31	20.48	23.21	29.59
11	17.28	19.68	21.92	24.72	31.26
12	18.55	21.03	23.34	26.22	32.91
13	19.81	22.36	24.74	27.69	34.53
14	21.06	23.68	26.12	29.14	36.12
15	22.31	25.00	27.49	30.58	37.70
16	23.54	26.30	28.85	32.00	39.25
17	24.77	27.59	30.19	33.41	40.79
18	25.99	28.87	31.53	34.81	42.31
19	27.20	30.14	32.85	36.19	43.82

```
#####
```

```
    PLOT UNIVARIATE CATEGORICAL
```

```
#####
```

```
P:\workspace-wgu\wgu_local_py\wgu_venu\lib\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will
result in an error or misinterpretation.
```

```
    warnings.warn(
```

```
figure saved at: [Fig_1_Churn_categorical_countplot.png]
```

```
P:\workspace-wgu\wgu_local_py\wgu_venu\lib\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will
result in an error or misinterpretation.
```

```
    warnings.warn(
```

```
figure saved at:
```

```
[Fig_2_InternetService_categorical_countplot.png]
```

```
#####
```

```
    DESCRIPTIVE STATS
```

```
#####
```

	Income	Outage_sec_perweek	Tenure	MonthlyCharge
Bandwidth_GB_Year				
count	10000.00	10000.00	10000.00	10000.00
mean	39806.93	10.00	34.53	172.62
std	28199.92	2.98	26.44	42.94
min	348.67	0.10	1.00	79.98
25%	19224.72	8.02	7.92	139.98
50%	33170.60	10.02	35.43	167.48
75%	53246.17	11.97	61.48	200.73
max	5586.14			

```

max      258900.70      21.21      72.00      290.16
7158.98

#####
      PLOT UNIVARIATE CONTINUOUS
#####
figure saved at:
[[Fig_3_MonthlyCharge_continuous_scatterplot.png]]
figure saved at: [[Fig_4_Income_continuous_scatterplot.png]]

#####
      PLOT BIVARIATE COUNTPLOT
#####
figure saved at:
[[Fig_11_InternetService_bivariate_countplot.png]]
figure saved at: [[Fig_12_TechSupport_bivariate_countplot.png]]
figure saved at:
[[Fig_13_PaymentMethod_bivariate_countplot.png]]
figure saved at: [[Fig_14_Tablet_bivariate_countplot.png]]
figure saved at: [[Fig_15_Gender_bivariate_countplot.png]]
figure saved at: [[Fig_16_Port_modem_bivariate_countplot.png]]
figure saved at: [[Fig_17_Techie_bivariate_countplot.png]]

#####
      PLOT BIVARIATE STACKED HISTOGRAM
#####

#####
      MonthlyCharge crosstab
#####
Churn      No      Yes      All
MonthlyCharge
(79.769, 115.009]    755      35      790
(115.009, 150.039]  2477     338     2815
(150.039, 185.07]   2356     693     3049
(185.07, 220.1]     1009     608     1617
(220.1, 255.13]     579      715     1294
(255.13, 290.16]    174      261      435
All                7350     2650    10000
figure saved at: [[Fig_18_MonthlyCharge_bivariate_stacked
histogram.png]]

#####
      Bandwidth_GB_Year crosstab
#####
Churn      No      Yes      All
Bandwidth_GB_Year
(148.503, 1322.753]  1544     1273     2817
(1322.753, 2489.998] 1000      999     1999
(2489.998, 3657.244]  147       97       244
(3657.244, 4824.49]   874       82       956
(4824.49, 5991.736]   2476     125     2601
(5991.736, 7158.982]  1309       74     1383
All                7350     2650    10000
figure saved at: [[Fig_19_Bandwidth_GB_Year_bivariate_stacked
histogram.png]]

```

```
#####
Tenure crosstab
#####
Churn      No    Yes    All
Tenure
(0.929, 12.833]    1876  1924  3800
(12.833, 24.667]    692   418  1110
(24.667, 36.5]      70    28   98
(36.5, 48.333]     511   83   594
(48.333, 60.166]   1526  105  1631
(60.166, 71.999]   2675   92  2767
All                7350  2650  10000
figure saved at: [Fig_20_Tenure_bivariate_stacked
histogram.png]

#####
Outage_sec_perweek crosstab
#####
Churn      No    Yes    All
Outage_sec_perweek
(0.0786, 3.618]    132   43   175
(3.618, 7.136]    1091  397  1488
(7.136, 10.653]   3086  1137  4223
(10.653, 14.171]  2447  855  3302
(14.171, 17.689]  557   200  757
(17.689, 21.207]   37    18   55
All                7350  2650  10000
figure saved at: [Fig_21_Outage_sec_perweek_bivariate_stacked
histogram.png]

#####
Income crosstab
#####
Churn      No    Yes    All
Income
(90.118, 43440.675]  4739  1682  6421
(43440.675, 86532.68] 2121   792  2913
(86532.68, 129624.685] 411   138   549
(129624.685, 172716.69] 70    34   104
(172716.69, 215808.695] 5     4     9
(215808.695, 258900.7] 4     0     4
All                7350  2650  10000
figure saved at: [Fig_22_Income_bivariate_stacked
histogram.png]

#####
PLOT BIVARIATE STACKED BAR
#####

#####
Gender crosstab
#####
Churn      No    Yes    All
Gender
Female     3753  1272  5025
Male       3425  1319  4744
Nonbinary   172    59   231
```

```
All          7350  2650  10000
figure saved at: [Fig_23_Gender_bivariate_stacked bar.png]
```

```
#####
```

```
InternetService crosstab
```

```
#####
```

```
Churn          No    Yes    All
```

```
InternetService
```

```
DSL           2349  1114   3463
```

```
Fiber Optic   3368  1040   4408
```

```
None          1633    496   2129
```

```
All           7350  2650  10000
```

```
figure saved at: [Fig_24_InternetService_bivariate_stacked
bar.png]
```

D207 Mattinson (2nd Submission)

Part K Video Script

Hello, my name is Mike Mattinson.

This is a video walk-through of my 2nd submission of the D207 final assessment.

The D207 class is “Exploratory Data Analysis”, WGU.

My instructor is Dr. Sewell.

I selected the ‘Churn’ dataset.

All of the code was written in Python and executed in Visual Studio Code.

The version of Python is 3.9.6.

The primary list of Python packages used:

- Pandas
- Numpy
- Matplotlib

I also created some helper functions located in the mm_lib.py file.

Lines 64-65: Used to create a folder in the working directory to contain the plot figures generated by the code. Some of the data was removed (Lines 79-91)

Lines 75: Used pandas to create dataframe from the provided .csv data file.

Lines 93-97: Created a list of numerical data.

Lines 99-100: Created a list of categorical data.

Lines 113-129: Part B – Chi-Square Independence test. The analysis code is located in the mm_lib.py file. The code performs the independence test and then uses logic to print out the results of the test.

Lines 132-166: Part C – The first section is the code to generate the univariate categorical plots.

Lines 169-204: Part C – The second section is the code to generate the univariate continuous plots.

Lines 207-348: Part D – The bivariate plots are executed by creating three different types of plots.

All code is executed without error.

Open file explorer and show the plot files.

This concludes my presentation.