D208 – Assessment NBM2 TASK 1: MULTIPLE REGRESSION FOR PREDICTIVE MODELING

PART I – RESEARCH QUESTION

PART A1 – SUMMARIZE ONE RESEARCH QUESTION THAT IS RELEVANT TO A REAL WORLD ORGANIZATIONAL SITUATION CAPTURED IN THE DATASET THAT I HAVE SELECTED AND I WILL ANSWER USING MULTIPLE REGRESSION

Question: A telecommunication company has requested a profile prediction of customers who are most likely to terminate their contract, churn. Since maintaining a customer is much cheaper than acquiring a new one, the analyst will have to review the customer data provided to identify which variables would predict customer churn the best. With that information, the telecommunication company will be able to create an effective plan to avoid losing current customers.

PART A2 – GOALS OF THE DATA ANALYSIS

The main goal of this analysis is to determine what is the customer profile who are more likely to churn and with this information create a machine learning algorithm to predict customer churn.

PART II – METHOD JUSTIFICATION

PART B1 – SUMMARIZE THE ASSUMPTIONS OF A MULTIPLE REGRESSION MODEL

1-) There must be a linear relationship in between the outcome variable and the independent variables.

2-) The independent variables cannot be dependent of each other, meaning no or very little multicollinearity can happen

3-) Variables should be in a normal distribution

4-) Residuals should be independent and equally distributed along the regression line

PART B2 – DESCRIBE THE BENEFITS OF USING THE TOOL YOU HAVE CHOSEN IN SUPPORT OF VARIOUS PHASES OF THE ANALYSIS

I have started D206 using PYTHON and I intend to continue using this free tool. PYTHON ahs been largely used in DATA ANALYTICS projects. As I stated in my previous report on D206 “PyCharm is an integrated development environment used in computer programming, specifically for the Python language”. It is a free and powerful language. Python is able to read Excel files, clean untidy data, manage missing values and more. I have done all kinds of graphs, from pie charts and histograms to scatter plots in Python, it also offers packages to perform

PCA and Factor analysis, Regression and Decision tree modeling to generate a prediction function.

PART B3 – EXPLAIN WHY MULTIPLE REGRESSION IS AN APPROPRIATE TECHNIQUE TO ANALYZE THE RESEARCH QUESTION SUMMARIZED IN PART I

Multiple regression is used when there is the need to analyze the relationship between the dependent variable, our case is churn, and the predictor variables and the importance of each predictor on the churn.

PART III – DATA PREPARATION

PART C – SUMMARIZE THE DATA PREP PROCESS FOR MULTIPLE REGRESSION ANALYSIS BY DOING THE FOLLOWING:

PART C1 – DESCRIBE YOUR DATA PREP GOALS AND THE DATA MANIPULATIONS THAT WILL BE USED TO ACHIEVE THE GOALS

Starting in D206, I worked with the "churn dataset". The first step was to drop some variables presented in the original dataset that would not help to predict customer churn. Those variables dropped were: CaseOrder, Interaction, City, State, County, Zip, Lat, Lng and Population and also the very first column with only numbers and no description was also dropped at this first step. In addition to these columns, I am deleting the variables "TimeZone", "Job" and "Customer_ID".

Second step was to rename the last 8 columns with the appropriate customer survey label:

```
#Start cleaning up the data, getting rid of irrelevant data
df = churn_df.drop(churn_df.columns[0], axis = 1)
df.head()
print(df)
```

```
#Removing the 9 columns: CaseOrder, Interaction, City, Zip, Lat and Lng
df_clean = df.drop(columns=['CaseOrder', 'Interaction', 'City', 'State',
'County', 'Zip', 'Lat', 'Lng', 'Population'])
print(df_clean)
```

```
#Renaming the last 8 survey columns for a more descriptive value
df_clean.rename(columns = {'item1':'CS Responses', 'item2':'CS Fixes',
'item3':'CS Replacements', 'item4':'CS Reliability', 'item5':'CS Options',
'item6':'CS Respectfulness', 'item7':'CS Courteous', 'item8':'CS Listening'},
inplace=True)
print(df_clean)
```

These steps are not part of my current code since I have been working with the clean churn data set.

We have numerical and categorical variables in the dataset. Mathematical models can not be done with categorical variables so for the ones with two levels of category, YES and NO, I converted them to 1 and 0, respectively.

```
#Changing Variables from NO/YES to 0/1
churn_df2.Churn.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Phone.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.PaperlessBilling.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Techie.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Port_modem.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Tablet.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.Multiple.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.OnlineSecurity.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.OnlineBackup.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.DeviceProtection.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.TechSupport.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.StreamingTV.replace({"Yes":1, "No":0}, inplace = True)
churn_df2.StreamingMovies.replace({"Yes":1, "No":0}, inplace = True)
```

For categorical variables with 3 levels or more, I created dummy variables and later concatenated both data sets and dropped the original variables:

```
#Creating a dummy variable for some of the categorical variables with 3 or
more levels
dummy1 = pd.get_dummies(churn_df2, columns=['Marital', 'Contract',
'PaymentMethod', 'Gender', 'InternetService', 'Area',
                                            'Employment'])
#Adding the results to the master dataframe
churn_df2 = pd.concat([churn_df2, dummy1], axis=1)

#We have created dummies for the below variables, so we can drop them
churn_df2 = churn_df2.drop(['Marital', 'Contract', 'PaymentMethod', 'Gender',
'InternetService', 'Area', 'Employment'], 1)
```

Before attempting to create any model we have to normalize the data:

```
# Normalize the data
churn_normalized = preprocessing.normalize(churn_df2)
print("Normalized Data = ", churn_normalized)
```

```
None
Normalized Data =  [[2.47439682e-05 1.68258984e-03 7.06737219e-01 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [3.25538225e-05 8.78953209e-04 7.06580718e-01 ... 3.25538225e-05
  0.00000000e+00 0.00000000e+00]
 [8.50629392e-05 1.06328674e-03 7.05745941e-01 ... 0.00000000e+00
  2.12657348e-05 0.00000000e+00]
 ...
 [2.11410639e-05 1.12047639e-03 7.01608488e-01 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [3.95452732e-05 1.54226565e-03 6.59140613e-01 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [2.09818789e-05 5.87492609e-04 6.96325615e-01 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]]
            Tenure        Tenure  MonthlyCharge  MonthlyCharge
```

Picture 1: Data Normalization

PART C2 – DISCUSS THE SUMMARY STATS, INCLUDING THE TARGET VARIABLE AND ALL PREDICTOR VARIABLES THAT YOU WILL NEED TO GATHER FROM THE DATA SET TO ANSWER THE RESEARCH QUESTION

```
# Loading the Clean Churn Dataset from D206
df = pd.read_csv('/Users/bia/Desktop/churn_clean.csv')
df_stats = df.describe()
# print(df_stats)
```

```
          Children          Age  ...     Courteous     Listening
count  7505.000000  7525.000000  ...  10000.000000  10000.000000
mean      2.095936    53.275748  ...      3.509500      3.495600
std       2.154758    20.753928  ...      1.028502      1.028633
min       0.000000    18.000000  ...      1.000000      1.000000
25%       0.000000    35.000000  ...      3.000000      3.000000
50%       1.000000    53.000000  ...      4.000000      3.000000
75%       3.000000    71.000000  ...      4.000000      4.000000
max      10.000000    89.000000  ...      7.000000      8.000000

[8 rows x 18 columns]
```

Picture 2: Summary Stats of the data

Calculating the churn rate:

```
#Calculating the Churn Rate
churn_rate=df.Churn.value_counts() / len(df)
print(churn_rate)
```

Picture 3: Churn Rate

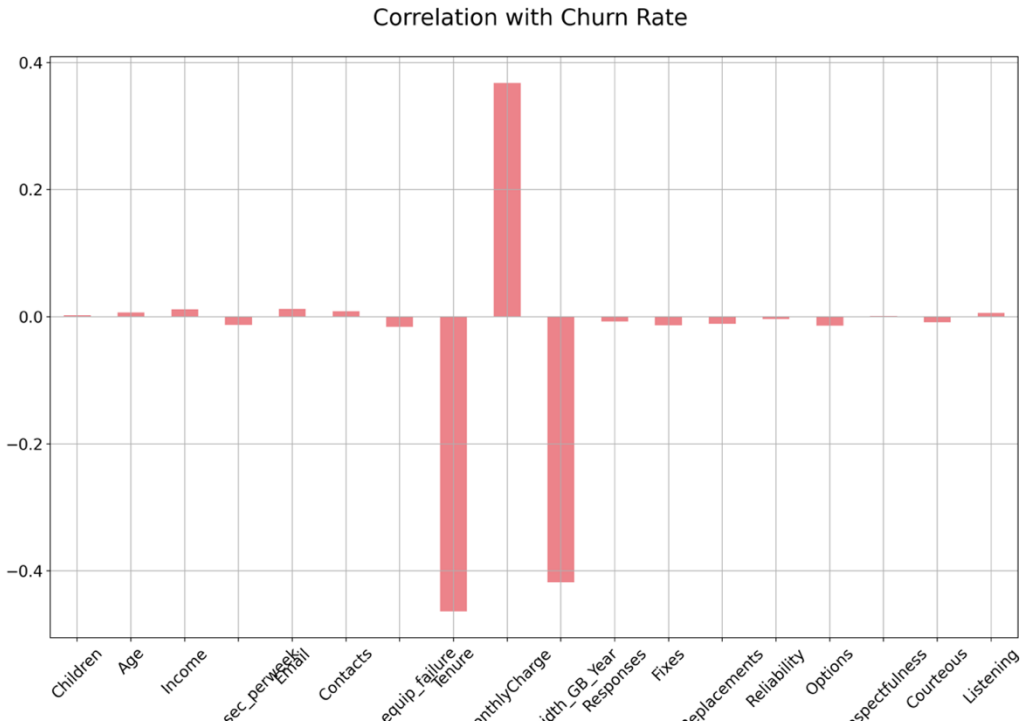
We can clearly see that the data provided is not equally distributed. We have the majority of cases of customers who haven't churned (73.5%) and only **26.5%** of total customers have churned. Ideally, we would have an equal ratio of cases to assure a good fit for the model prediction of churn rate.

From my reports delivered in **D206** & **D207**, I could find some relationship in between these variables and churn rate:

Num: Tenure, Bandwidth_GB_Year, MonthlyCharge,
Cat: StreamingTV, StreamingMovies, Multiple, InternetService and Contract

For the numerical variables and categorical with 2 or less values that I had to encode in D206, I generated a correlation plot so I could visualize which variables had a greater influence on the churn rate. I found that Bandwidth, Tenure, MonthlyCharge, StreamingTV, StreamingMovies and Multiple were the key variables. I also included InternetService because from the bar plot against churn rate, we can clearly see that customers with DSL internet churn more and Contract because customers on a month-to-month clearly are more likely to churn.

```
#Correclation Matrix for Numeric and label encoded columns
sns.heatmap(churn_df2.corr(), annot=True)
plt.show()
```

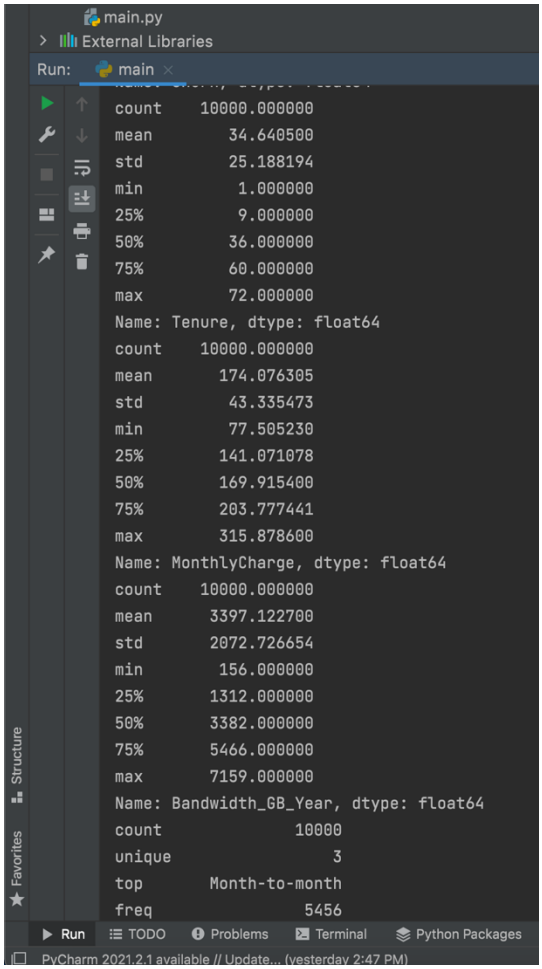


Picture 4: Correlation Plot

Churn	1.000000
MonthlyCharge	**0.367495**
StreamingMovies	**0.289262**
StreamingTV	**0.230151**
Multiple	**0.131771**
DeviceProtection	0.056489
OnlineBackup	0.050508
Email	0.012326
Income	0.011542
Contacts	0.008567
Port_modem	0.008157
PaperlessBilling	0.007030
Age	0.006131
Listening	0.005653
Children	0.002397
Respectfulness	0.001130
Tablet	-0.002779
Reliability	-0.003396
Responses	-0.007341
Courteous	-0.008851
Replacements	-0.011143
Outage_sec_perweek	-0.012813

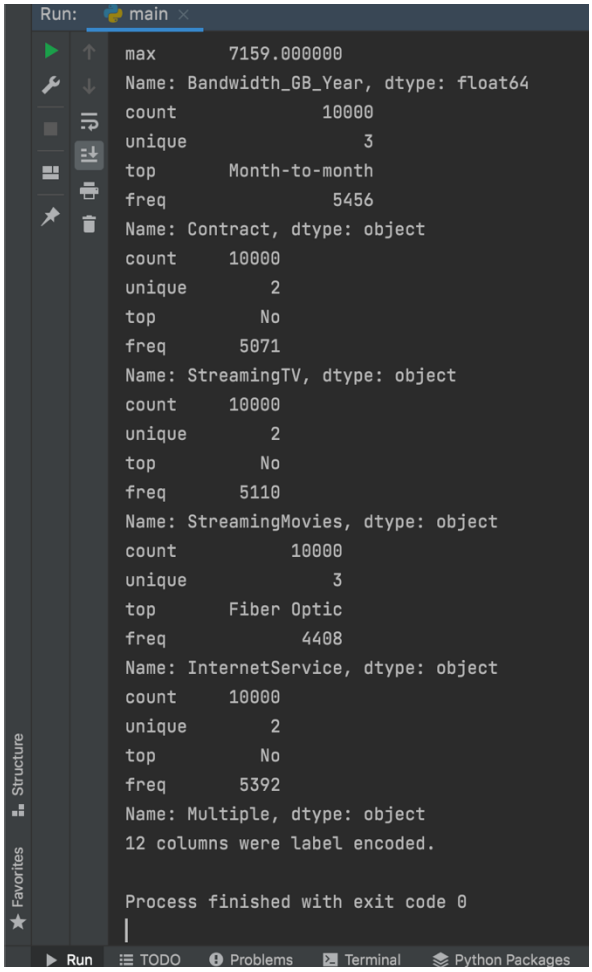
Fixes	-0.013253
OnlineSecurity	-0.013540
Options	-0.013971
Yearly_equip_failure	-0.015927
Bandwidth_GB_Year	**-0.418047**
Tenure	**-0.463720**

```
#Basic Stats of important variables

#Tenure
df_stats_tenure = churn_df['Tenure'].describe()
print(df_stats_tenure)

#Monthly Charges
df_stats_charge = churn_df['MonthlyCharge'].describe()
print(df_stats_charge)

#Bandwidth
df_stats_band = churn_df['Bandwidth_GB_Year'].describe()
print(df_stats_band)

#Contract
df_stats_contract = churn_df['Contract'].describe()
print(df_stats_contract)

#Streaming TV
df_stats_sttv = churn_df['StreamingTV'].describe()
print(df_stats_sttv)

#Straeming Movies
df_stats_stmovies = churn_df['StreamingMovies'].describe()
print(df_stats_stmovies)

#Internet
df_stats_internet = churn_df['InternetService'].describe()
print(df_stats_internet)

#Multiple
df_stats_multiple = churn_df['Multiple'].describe()
print(df_stats_multiple)
```



Picture 5: Displaying Basic Stats I

Tenure: we notice that the average customer is retained for an average of 34 months with a monthly charge average of $174.



Picture 6: Displaying Basic Stats II

Most customers are on a month-to-month contract. Which I have already explained in D207 that it's not good for customer retention. We will also see in sections to come in this report that customers with a month-to-month contract churn more than customers with longer contracts.

PART C3 – EXPLAIN THE STEPS USED TO PREPARE THE DATA FOR THE ANALYSIS, INCLUDING THE ANNOTATED CODE

The dataset has is originally presented with missing values. I replaced numerical missing values with the median of each column and categorical missing values with the mode of each variable (the most common value).

Searching for missing values:

```
null_values = df_clean.isna().any()
print(null_values)
```

Numerical:

```
#Filling the missing data with the median of each variable
#We saw that the columns Children, Age, Income, Tenure and Bandwidth_GB_Year
have missing values

na_cols = df_clean.isna().any()
na_cols = na_cols[na_cols == True].reset_index()
na_cols = na_cols["index"].tolist()
for col in df_clean.columns[1:]:
    if col in na_cols:
        if df_clean[col].dtype != 'object':
            df_clean[col] =
df_clean[col].fillna(df_clean[col].median()).round(0)
```

Categorical:
PHONE → YES
TECHIE → NO

```
#Phone and Techie Columns are categorical with missing values as well
print(df_clean['Phone'].unique())
print(df_clean['Techie'].unique())
#Since We have "YES" "NO" and "NAN" we will need to replace the nan values
for something
df_stats_phone = df_clean['Phone'].describe()
print(df_stats_phone)

df_stats_phone = df_clean['Techie'].describe()
print(df_stats_phone)

#Since these are categorical columns, I am going to replace the "NAN" values
for whatever shows more
#Phone --> "YES"
#Techie --> "NO"

df_clean = df_clean.fillna(df.mode().iloc[0])
```

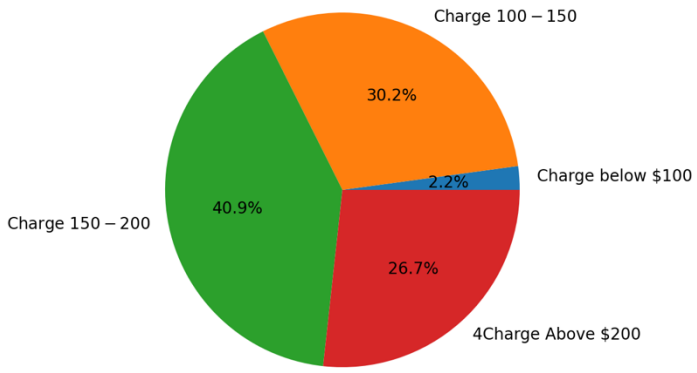
PART C4 – GENERATE UNIVARIATE AND BIVARIATE VISUALIZATIONS FOR THE DISTRIBUTIONS OF VARIABLES IN THE CLEANED DATA SET. INCLUDE THE TARGET VARIABLE IN YOUR BIVARIATE VISUALIZATIONS

As demonstrated in my report in D207:

1. Univariate Stats: the simplest form of statistics. I am going to generate pie charts for all important variables in this analysis:

```
#Monthly Charge Evaluation
def monthly_charge(df_clean):
    if df_clean['MonthlyCharge'] <= 100:
        return "Charge Level 1"
    elif (df_clean['MonthlyCharge'] > 101) & (df_clean['MonthlyCharge'] <=
150):
        return "Charge Level 2"
```

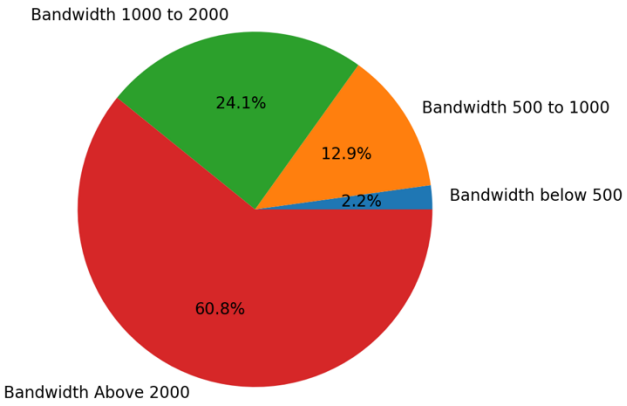
```
    elif (df_clean['MonthlyCharge'] > 151) & (df_clean['MonthlyCharge'] <=
200):
        return "Charge Level 3"
    elif (df_clean['MonthlyCharge'] > 201):
        return "Charge Level 4"

df_clean_6 = df_clean.copy()
df_clean_6['Monthly_Charge'] = df_clean_6.apply(lambda
df_clean_6:monthly_charge(df_clean_6), axis=1)

charge1_count = df_clean_6['Monthly_Charge'].value_counts()['Charge Level 1']
charge2_count = df_clean_6['Monthly_Charge'].value_counts()['Charge Level 2']
charge3_count = df_clean_6['Monthly_Charge'].value_counts()['Charge Level 3']
charge4_count = df_clean_6['Monthly_Charge'].value_counts()['Charge Level 4']

charge_total = charge1_count + charge2_count + charge3_count + charge4_count

pct_charge1 = (charge1_count / charge_total) * 100
pct_charge2 = (charge2_count / charge_total) * 100
pct_charge3 = (charge3_count / charge_total) * 100
pct_charge4 = (charge4_count / charge_total) * 100

#Pie Chart of Income

plt.figure(figsize=(5,5))
labels = ["Charge below 100", "Charge $101-$150", "Charge $151 - $200",
"Charge above $200"]
values = [pct_charge1, pct_charge2, pct_charge3, pct_charge4]
plt.pie(values, labels=labels, autopct="%.1f%%")
plt.show()
```



Picture 7: MonthlyCharge Pie Chart

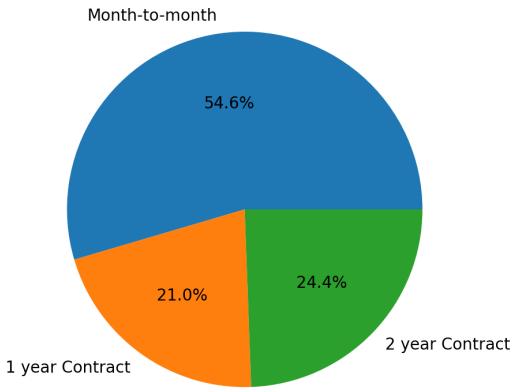
```
#Bandwidth_GB_Year Aalysis
print(df_clean['Bandwidth_GB_Year'].unique())
```

```
def bandwidth(df_clean):
    if df_clean['Bandwidth_GB_Year'] <= 500:
        return "Bandwidth Below 500"
    elif (df_clean['Bandwidth_GB_Year'] > 500) &
(df_clean['Bandwidth_GB_Year'] <= 1000):
        return "Bandwidth 500 - 1000"
    elif (df_clean['Bandwidth_GB_Year'] > 1000) &
(df_clean['Bandwidth_GB_Year'] <= 2000):
        return "Bandwidth 1000 - 2000"
    elif (df_clean['Bandwidth_GB_Year'] > 2000):
        return "Bandwidth Above 2000"

df_clean_10 = df_clean.copy()
df_clean_10['Bandwidth_GB_Year'] = df_clean_10.apply(lambda
df_clean_10:bandwidth(df_clean_10), axis=1)

bd1_count = df_clean_10['Bandwidth_GB_Year'].value_counts()['Bandwidth Below
500']
bd2_count = df_clean_10['Bandwidth_GB_Year'].value_counts()['Bandwidth 500 -
1000']
bd3_count = df_clean_10['Bandwidth_GB_Year'].value_counts()['Bandwidth 1000 -
2000']
bd4_count = df_clean_10['Bandwidth_GB_Year'].value_counts()['Bandwidth Above
2000']

bd_total = bd1_count + bd2_count + bd3_count + bd4_count

pct_bd1 = (bd1_count / bd_total) * 100
pct_bd2 = (bd2_count / bd_total) * 100
pct_bd3 = (bd3_count / bd_total) * 100
pct_bd4 = (bd4_count / bd_total) * 100
#Pie Chart of Monthly Charge

plt.figure(figsize=(5,5))
labels = ["Bandwidth below 500", "Bandwidth 500 to 1000", "Bandwidth 1000 to
2000", "Bandwidth Above 2000"]
values = [pct_bd1, pct_bd2, pct_bd3, pct_bd4]
plt.pie(values, labels=labels, autopct="%.1f%%")
plt.show()
```



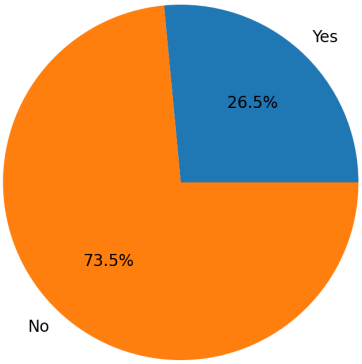
Picture 8: Bandwidth_GB_Year Pie Chart

```
print(df_clean['Contract'].unique())

month_to_month_count= df_clean['Contract'].value_counts()['Month-to-month']
one_year_count= df_clean['Contract'].value_counts()['One year']
two_year_count = df_clean['Contract'].value_counts()['Two Year']

contract_total = month_to_month_count + one_year_count + two_year_count

contract1_pct = (month_to_month_count / contract_total) * 100
contract2_pct = (one_year_count / contract_total) * 100
contract3_pct = (two_year_count / contract_total) * 100

#Pie Chart of types of Contracts

plt.figure(figsize=(5,5))
labels = ["Month-to-month", "1 year Contract", "2 year Contract"]
values = [contract1_pct, contract2_pct, contract3_pct]
plt.pie(values, labels=labels, autopct="%.1f%%")
plt.show()
```



Picture 9:Contract Pie Chart

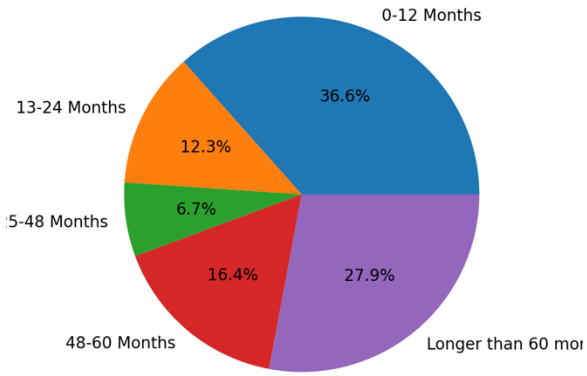
```
#Creating a Pie Chart to Visualize the Churn rate

plt.figure(figsize=(5,5))
labels = ["Yes", "No"]
values = [26.5, 73.5]
plt.pie(values, labels=labels, autopct="%.1f%%")
plt.show()
```



Picture 10:Churn Pie Chart

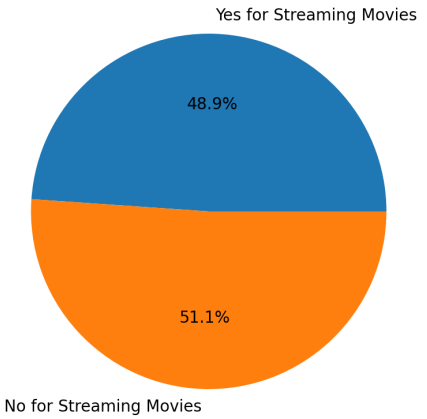
```
#Tenure Evaluation: I am going to create tenure intervals (0-12 months, 13-24
months, 25-48 months, 49-60 months, greater than 60 months)

def tenure_bands(df_clean):
    if df_clean['Tenure'] <= 12:
```

```
        return "Tenure_0-12"
    elif (df_clean['Tenure'] > 12) & (df_clean['Tenure'] <= 24):
        return "Tenure_13-24"
    elif (df_clean['Tenure'] > 24) & (df_clean['Tenure'] <= 48):
        return "Tenure_25-48"
    elif (df_clean['Tenure'] > 48) & (df_clean['Tenure'] <= 60):
        return "Tenure_49-60"
    elif df_clean['Tenure'] > 60:
        return "Tenure_gt_60"

df_clean_2 = df_clean.copy()
df_clean_2['tenure_group'] = df_clean_2.apply(lambda
df_clean_2:tenure_bands(df_clean_2), axis=1)
df_clean_2['tenure_group']

band1_count= df_clean_2['tenure_group'].value_counts()['Tenure_0-12']
band2_count = df_clean_2['tenure_group'].value_counts()['Tenure_13-24']
band3_count= df_clean_2['tenure_group'].value_counts()['Tenure_25-48']
band4_count= df_clean_2['tenure_group'].value_counts()['Tenure_49-60']
band5_count= df_clean_2['tenure_group'].value_counts()['Tenure_gt_60']

band_total = band1_count + band2_count + band3_count + band4_count +
band5_count

pct_band1 = (band1_count / band_total) * 100
pct_band2 = (band2_count / band_total) * 100
pct_band3 = (band3_count / band_total) * 100
pct_band4 = (band4_count / band_total) * 100
pct_band5 = (band5_count / band_total) * 100

#Pie Chart of Tenure Time

plt.figure(figsize=(5,5))
labels = ["0-12 Months", "13-24 Months", "25-48 Months", "48-60 Months",
"Longer than 60 months"]
values = [pct_band1, pct_band2, pct_band3, pct_band4, pct_band5]
plt.pie(values, labels=labels, autopct="%.1f%%")
plt.show()
```



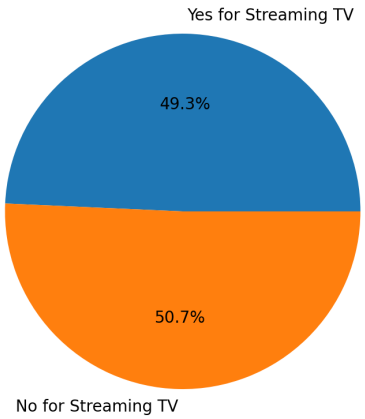
Picture 11: Tenure Pie Chart

```
#Streaming Movies
sm_yes_count = churn_df['StreamingMovies'].value_counts()['Yes']
sm_no_count = churn_df['StreamingMovies'].value_counts()['No']

streamingm_total = sm_yes_count + sm_no_count

sm_yes_pct = (sm_yes_count / streamingm_total) * 100
sm_no_pct = (sm_no_count / streamingm_total) * 100

#Pie Chart of type of Streaming Movies
plt.figure(figsize=(5,5))
labels = ["Yes for Streaming Movies", "No for Streaming Movies"]
values = [sm_yes_pct, sm_no_pct]
plt.pie(values, labels=labels, autopct="%.1f%%")
plt.show()
```



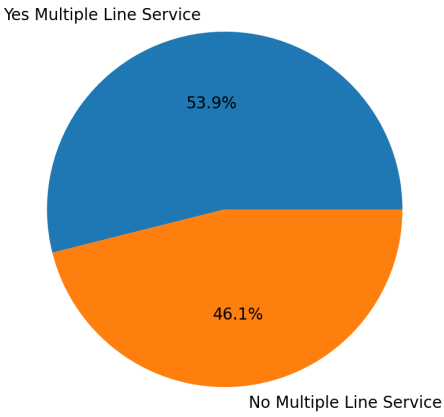
Picture 12: StreamingMovies Pie Chart

```
#Streaming TV
st_yes_count = churn_df['StreamingTV'].value_counts()['Yes']
st_no_count = churn_df['StreamingTV'].value_counts()['No']

streamingtv_total = st_yes_count + st_no_count

st_yes_pct = (st_yes_count / streamingtv_total) * 100
st_no_pct = (st_no_count / streamingtv_total) * 100

#Pie Chart of type of Streaming TV

plt.figure(figsize=(5,5))
labels = ["Yes for Streaming TV", "No for Streaming TV"]
values = [st_yes_pct, st_no_pct]
plt.pie(values, labels=labels, autopct="%.1f%%")
plt.show()
```



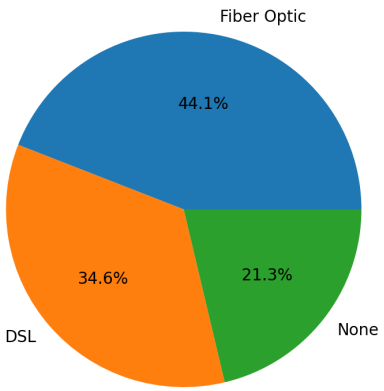
Picture 13: StreamingTV Pie Chart

```
#Multiple lines service
no_multiple_count = churn_df['Multiple'].value_counts()['Yes']

yes_multiple_count = churn_df['Multiple'].value_counts()['No']

multiple_total = no_multiple_count + yes_multiple_count

no_multiple_pct = (no_multiple_count / multiple_total) *100
yes_multiple_pct = (yes_multiple_count / multiple_total) *100


#Pie Chart of Multiple
plt.figure(figsize=(5,5))
labels = ["Yes Multiple Line Service", "No Multiple Line Service"]
values = [yes_multiple_pct, no_multiple_pct]
plt.pie(values, labels=labels, autopct="%.1f%%")
plt.show()
```



Picture 14:Multiple Pie Chart

```
#Type of Internet Service
print(df_clean['InternetService'].unique())

fiber_optic_count = df_clean['InternetService'].value_counts()['Fiber Optic']
dsl_count = df_clean['InternetService'].value_counts()['DSL']
none_count = df_clean['InternetService'].value_counts()['None']

service_total = fiber_optic_count + dsl_count + none_count

fiber_pct = (fiber_optic_count / service_total) * 100
dsl_pct = (dsl_count / service_total) * 100
none_pct = (none_count / service_total) * 100

#Pie Chart of type of Internet Service

plt.figure(figsize=(5,5))
labels = ["Fiber Optic", "DSL", "None"]
values = [fiber_pct, dsl_pct, none_pct]
plt.pie(values, labels=labels, autopct="%.1f%%")
plt.show()
```

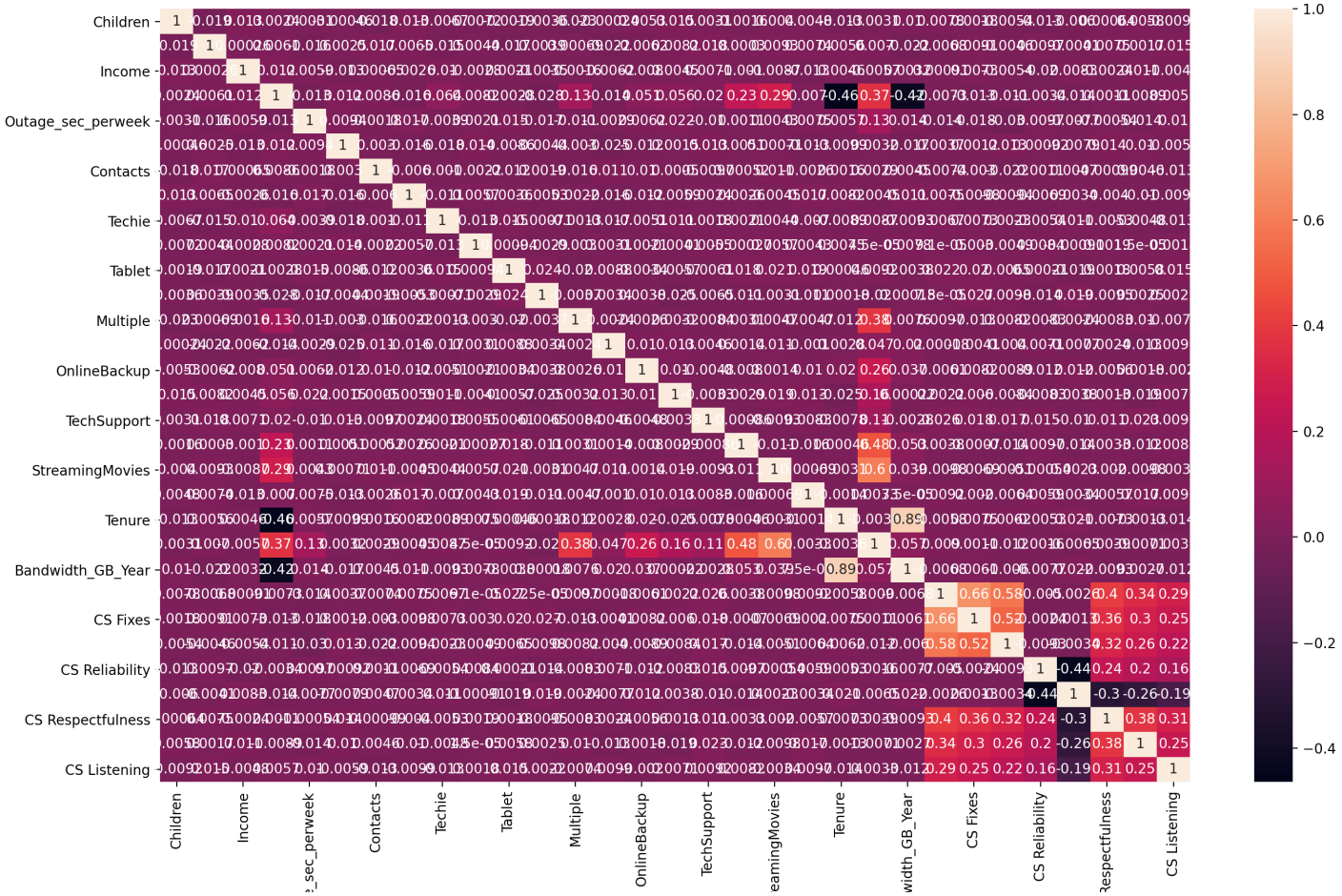


Picture 15: InternetService Pie Chart

2. **Bivariate Stats**: After analyzing all important variables independently, it's time to analyze if there is a relationship between two variables. When two variables have a correlation it means that when one variable changes the second variable will also change by a certain amount.

I am going to create graphs of each variable against the churn rate as I presented in my report from D206 [4]. And also a correlation matrix.

```
#Creating a label encoder object
le = LabelEncoder()
churn_df2['Churn'] = le.fit_transform(churn_df2['Churn'])
# Label Encoding will be used for categorical variables with 2 or less unique
values
le_count = 0
for col in churn_df.columns[1:]:
    if churn_df2[col].dtype == 'object':
        if len(list(churn_df2[col].unique())) <= 2:
            le.fit(churn_df2[col])
            churn_df2[col] = le.transform(churn_df2[col])
            le_count += 1
print('{} columns were label encoded.'.format(le_count))
#**********************************************************************
****************************************

#Correclation Matrix for all Numeric Columns
sns.heatmap(churn_df.corr(), annot=True)
plt.show()
```



Picture 16: Correlation Matrix

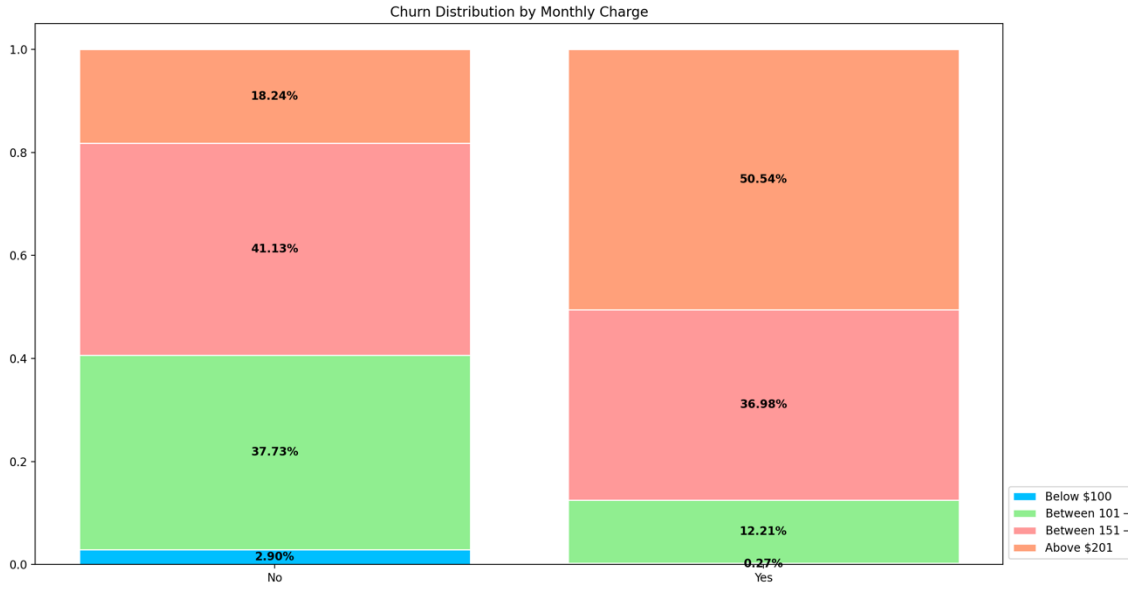
From the correlation matrix above we see that the customer survey variables are all correlated to each other and "Bandwidth_GB_Year" is highly correlated with "Tenure". I am going to drop Bandwidth and customer survey variables.

```
#Dropping Bandwidth due to its highly correlation with Tenure
churn_df2 = churn_df2.drop(columns=['Bandwidth_GB_Year'])
#Dropping Customer Survey Answers
churn_df2 = churn_df2.drop(columns=['CS Responses', 'CS Fixes', 'CS
Replacements', 'CS Reliability', 'CS Options',
'CS Respectfulness', 'CS Courteous', 'CS Listening'])
```

Monthly Charge and Churn Rate:

```
#Monthly Charge and Churn Rate
no_churn = ((df_clean_2[df_clean_2['Churn']
=='No']['Monthly_Charge'].value_counts()) /(df_clean_2[df_clean_2['Churn']==
'No']['Monthly_Charge'].value_counts().sum()))
yes_churn = ((df_clean_2[df_clean_2['Churn']
=='Yes']['Monthly_Charge'].value_counts())
```

```
/(df_clean_2[df_clean_2['Churn']=='Yes']['Monthly_Charge'].value_counts().sum
())

# Getting values from the group and categories
x_labels = df_clean_2['Churn'].value_counts().keys().tolist()
level1 = [no_churn['Charge below $100'], yes_churn['Charge below $100']]
level2 = [no_churn['Charge $100 - $150'], yes_churn['Charge $100 - $150']]
level3 = [no_churn['Charge $150 - $200'], yes_churn['Charge $150 - $200']]
level4 = [no_churn['Charge Above $200'], yes_churn['Charge Above $200']]

# Plotting bars
barWidth = 0.8
plt.figure(figsize=(7,7))
ax1 = plt.bar(x_labels, level1, color='#00BFFF', label=('Charge below $100'),
edgecolor='white', width=barWidth)
ax2 = plt.bar(x_labels, level2, bottom=level1, color='lightgreen',
label=('Charge $100 - $150'), edgecolor='white', width=barWidth)
ax3 = plt.bar(x_labels, level3, bottom=np.array(level2) + np.array(level1),
color='#FF9999', label=('Charge $150 - $200'), edgecolor='white',
width=barWidth)
ax4 = plt.bar(x_labels, level4, bottom=np.array(level1) + np.array(level2) +
np.array(level3), color='#FFA07A', label=('Charge Above $200'),
edgecolor='white', width=barWidth)

plt.legend(loc='lower left', bbox_to_anchor=(1,0))
plt.title('Churn Distribution by Monthly Charge')

for r1, r2, r3, r4 in zip(ax1, ax2, ax3, ax4):
    h1 = r1.get_height()
    h2 = r2.get_height()
    h3 = r3.get_height()
    h4 = r4.get_height()
    plt.text(r1.get_x() + r1.get_width() / 2., h1 / 2., '{:.2%}'.format(h1),
ha='center', va='center', color='black', fontweight='bold')
    plt.text(r2.get_x() + r2.get_width() / 2., h1 + h2 / 2.,
'{:.2%}'.format(h2), ha='center', va='center', color='black',
fontweight='bold')
    plt.text(r3.get_x() + r3.get_width() / 2., h1 + h2 + h3 / 2.,
'{:.2%}'.format(h3), ha='center', va='center', color='black',
fontweight='bold')
    plt.text(r4.get_x() + r4.get_width() / 2., h1 + h2 + h3 + h4 / 2.,
'{:.2%}'.format(h4), ha='center', va='center', color='black',
fontweight='bold')
plt.show()
```



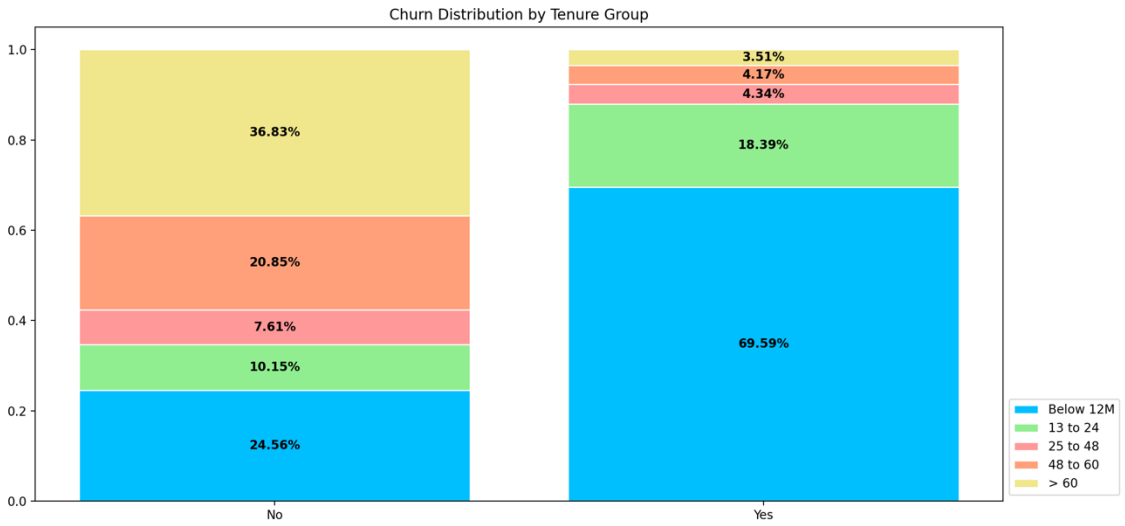
Picture 17: MonthlyCharge and Churn Rate

This graph above shows that we see a big customer churn when monthly charges are above $200 per month.

Tenure and Churn Rate:

```
#Tenure and Churn Rate
no_churn =
((df_clean_3[df_clean_3['Churn']=='No']['tenure_group'].value_counts())
/(df_clean_3[df_clean_3['Churn']=='No']['tenure_group'].value_counts().sum())
)
yes_churn =
((df_clean_3[df_clean_3['Churn']=='Yes']['tenure_group'].value_counts())
/(df_clean_3[df_clean_3['Churn']=='Yes']['tenure_group'].value_counts().sum()
))

# Getting values from the group and categories
x_labels = churn_df['Churn'].value_counts().keys().tolist()
t_0_12 = [no_churn['Tenure_0-12'], yes_churn['Tenure_0-12']]
t_13_24 = [no_churn['Tenure_13-24'], yes_churn['Tenure_13-24']]
t_25_48 = [no_churn['Tenure_25-48'], yes_churn['Tenure_25-48']]
t_49_60 = [no_churn['Tenure_49-60'], yes_churn['Tenure_49-60']]
t_gt_60 = [no_churn['Tenure_gt_60'], yes_churn['Tenure_gt_60']]

# Plotting bars
barWidth = 0.8
plt.figure(figsize=(7,7))
ax1 = plt.bar(x_labels, t_0_12, color='#00BFFF', label=('Below 12M'),
edgecolor='white', width=barWidth)
ax2 = plt.bar(x_labels, t_13_24, bottom=t_0_12, color='lightgreen',
label=('13 to 24'), edgecolor='white', width=barWidth)
ax3 = plt.bar(x_labels, t_25_48, bottom=np.array(t_0_12) + np.array(t_13_24),
```

```
color='#FF9999',  label=('25 to 48'), edgecolor='white', width=barWidth)
ax4 = plt.bar(x_labels, t_49_60, bottom=np.array(t_0_12) + np.array(t_13_24)
+ np.array(t_25_48), color='#FFA07A', label=('48 to 60'), edgecolor='white',
width=barWidth)
ax5 = plt.bar(x_labels, t_gt_60, bottom=np.array(t_0_12) + np.array(t_13_24)
+ np.array(t_25_48) + np.array(t_49_60), color='#F0E68C', label=('> 60'),
edgecolor='white', width=barWidth)

plt.legend(loc='lower left', bbox_to_anchor=(1,0))
plt.title('Churn Distribution by Tenure Group')

for r1, r2, r3, r4, r5 in zip(ax1, ax2, ax3, ax4, ax5):
    h1 = r1.get_height()
    h2 = r2.get_height()
    h3 = r3.get_height()
    h4 = r4.get_height()
    h5 = r5.get_height()
    plt.text(r1.get_x() + r1.get_width() / 2., h1 / 2., '{:.2%}'.format(h1),
ha='center', va='center', color='black', fontweight='bold')
    plt.text(r2.get_x() + r2.get_width() / 2., h1 + h2 / 2.,
'{:.2%}'.format(h2), ha='center', va='center', color='black',
fontweight='bold')
    plt.text(r3.get_x() + r3.get_width() / 2., h1 + h2 + h3 / 2.,
'{:.2%}'.format(h3), ha='center', va='center', color='black',
fontweight='bold')
    plt.text(r4.get_x() + r4.get_width() / 2., h1 + h2 + h3 + h4 / 2.,
'{:.2%}'.format(h4), ha='center', va='center', color='black',
fontweight='bold')
    plt.text(r5.get_x() + r5.get_width() / 2., h1 + h2 + h3 + h4 + h5 / 2.,
'{:.2%}'.format(h5), ha='center', va='center', color='black',
fontweight='bold')
plt.show()
```



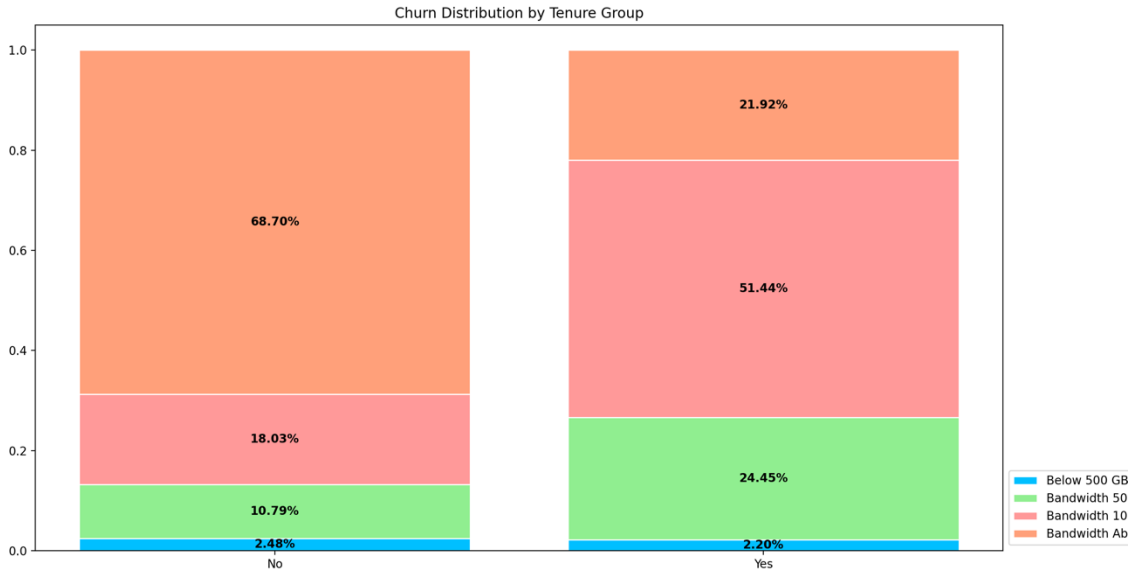
Picture 18: Tenure and Churn Rate

The graph above tells us that customers who have been a short period of time with the company (below 12 months) tend to churn more.

Bandwidth and Churn Rate:

```
#Bandwidth and churn
no_churn =
((df_clean_4[df_clean_4['Churn']=='No']['Bandwidth_GB_Year'].value_counts())
/(df_clean_4[df_clean_4['Churn']=='No']['Bandwidth_GB_Year'].value_counts().s
um()))
yes_churn =
((df_clean_4[df_clean_4['Churn']=='Yes']['Bandwidth_GB_Year'].value_counts())
/(df_clean_4[df_clean_4['Churn']=='Yes']['Bandwidth_GB_Year'].value_counts().
sum()))

# Getting values from the group and categories
x_labels = churn_df['Churn'].value_counts().keys().tolist()
b_500 = [no_churn['Bandwidth Below 500'], yes_churn['Bandwidth Below 500']]
b_500_1000 = [no_churn['Bandwidth 500 - 1000'], yes_churn['Bandwidth 500 -
1000']]
b_1000_2000 = [no_churn['Bandwidth 1000 - 2000'], yes_churn['Bandwidth 1000 -
2000']]
b_gt_2000 = [no_churn['Bandwidth Above 2000'], yes_churn['Bandwidth Above
2000']]

# Plotting bars
barWidth = 0.8
plt.figure(figsize=(7,7))
ax1 = plt.bar(x_labels, b_500, color='#00BFFF', label=('Below 500 GB'),
edgecolor='white', width=barWidth)
ax2 = plt.bar(x_labels, b_500_1000, bottom=b_500, color='lightgreen',
label=('Band 500-1000'), edgecolor='white', width=barWidth)
ax3 = plt.bar(x_labels, b_1000_2000, bottom=np.array(b_500) +
np.array(b_500_1000), color='#FF9999',  label=('Band 1000-2000'),
edgecolor='white', width=barWidth)
ax4 = plt.bar(x_labels, b_gt_2000, bottom=np.array(b_500) +
np.array(b_500_1000) + np.array(b_1000_2000), color='#FFA07A', label=('Above
2000'), edgecolor='white', width=barWidth)

plt.legend(loc='lower left', bbox_to_anchor=(1,0))
plt.title('Churn Distribution by Bandwidth_GB_Year')

for r1, r2, r3, r4 in zip(ax1, ax2, ax3, ax4):
    h1 = r1.get_height()
    h2 = r2.get_height()
    h3 = r3.get_height()
    h4 = r4.get_height()
    plt.text(r1.get_x() + r1.get_width() / 2., h1 / 2., '{:.2%}'.format(h1),
ha='center', va='center', color='black', fontweight='bold')
    plt.text(r2.get_x() + r2.get_width() / 2., h1 + h2 / 2.,
'{:.2%}'.format(h2), ha='center', va='center', color='black',
fontweight='bold')
    plt.text(r3.get_x() + r3.get_width() / 2., h1 + h2 + h3 / 2.,
'{:.2%}'.format(h3), ha='center', va='center', color='black',
fontweight='bold')
    plt.text(r4.get_x() + r4.get_width() / 2., h1 + h2 + h3 + h4 / 2.,
```

```
'{:.2%}'.format(h4), ha='center', va='center', color='black',
fontweight='bold')
plt.show()
```



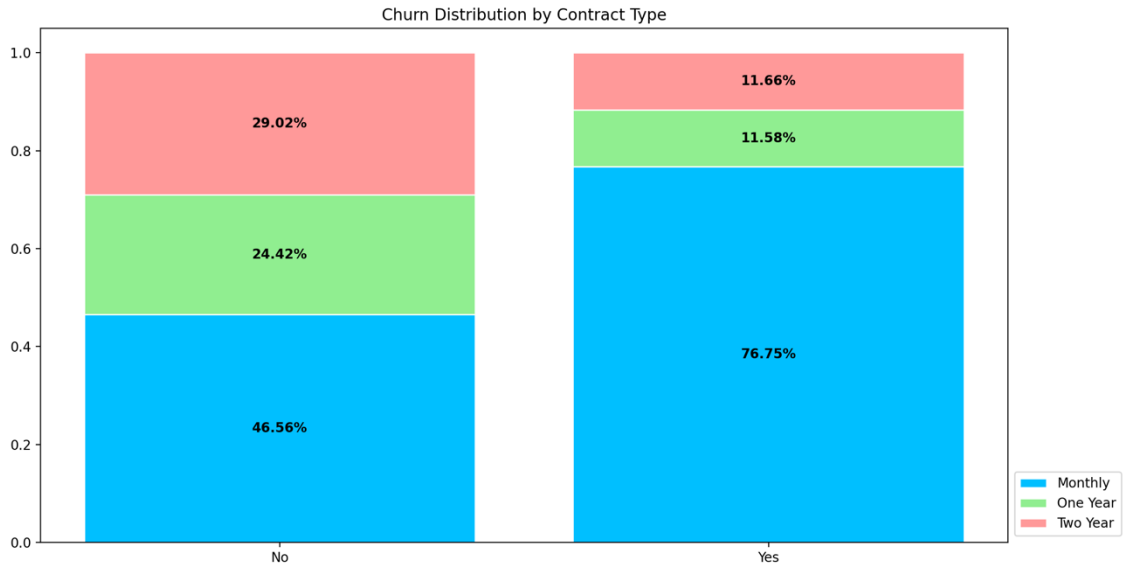
Picture 19: Bandwidth and Churn Rate

The previous graph shows that customers who signed up for having 1000-2000 GB used per year churn more.

Contract and Churn Rate:

```
# #Contract Type and Churn Rate
no_churn = ((churn_df[churn_df['Churn']=='No']['Contract'].value_counts())
/(churn_df[churn_df['Churn']=='No']['Contract'].value_counts().sum()))
yes_churn = ((churn_df[churn_df['Churn']=='Yes']['Contract'].value_counts())
/(churn_df[churn_df['Churn']=='Yes']['Contract'].value_counts().sum()))

# Getting values from the group and categories
x_labels = churn_df['Churn'].value_counts().keys().tolist()
monthly = [no_churn['Month-to-month'], yes_churn['Month-to-month']]
one_year = [no_churn['One year'], yes_churn['One year']]
two_year = [no_churn['Two Year'], yes_churn['Two Year']]

# Plotting bars
barWidth = 0.8
plt.figure(figsize=(7,7))
ax1 = plt.bar(x_labels, monthly, color='#00BFFF', label=('Monthly'),
edgecolor='white', width=barWidth)
ax2 = plt.bar(x_labels, one_year, bottom=monthly, color='lightgreen',
label=('One Year'), edgecolor='white', width=barWidth)
ax3 = plt.bar(x_labels, two_year, bottom=np.array(one_year) +
np.array(monthly), color='#FF9999', label=('Two Year'), edgecolor='white',
width=barWidth)
```

```
plt.legend(loc='lower left', bbox_to_anchor=(1,0))
plt.title('Churn Distribution by Contract Type')

for r1, r2, r3 in zip(ax1, ax2, ax3):
    h1 = r1.get_height()
    h2 = r2.get_height()
    h3 = r3.get_height()
    plt.text(r1.get_x() + r1.get_width() / 2., h1 / 2., '{:.2%}'.format(h1),
ha='center', va='center', color='black', fontweight='bold')
    plt.text(r2.get_x() + r2.get_width() / 2., h1 + h2 / 2.,
'{:.2%}'.format(h2), ha='center', va='center', color='black',
fontweight='bold')
    plt.text(r3.get_x() + r3.get_width() / 2., h1 + h2 + h3 / 2.,
'{:.2%}'.format(h3), ha='center', va='center', color='black',
fontweight='bold')
plt.show()
```



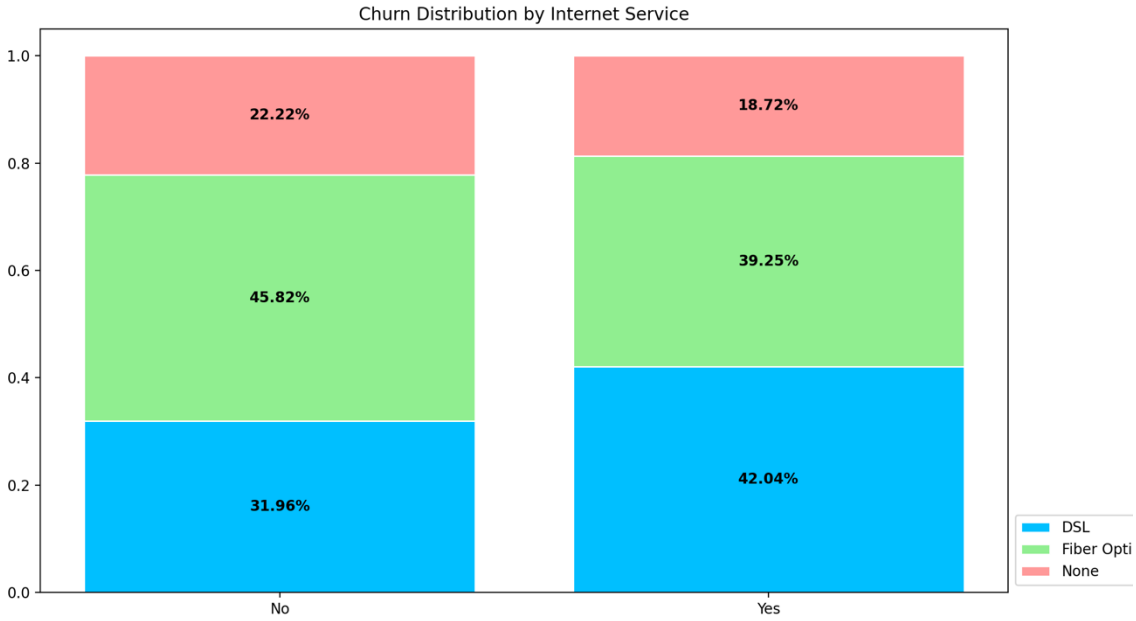
Picture 20: Contract and Churn Rate

We can clearly see that customers on a month-to-month contract churn more than other customers.

Internet Service and Churn Rate:

```
#Internet Service and Churn Rate
no_churn =
((churn_df[churn_df['Churn']=='No']['InternetService'].value_counts())
/(churn_df[churn_df['Churn']=='No']['InternetService'].value_counts().sum()))
yes_churn =
((churn_df[churn_df['Churn']=='Yes']['InternetService'].value_counts())
/(churn_df[churn_df['Churn']=='Yes']['InternetService'].value_counts().sum())
)

# Getting values from the group and categories
x_labels = churn_df['Churn'].value_counts().keys().tolist()
```

```
dsl = [no_churn['DSL'], yes_churn['DSL']]
fiber_optic = [no_churn['Fiber Optic'], yes_churn['Fiber Optic']]
none = [no_churn['None'], yes_churn['None']]

# Plotting bars
barWidth = 0.8
plt.figure(figsize=(7,7))
ax1 = plt.bar(x_labels, dsl, color='#00BFFF', label=('DSL'),
edgecolor='white', width=barWidth)
ax2 = plt.bar(x_labels, fiber_optic, bottom=dsl, color='lightgreen',
label=('Fiber Optic'), edgecolor='white', width=barWidth)
ax3 = plt.bar(x_labels, none, bottom=np.array(fiber_optic) + np.array(dsl),
color='#FF9999', label=('None'), edgecolor='white', width=barWidth)

plt.legend(loc='lower left', bbox_to_anchor=(1,0))
plt.title('Churn Distribution by Internet Service')

for r1, r2, r3 in zip(ax1, ax2, ax3):
    h1 = r1.get_height()
    h2 = r2.get_height()
    h3 = r3.get_height()
    plt.text(r1.get_x() + r1.get_width() / 2., h1 / 2., '{:.2%}'.format(h1),
ha='center', va='center', color='black', fontweight='bold')
    plt.text(r2.get_x() + r2.get_width() / 2., h1 + h2 / 2.,
'{:.2%}'.format(h2), ha='center', va='center', color='black',
fontweight='bold')
    plt.text(r3.get_x() + r3.get_width() / 2., h1 + h2 + h3 / 2.,
'{:.2%}'.format(h3), ha='center', va='center', color='black',
fontweight='bold')
plt.show()
```



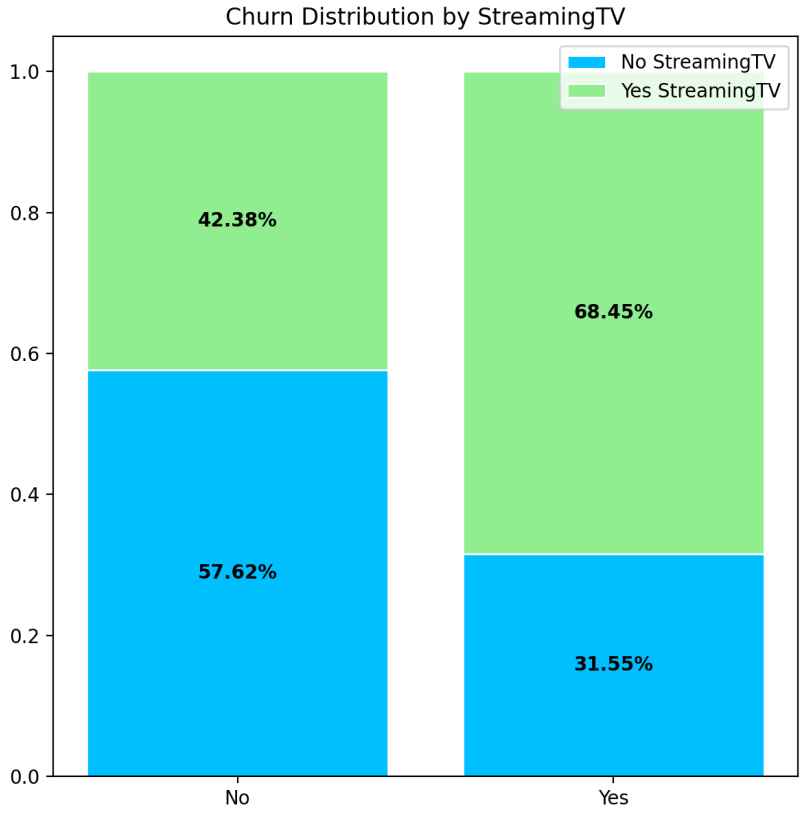
Picture 21: Internet Service and Churn Rate

The internet and churn graph tells us that customers with DSL service churn more than other customers.

Streaming TV and Churn Rate:

```
# #StreamingTV and Churn Rate
no_churn = ((churn_df[churn_df['Churn']=='No']['StreamingTV'].value_counts())
/(churn_df[churn_df['Churn']=='No']['StreamingTV'].value_counts().sum()))
yes_churn =
((churn_df[churn_df['Churn']=='Yes']['StreamingTV'].value_counts())
/(churn_df[churn_df['Churn']=='Yes']['StreamingTV'].value_counts().sum()))

# Getting values from the group and categories
x_labels = churn_df['Churn'].value_counts().keys().tolist()
yes_st = [no_churn['Yes'], yes_churn['Yes']]
no_st = [no_churn['No'], yes_churn['No']]

# Plotting bars
barWidth = 0.8
plt.figure(figsize=(7,7))
ax1 = plt.bar(x_labels, yes_st, color='#00BFFF', label=('Yes StreamingTV'),
edgecolor='white', width=barWidth)
ax2 = plt.bar(x_labels, no_st, bottom=yes_st, color='lightgreen', label=('No
StreamingTV'), edgecolor='white', width=barWidth)

plt.legend(loc='lower left', bbox_to_anchor=(1,0))
plt.title('Churn Distribution by StreamingTV')

for r1, r2 in zip(ax1, ax2):
```

```
    h1 = r1.get_height()
    h2 = r2.get_height()
    plt.text(r1.get_x() + r1.get_width() / 2., h1 / 2., '{:.2%}'.format(h1),
ha='center', va='center', color='black', fontweight='bold')
    plt.text(r2.get_x() + r2.get_width() / 2., h1 + h2 / 2.,
'{:.2%}'.format(h2), ha='center', va='center', color='black',
fontweight='bold')
plt.show()
```



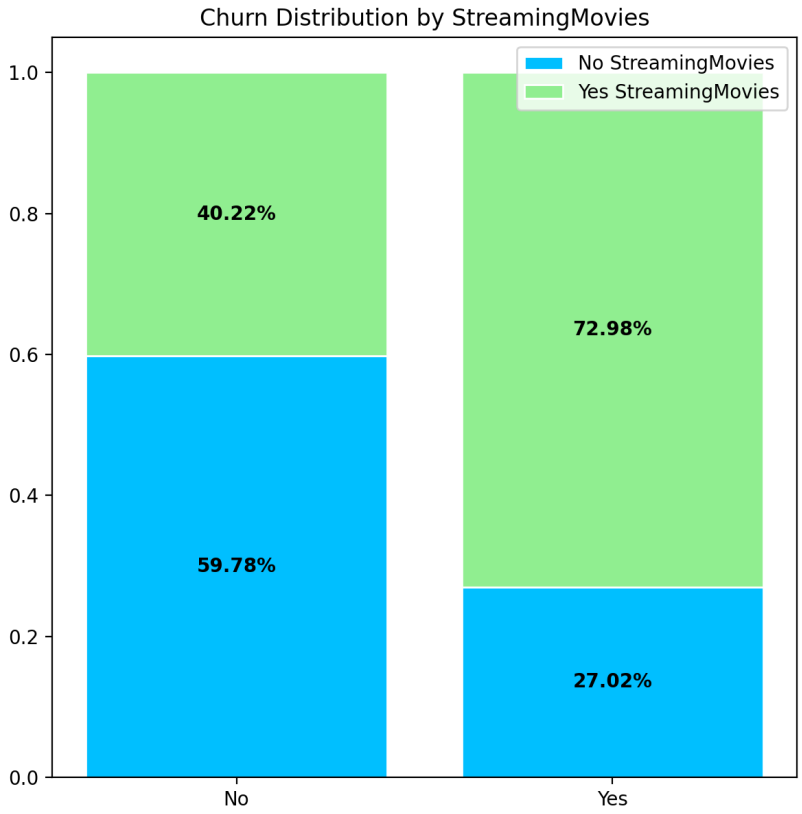
Picture 22: StreamingTV and Churn Rate

Now we can see that customers with streaming tv service churn more.

Streaming Movies and Churn Rate:

```
#StreamingMovies and Churn Rate
no_churn =
((churn_df[churn_df['Churn']=='No']['StreamingMovies'].value_counts())
/(churn_df[churn_df['Churn']=='No']['StreamingMovies'].value_counts().sum()))
yes_churn =
((churn_df[churn_df['Churn']=='Yes']['StreamingMovies'].value_counts())
/(churn_df[churn_df['Churn']=='Yes']['StreamingMovies'].value_counts().sum())
)

# Getting values from the group and categories
```

```
x_labels = churn_df['Churn'].value_counts().keys().tolist()
yes_sm = [no_churn['Yes'], yes_churn['Yes']]
no_sm = [no_churn['No'], yes_churn['No']]

# Plotting bars
barWidth = 0.8
plt.figure(figsize=(7,7))
ax1 = plt.bar(x_labels, yes_sm, color='#00BFFF', label=('Yes
StreamingMovies'), edgecolor='white', width=barWidth)
ax2 = plt.bar(x_labels, no_sm, bottom=yes_sm, color='lightgreen', label=('No
StreamingMovies'), edgecolor='white', width=barWidth)

plt.legend(loc='lower left', bbox_to_anchor=(1,0))
plt.title('Churn Distribution by StreamingMovies')

for r1, r2 in zip(ax1, ax2):
    h1 = r1.get_height()
    h2 = r2.get_height()
    plt.text(r1.get_x() + r1.get_width() / 2., h1 / 2., '{:.2%}'.format(h1),
ha='center', va='center', color='black', fontweight='bold')
    plt.text(r2.get_x() + r2.get_width() / 2., h1 + h2 / 2.,
'{:.2%}'.format(h2), ha='center', va='center', color='black',
fontweight='bold')
plt.show()
```



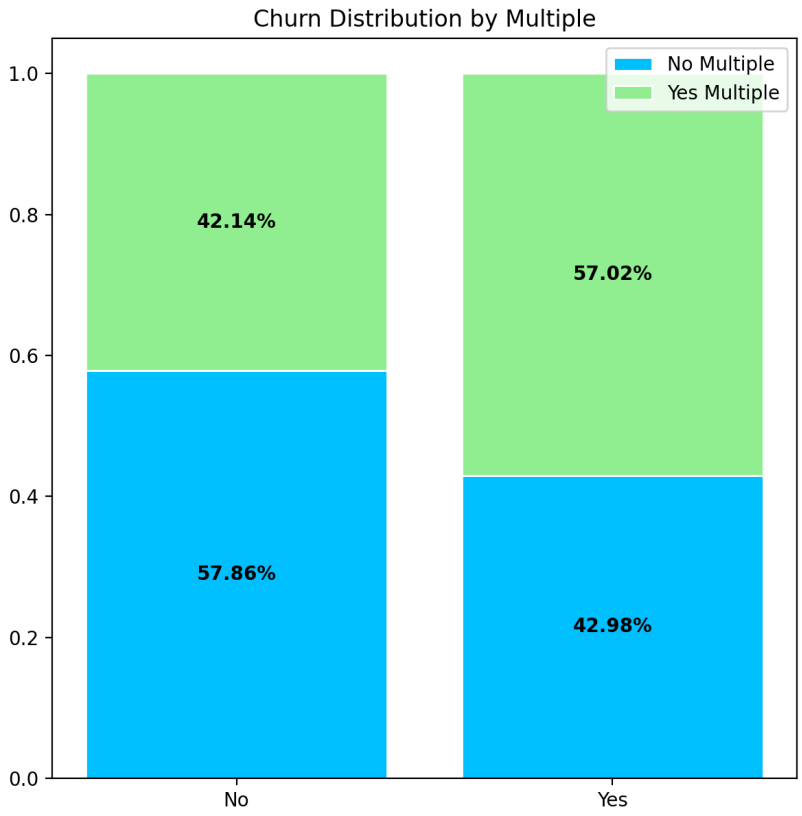
Picture 23: StreamingMovies and Churn Rate

Now we can see that customers with streaming movies service churn more.

Multiple and Churn Rate:

```
# Getting values from the group and categories
#Churn is my X axis
x_labels = churn_df['Churn'].value_counts().keys().tolist()
#Y axis will be my other variables
n_var = [no_churn['No'], yes_churn['No']]
y_var = [no_churn['Yes'], yes_churn['Yes']]

# Multiple and Churn
barWidth = 0.8
plt.figure(figsize=(7, 7))
ax1 = plt.bar(x_labels, n_var, color='#00BFFF', label=('No Multiple'),
edgecolor='white', width=barWidth)
ax2 = plt.bar(x_labels, y_var, bottom=n_var, color='lightgreen', label=('Yes
Multiple'), edgecolor='white', width=barWidth)
plt.legend()
plt.title('Churn Distribution by Multiple')
```

```
for r1, r2 in zip(ax1, ax2):
    h1 = r1.get_height()
    h2 = r2.get_height()
    plt.text(r1.get_x() + r1.get_width() / 2., h1 / 2., '{:.2%}'.format(h1),
ha='center', va='center', color='black', fontweight='bold')
    plt.text(r2.get_x() + r2.get_width() / 2., h1 + h2 / 2.,
'{:.2%}'.format(h2), ha='center', va='center', color='black',
fontweight='bold')
plt.show()
```



Picture 24: Multiple and Churn Rate

Customers who have signed up for multiple lines service also churn more.

PART C5 – PROVIDE A COPY OF THE PREPARED DATASET

The prepared data set will be attached as “prepared_churn_data.csv”.

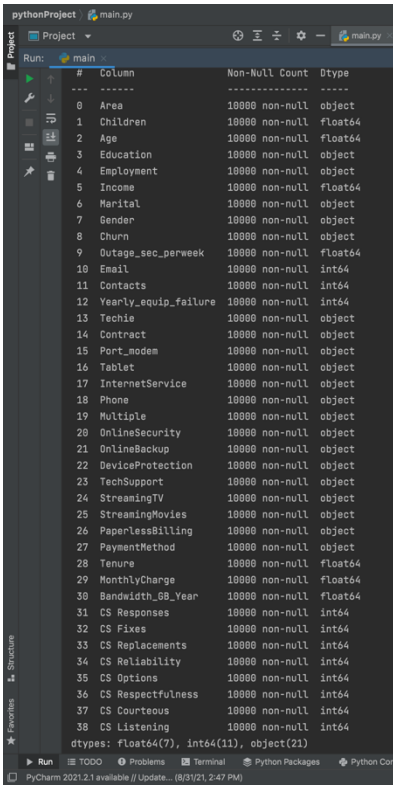
PART IV - MODEL COMPARISON AND ANALYSIS

PART D – COMPARE AN INITIAL AND A REDUCED MULTIPLE REGRESSION MODEL BY THE DOING THE FOLLOWING:

PART D1. Construct an initial multiple regression model from *all* predictors that were identified in Part C2.

We can formulate this customer churn prediction as a regression task. This technique is used to estimate the relationship between a dependent variable (churn) and independent variables that might predict the churn.

```
#Additional info about the remaining data:
data_types = churn_df2.info()
print(data_types)
```

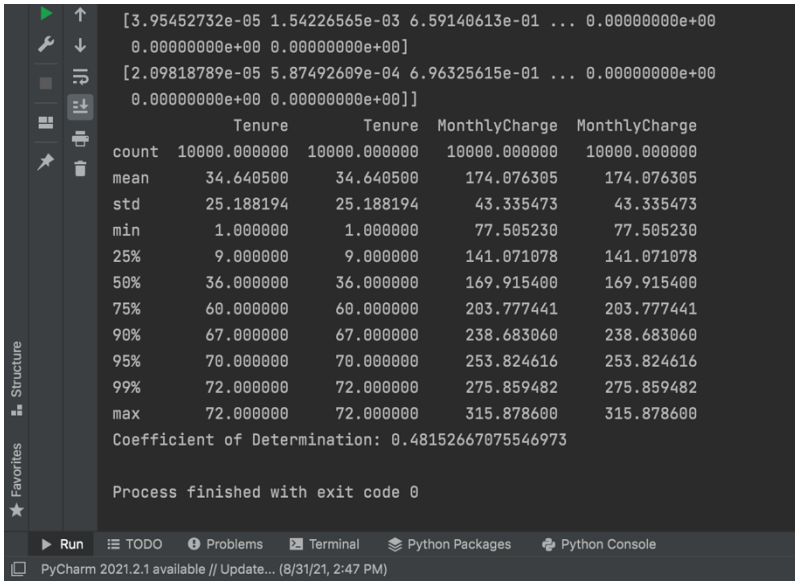


Picture 25: Types of data

From the correlation matrix showed previously we saw that Bandwidth and Tenure and highly correlated, so I have to drop one. I decided to drop Bandwidth_GB_Year. Also I dropped all customer survey columns since they are all correlated to each other and it will be hard to measure importance of one over the other.

Checking if the continuous variables present outliers:

```
# Checking for outliers in the continuous variables
numerical_churn_df2 = churn_df2[['Tenure','MonthlyCharge']]

# Checking for outliers at 25%, 50%, 75%, 90%, 95% and 99%
print(numerical_churn_df2.describe(percentiles=[.25, .5, .75, .90, .95,
.99]))
```



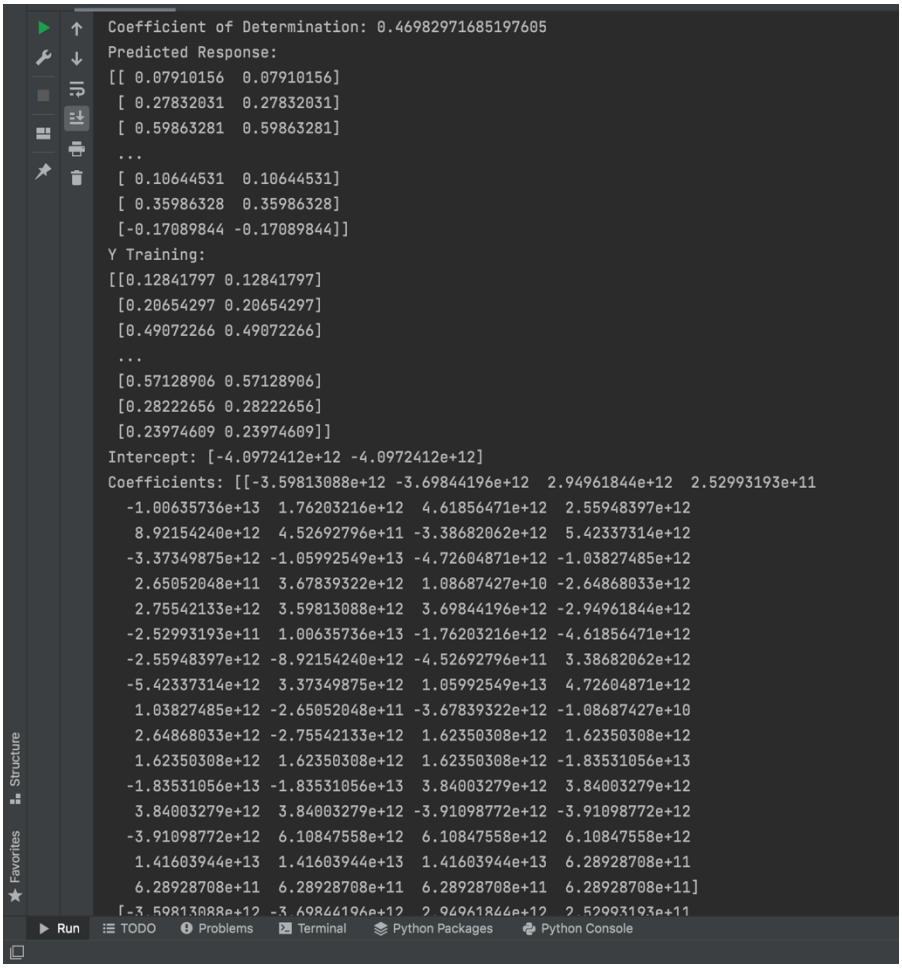
Picture 26: Outliers check for continuous variables

These variables do not have outliers since they get higher with the percentiles.

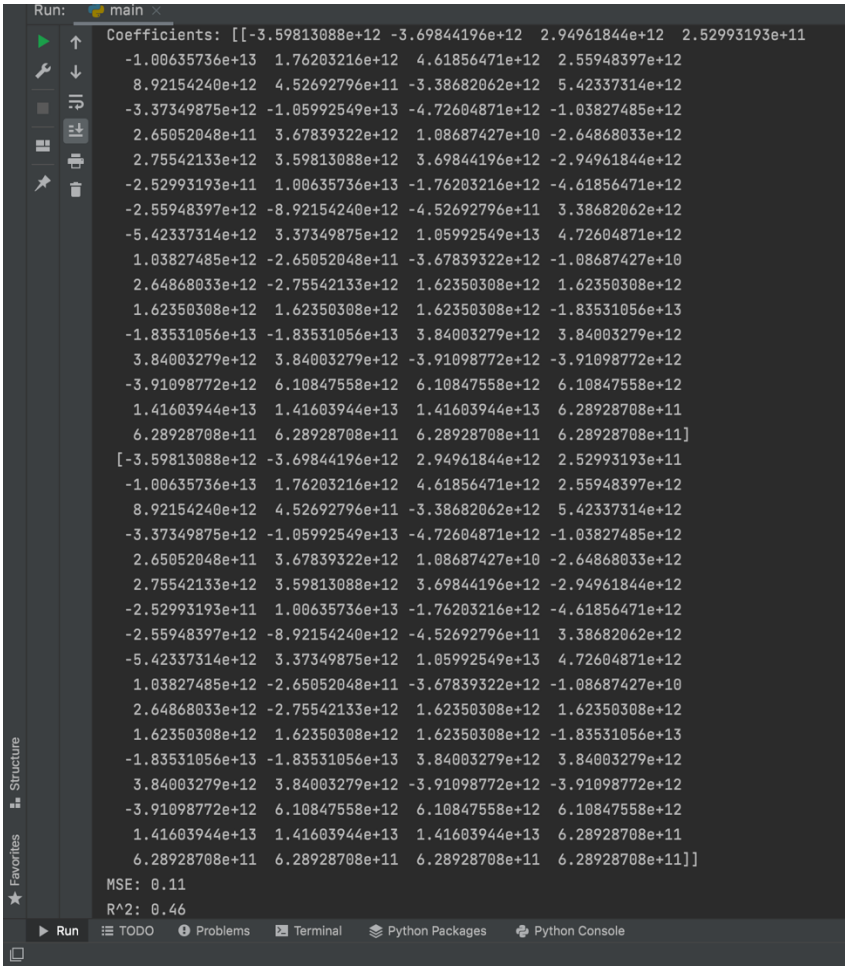
Linear Regression Model: [5]

```
#Linear Regression
#Target is the churn (Y)
y_data = churn_df2.Churn.values

#Remove Churn from remaining features
X_data = churn_df2.drop('Churn', axis = 1)

#Saving dataframe column titles to list
cols = X_data.columns

#Min-Max scaling object
mm = MinMaxScaler()
X_data = pd.DataFrame(mm.fit_transform(X_data))

#Training and Test data split 70/30
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, test_size
= 0.3, random_state = 0)

#Creating the model
model = LinearRegression().fit(X_train, y_train)

r_squared = model.score(X_train, y_train)
print('Coefficient of Determination:', r_squared)

#Creating predictions
y_hat_train = model.predict(X_train)
y_pred = model.predict(X_test)
```

```
print('Predicted Response:', y_pred, sep='\n')
print('Y Training:', y_hat_train, sep='\n')

print('Intercept:', model.intercept_)
print('Coefficients:', model.coef_)
print('MSE: %.2f' % mean_squared_error(y_test, y_pred))
print('R^2: %.2f' % r2_score(y_test, y_pred))
```



Picture 27: Model built based on all dataset



Picture 28: Coefficients, MSE and R^2

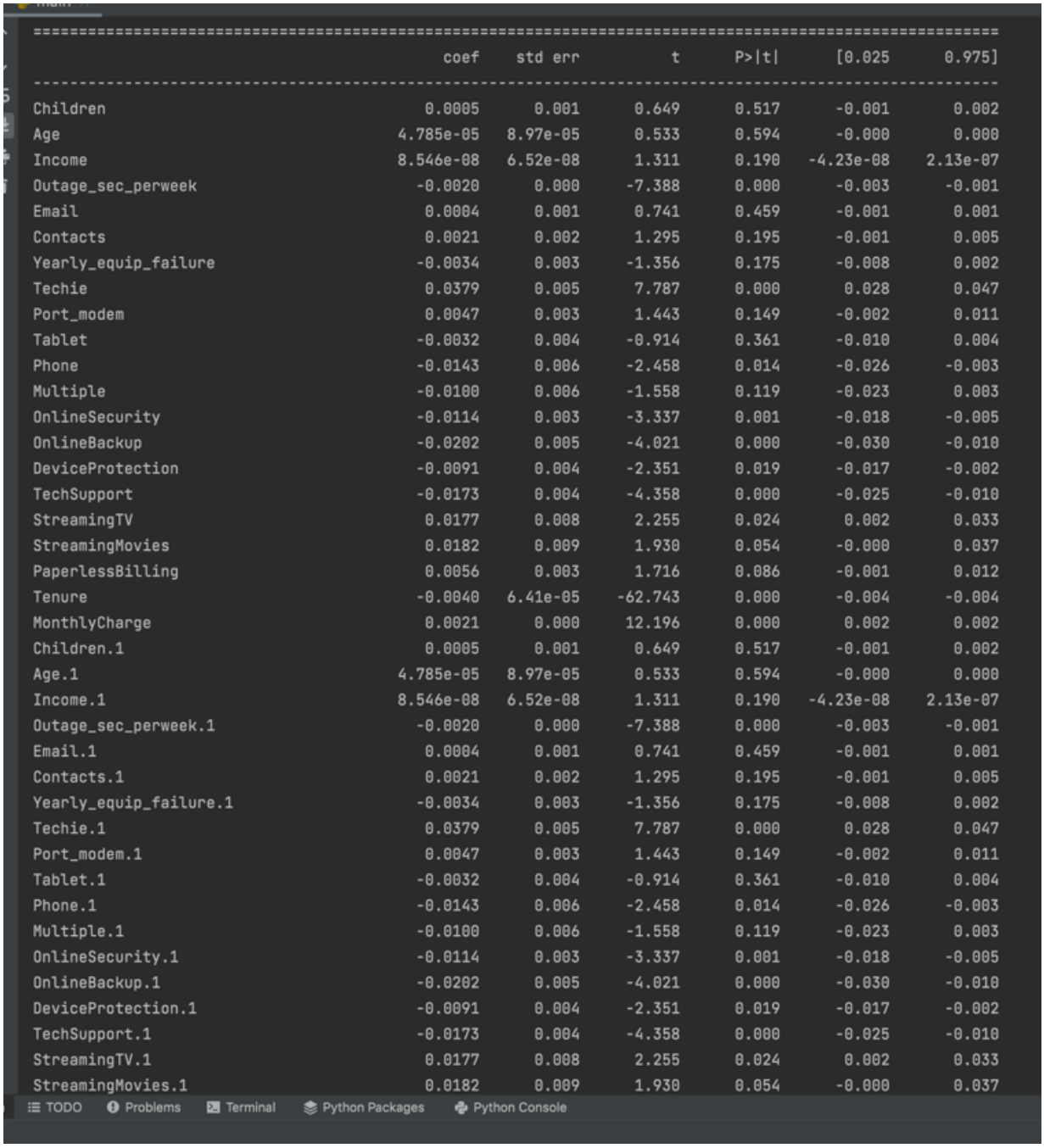
MSE is used to check how **close estimates or forecasts** are to actual values. Lower the MSE, the closer is forecast to actual. This is used as a model evaluation measure for regression models and the lower value indicates a better fit.[2]

```
churn_df2['intercept'] = 1
lm_churn = sm.OLS(churn_df2['Churn'], churn_df2[['Children', 'Age', 'Income',
       'Outage_sec_perweek', 'Email', 'Contacts', 'Yearly_equip_failure',
       'Techie', 'Port_modem', 'Tablet', 'Phone', 'Multiple',
'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'PaperlessBilling', 'Tenure', 'MonthlyCharge',
       'Children.1', 'Age.1', 'Income.1', 'Outage_sec_perweek.1',
       'Email.1', 'Contacts.1', 'Yearly_equip_failure.1', 'Techie.1',
       'Port_modem.1', 'Tablet.1', 'Phone.1', 'Multiple.1',
'OnlineSecurity.1',
       'OnlineBackup.1', 'DeviceProtection.1', 'TechSupport.1',
       'StreamingTV.1', 'StreamingMovies.1', 'PaperlessBilling.1',
'Tenure.1',
       'MonthlyCharge.1', 'Marital_Divorced', 'Marital_Married',
       'Marital_Never Married', 'Marital_Separated', 'Marital_Widowed',
       'Contract_Month-to-month', 'Contract_One year', 'Contract_Two Year',
```

```
        'PaymentMethod_Bank Transfer(automatic)',
        'PaymentMethod_Credit Card (automatic)',
        'PaymentMethod_Electronic Check', 'PaymentMethod_Mailed Check',
        'Gender_Female', 'Gender_Male', 'Gender_Prefer not to answer',
        'InternetService_DSL', 'InternetService_Fiber Optic',
        'InternetService_None', 'Area_Rural', 'Area_Suburban', 'Area_Urban',
        'Employment_Full Time', 'Employment_Part Time', 'Employment_Retired',
        'Employment_Student', 'Employment_Unemployed','intercept']]).fit()
print(lm_churn.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Churn   R-squared:                       0.469
Model:                            OLS   Adj. R-squared:                  0.467
Method:                 Least Squares   F-statistic:                     220.1
Date:                Tue, 07 Sep 2021   Prob (F-statistic):               0.00
Time:                        21:36:26   Log-Likelihood:                -2842.7
No. Observations:               10000   AIC:                             5767.
Df Residuals:                    9959   BIC:                             6063.
Df Model:                          40
Covariance Type:            nonrobust
```

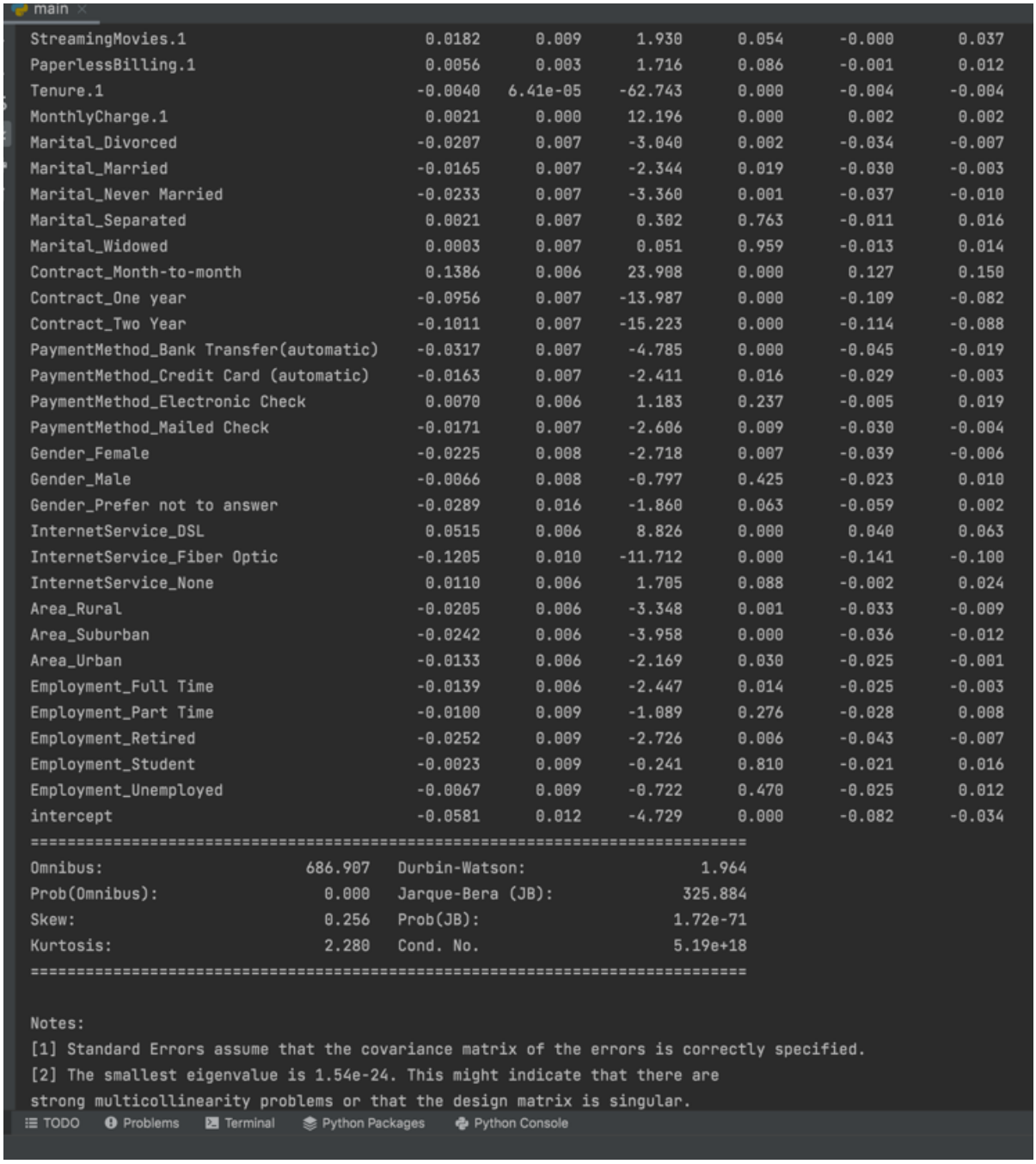
Picture 29: OLS Regression Results



	coef	std err	t	P>|t|	[0.025	0.975]
Children	0.0005	0.001	0.649	0.517	-0.001	0.002
Age	4.785e-05	8.97e-05	0.533	0.594	-0.000	0.000
Income	8.546e-08	6.52e-08	1.311	0.190	-4.23e-08	2.13e-07
Outage_sec_perweek	-0.0020	0.000	-7.388	0.000	-0.003	-0.001
Email	0.0004	0.001	0.741	0.459	-0.001	0.001
Contacts	0.0021	0.002	1.295	0.195	-0.001	0.005
Yearly_equip_failure	-0.0034	0.003	-1.356	0.175	-0.008	0.002
Techie	0.0379	0.005	7.787	0.000	0.028	0.047
Port_modem	0.0047	0.003	1.443	0.149	-0.002	0.011
Tablet	-0.0032	0.004	-0.914	0.361	-0.010	0.004
Phone	-0.0143	0.006	-2.458	0.014	-0.026	-0.003
Multiple	-0.0100	0.006	-1.558	0.119	-0.023	0.003
OnlineSecurity	-0.0114	0.003	-3.337	0.001	-0.018	-0.005
OnlineBackup	-0.0202	0.005	-4.021	0.000	-0.030	-0.010
DeviceProtection	-0.0091	0.004	-2.351	0.019	-0.017	-0.002
TechSupport	-0.0173	0.004	-4.358	0.000	-0.025	-0.010
StreamingTV	0.0177	0.008	2.255	0.024	0.002	0.033
StreamingMovies	0.0182	0.009	1.930	0.054	-0.000	0.037
PaperlessBilling	0.0056	0.003	1.716	0.086	-0.001	0.012
Tenure	-0.0040	6.41e-05	-62.743	0.000	-0.004	-0.004
MonthlyCharge	0.0021	0.000	12.196	0.000	0.002	0.002
Children.1	0.0005	0.001	0.649	0.517	-0.001	0.002
Age.1	4.785e-05	8.97e-05	0.533	0.594	-0.000	0.000
Income.1	8.546e-08	6.52e-08	1.311	0.190	-4.23e-08	2.13e-07
Outage_sec_perweek.1	-0.0020	0.000	-7.388	0.000	-0.003	-0.001
Email.1	0.0004	0.001	0.741	0.459	-0.001	0.001
Contacts.1	0.0021	0.002	1.295	0.195	-0.001	0.005
Yearly_equip_failure.1	-0.0034	0.003	-1.356	0.175	-0.008	0.002
Techie.1	0.0379	0.005	7.787	0.000	0.028	0.047
Port_modem.1	0.0047	0.003	1.443	0.149	-0.002	0.011
Tablet.1	-0.0032	0.004	-0.914	0.361	-0.010	0.004
Phone.1	-0.0143	0.006	-2.458	0.014	-0.026	-0.003
Multiple.1	-0.0100	0.006	-1.558	0.119	-0.023	0.003
OnlineSecurity.1	-0.0114	0.003	-3.337	0.001	-0.018	-0.005
OnlineBackup.1	-0.0202	0.005	-4.021	0.000	-0.030	-0.010
DeviceProtection.1	-0.0091	0.004	-2.351	0.019	-0.017	-0.002
TechSupport.1	-0.0173	0.004	-4.358	0.000	-0.025	-0.010
StreamingTV.1	0.0177	0.008	2.255	0.024	0.002	0.033
StreamingMovies.1	0.0182	0.009	1.930	0.054	-0.000	0.037

TODO Problems Terminal Python Packages Python Console

Picture 30: OLS Regression Results all dataset



main

StreamingMovies.1	0.0182	0.009	1.930	0.054	-0.000	0.037
PaperlessBilling.1	0.0056	0.003	1.716	0.086	-0.001	0.012
Tenure.1	-0.0040	6.41e-05	-62.743	0.000	-0.004	-0.004
MonthlyCharge.1	0.0021	0.000	12.196	0.000	0.002	0.002
Marital_Divorced	-0.0207	0.007	-3.040	0.002	-0.034	-0.007
Marital_Married	-0.0165	0.007	-2.344	0.019	-0.030	-0.003
Marital_Never Married	-0.0233	0.007	-3.360	0.001	-0.037	-0.010
Marital_Separated	0.0021	0.007	0.302	0.763	-0.011	0.016
Marital_Widowed	0.0003	0.007	0.051	0.959	-0.013	0.014
Contract_Month-to-month	0.1386	0.006	23.908	0.000	0.127	0.150
Contract_One year	-0.0956	0.007	-13.987	0.000	-0.109	-0.082
Contract_Two Year	-0.1011	0.007	-15.223	0.000	-0.114	-0.088
PaymentMethod_Bank Transfer(automatic)	-0.0317	0.007	-4.785	0.000	-0.045	-0.019
PaymentMethod_Credit Card (automatic)	-0.0163	0.007	-2.411	0.016	-0.029	-0.003
PaymentMethod_Electronic Check	0.0070	0.006	1.183	0.237	-0.005	0.019
PaymentMethod_Mailed Check	-0.0171	0.007	-2.606	0.009	-0.030	-0.004
Gender_Female	-0.0225	0.008	-2.718	0.007	-0.039	-0.006
Gender_Male	-0.0066	0.008	-0.797	0.425	-0.023	0.010
Gender_Prefer not to answer	-0.0289	0.016	-1.860	0.063	-0.059	0.002
InternetService_DSL	0.0515	0.006	8.826	0.000	0.040	0.063
InternetService_Fiber Optic	-0.1205	0.010	-11.712	0.000	-0.141	-0.100
InternetService_None	0.0110	0.006	1.705	0.088	-0.002	0.024
Area_Rural	-0.0205	0.006	-3.348	0.001	-0.033	-0.009
Area_Suburban	-0.0242	0.006	-3.958	0.000	-0.036	-0.012
Area_Urban	-0.0133	0.006	-2.169	0.030	-0.025	-0.001
Employment_Full Time	-0.0139	0.006	-2.447	0.014	-0.025	-0.003
Employment_Part Time	-0.0100	0.009	-1.089	0.276	-0.028	0.008
Employment_Retired	-0.0252	0.009	-2.726	0.006	-0.043	-0.007
Employment_Student	-0.0023	0.009	-0.241	0.810	-0.021	0.016
Employment_Unemployed	-0.0067	0.009	-0.722	0.470	-0.025	0.012
intercept	-0.0581	0.012	-4.729	0.000	-0.082	-0.034

Omnibus:	686.907	Durbin-Watson:	1.964
Prob(Omnibus):	0.000	Jarque-Bera (JB):	325.884
Skew:	0.256	Prob(JB):	1.72e-71
Kurtosis:	2.280	Cond. No.	5.19e+18

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.54e-24. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

TODO Problems Terminal Python Packages Python Console

Picture 31: Cont

Regression Equation I:

Y = -0.0581+ 0.0005*Children + 4.785e-05*Age + 8.546e-08*Income - 0.0020*Outage_sec_perweek + 0.0004*Email + 0.0021*Contacts - 0.0034*Equip_failure + 0.0379*Techie + 0.0047*Port_modem -0.0032*Tablet -0.0143*Phone -0.0100*Multiple - 0.0114*OnlineSecurity - 0.0202*OnlineBackup -0.0091*DeviceProtection -

0.0173*TechSupport + 0.0177*StreamingTV + 0.0182*StreamingMovies + 0.0056*PaperlessBilling -0.0040*Tenure + 0.0021*MonthlyCharge + 0.0005*Children.1 + 4.785e-05*Age.1 + 8.546e-08*Income.1 + -0.0020* Outage_sec_perweek.1 + 0.0004*Email.1 + 0.0021*Contacts.1 -0.0034*Yearly_equip_failure.1 + 0.0379*Techie.1 + 0.0047*Port_modem.1 - 0.0032*Tablet.1 -0.0143*Phone.1-0.0100*Multiple.1 - 0.0114*OnlineSecurity.1 -0.0202*OnlineBackup.1 -0.0091*DeviceProtection.1 - 0.0173*TechSupport.1 + 0.0177 *StreamingTV.1 + 0.0182*StreamingMovies.1 + 0.0056*PaperlessBilling.1 -0.0040*Tenure.1 + 0.0021*MonthlyCharge.1- 0.0207*Marital_Divorced-0.0165*Marital_Married -0.0233*Marital_Never Married+ 0.0021*Marital_Separated+ 0.0003*Marital_Widowed + 0.1386*Contract_Month-to-month - 0.0956*Contract_One year-0.1011*Contract_Two Year-0.0317*PaymentMethod_Bank Transfer(automatic) -0.0163*PaymentMethod_Credit Card (automatic) + 0.0070PaymentMethod_Electronic Check -0.0171*PaymentMethod_Mailed Check - 0.0225*Gender_Female-0.0066*Gender_Male-0.0289*Gender_Prefer not to answer + 0.0515*InternetService_DSL -0.1205 *InternetService_Fiber Optic + 0.0110*InternetService_None-0.0205*Area_Rural -0.0242*Area_Suburban- 0.0133*Area_Urban -0.0139*Employment_Full Time -0.0100*Employment_Part Time -0.0252 *Employment_Retired -0.0023*Employment_Student -0.0067* Employment_Unemployed

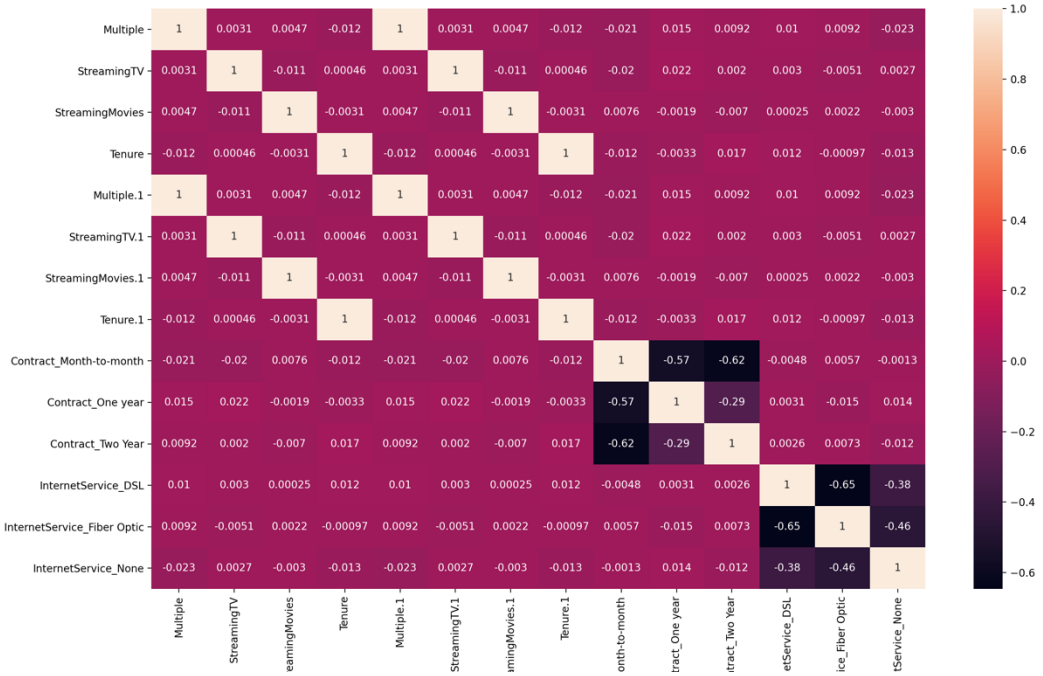
Using all variables in the dataset we can explain 47% of the variance, which is not great. The condition number is very large which indicates multicollinearity. I am going to decrease the dataset to the variables that I previously found that matter the most except MonthlyCharges since the previous heatmap already showed some correlations between it and other variables. Tenure, contract, streaming services, Multiple and internet service.

PART D2. Justify a statistically based variable selection procedure and a model evaluation metric to reduce the initial model in a way that aligns with the research question.

Decreasing the variables to:

```
churn_reduced_analysis = churn_df2[['Multiple', 'StreamingTV',
'StreamingMovies', 'Tenure',
           'Multiple.1', 'StreamingTV.1', 'StreamingMovies.1', 'Tenure.1',
'Contract_Month-to-month', 'Contract_One year',
           'Contract_Two Year', 'InternetService_DSL',
           'InternetService_Fiber Optic', 'InternetService_None']]
```

Plotting these variables into a correlation matrix we can see that there's no dependency besides contracts with each other.

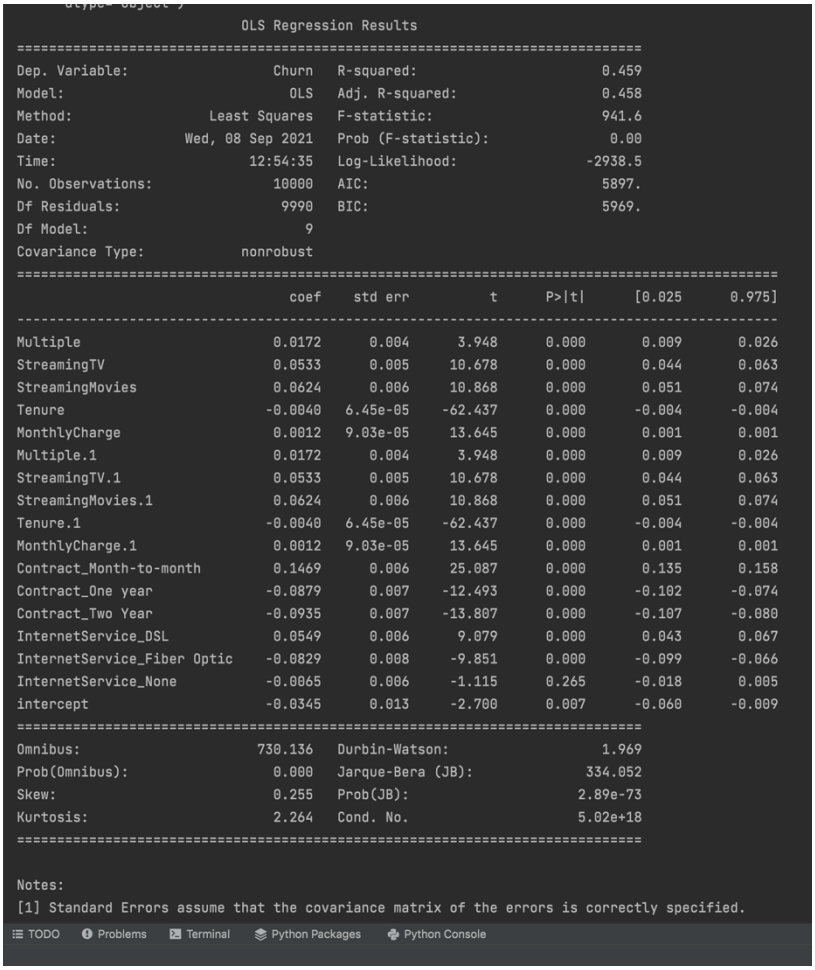


Picture 32: Correlation Matrix of Reduced Set of Variables

PART D3. Provide a reduced multiple regression model that includes *both* categorical and continuous variables.

With the bar graphs we can see which variables could possibly influence the churn. From that analysis we can filter the relevant variables and build a model with them only. I am including Tenure, numerical and the rest categorical.

```
churn_df2['intercept'] = 1
lm_churn = sm.OLS(churn_df2['Churn'], churn_df2[['Multiple', 'StreamingTV',
'StreamingMovies', 'Tenure','Multiple.1', 'StreamingTV.1',
'StreamingMovies.1', 'Tenure.1', 'Contract_Month-to-month', 'Contract_One
year', 'Contract_Two Year', 'InternetService_DSL', 'InternetService_Fiber
Optic', 'InternetService_None','intercept']]).fit()
print(lm_churn.summary())
```



OLS Regression Results

Dep. Variable:	Churn	R-squared:	0.459
Model:	OLS	Adj. R-squared:	0.458
Method:	Least Squares	F-statistic:	941.6
Date:	Wed, 08 Sep 2021	Prob (F-statistic):	0.00
Time:	12:54:35	Log-Likelihood:	-2938.5
No. Observations:	10000	AIC:	5897.
Df Residuals:	9990	BIC:	5969.
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Multiple	0.0172	0.004	3.948	0.000	0.009	0.026
StreamingTV	0.0533	0.005	10.678	0.000	0.044	0.063
StreamingMovies	0.0624	0.006	10.868	0.000	0.051	0.074
Tenure	-0.0040	6.45e-05	-62.437	0.000	-0.004	-0.004
MonthlyCharge	0.0012	9.03e-05	13.645	0.000	0.001	0.001
Multiple.1	0.0172	0.004	3.948	0.000	0.009	0.026
StreamingTV.1	0.0533	0.005	10.678	0.000	0.044	0.063
StreamingMovies.1	0.0624	0.006	10.868	0.000	0.051	0.074
Tenure.1	-0.0040	6.45e-05	-62.437	0.000	-0.004	-0.004
MonthlyCharge.1	0.0012	9.03e-05	13.645	0.000	0.001	0.001
Contract_Month-to-month	0.1469	0.006	25.087	0.000	0.135	0.158
Contract_One year	-0.0879	0.007	-12.493	0.000	-0.102	-0.074
Contract_Two Year	-0.0935	0.007	-13.807	0.000	-0.107	-0.080
InternetService_DSL	0.0549	0.006	9.079	0.000	0.043	0.067
InternetService_Fiber Optic	-0.0829	0.008	-9.851	0.000	-0.099	-0.066
InternetService_None	-0.0065	0.006	-1.115	0.265	-0.018	0.005
intercept	-0.0345	0.013	-2.700	0.007	-0.060	-0.009

Omnibus:	730.136	Durbin-Watson:	1.969
Prob(Omnibus):	0.000	Jarque-Bera (JB):	334.052
Skew:	0.255	Prob(JB):	2.89e-73
Kurtosis:	2.264	Cond. No.	5.02e+18

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

TODO Problems Terminal Python Packages Python Console

Picture 33: Regression Model with reduced variables

We pretty much got the same results with way less variables!

The new regression equation is: (of course the dummy variables have the same coef)

Y = -0.0345 + 0.0172* Multiple + 0.0533* StreamingTV + 0.0624*StreamingMovies - 0.0040*Tenure + 0.0172* Multiple.1 + 0.0533*StreamingTV.1 + 0.0624*StreamingMovies.1 – 0.0040*Tenure.1 + 0.1469*Contract_Month-to-Month – 0.0879*Contract_One year – 0.0935*Contract_Two year + 0.0549*InternetService_DSL – 0.0829*InternetService_Fiber Optic – 0.0065*InternetService_None

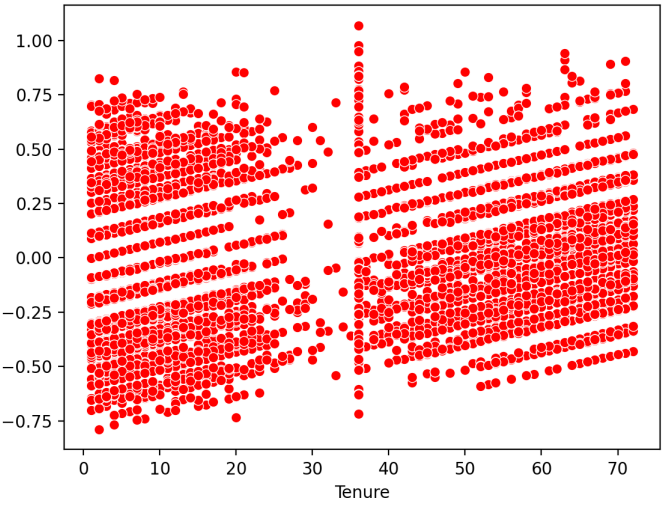
PART E - Analyze the data set using your reduced multiple regression model by doing the following:

PART E1. Explain your data analysis process by comparing the initial and reduced multiple regression models, including the following elements:

- **the logic of the variable selection technique: explained above**

- **the model evaluation metric: explained above**
- **a residual plot**

```
#Residual Plot
churn_df = pd.read_csv('prepared_churn_data.csv')
churn_df['intercept'] = 1
residuals = churn_df['Churn'] -
lm_churn_reduced.predict(churn_df[['Multiple', 'StreamingTV',
'StreamingMovies', 'Tenure',

'Contract_Month-to-month', 'Contract_One year','Contract_Two Year',
'InternetService_DSL','InternetService_Fiber Optic', 'InternetService_None',
'intercept']])
sns.scatterplot(x=churn_df['Tenure'],y=residuals,color='red')
plt.show()
```



Picture 34: Residual Plot

PART E2. Provide the output and *any* calculations of the analysis you performed, including the model's residual error. Explained above.

PART E3. Provide the code used to support the implementation of the multiple regression models. It is along the document

Part V: DATA SUMMARY AND IMPLICATIONS

PART F. SUMMARIZE YOUR FINDINGS AND ASSUMPTIONS BY DOING THE FOLLOWING:

PART F1. DISCUSS THE RESULTS OF YOUR DATA ANALYSIS, INCLUDING THE FOLLOWING ELEMENTS:

- a regression equation for the reduced model: **Y = -0.0345 + 0.0172* Multiple + 0.0533* StreamingTV + 0.0624*StreamingMovies - 0.0040*Tenure – 0.0040*Tenure.1 + 0.1469*Contract_Month-to-Month – 0.0879*Contract_One year – 0.0935*Contract_Two year + 0.0549*InternetService_DSL – 0.0829*InternetService_Fiber Optic – 0.0065*InternetService_None**

- an interpretation of coefficients of the statistically significant variables of the model: for every 1 unit of

Multiple: customer churn will increase **0.0172 units**

StreamingTV: customer churn will increase **0.0533 units**

StreamingMovies: customer churn will increase **0.0624 units**

Tenure: customer churn will decrease **0.0040 units**

Contract_Month-to-Month: customer churn will increase **0.1469 units**

Contract_One year: customer churn will decrease **0.0879 units**

Contract_Two year: customer churn will decrease **0.0935 units**

InternetService_DSL: customer churn will increase **0.0549 units**

InternetService_Fiber Optic: customer churn will decrease **0.0829 units**

InternetService_None: customer churn will decrease **0.0065 units**

- the statistical and practical significance of the model

p-values: are statistically significant at 0 for all variables except InternetService_none. We could exclude this last variable from the model.

- the limitations of the data analysis: as mentioned before the churn rate is smaller compared with customers who haven't churned. Maybe a bigger ration in between those who churn and those who don't, would provide a more accurate model.

PART F2. RECOMMEND A COURSE OF ACTION BASED ON YOUR RESULTS

Based on the results we obtained with the regression equation, the course of action would be to sign up customers with one or two year contracts, since DSL internet has a positive sign, try to sign up customers with Fiber Optic. Maybe that would be good enough if customers want to stream movie or TV, since DSL is slower.

PART VI – DEMONSTRATION

PART G – VIDEO

The Panopto video will be uploaded here:
https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=1811f033-341f-48b6-9591-ad9e0126edd8&edit=true

PART H – THIRD PARTY CODE
N/A

PART I – SOURCES

[1] (2017, March 30th) SRINIVASAN, VITTHAL Understanding and Applying Logistic Regression https://app.pluralsight.com/library/courses/numerical-optimization-techniques/table-of-contents

[2] (2020, Aug, 8th) Deval, Swati https://www.mygreatlearning.com/blog/mean-square-error-explained/ Mean Squared Error – Explained | What is Mean Square Error?

[3] 2021, July BISCHOFF, BIANCA D206 Assessment Report

[4] 2020, March 6th SETHI, Alakh One-hot Encoding Versus Label Encoding using Scikit-Learn https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/

[5]2021, May 2nd SHARMA, Akhil Predictive Analysis Using Multiple Linear Regression https://medium.com/data-science-on-customer-churn-data/predictive-analysis-using-multiple-linear-regression-b6b3b79b36b6