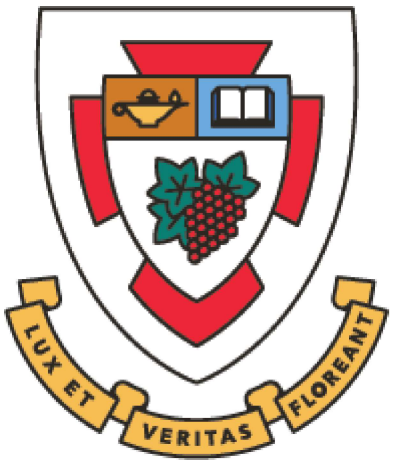


THE UNIVERSITY OF
WINNIPEG

Applied Statistics for Data Science

Final Project

By

Ayotunde Oyewole - 3177106

Content

Click any of the headers below to jump to the section.

Summary

Two models were developed as part of this project, namely a linear model to predict customer value and a logistic model to classify customers based on their likelihood of churn. The data used was obtained from the UCI archives. The models performed optimally, with scores of 84% and 81% respectively.

1.0 Introduction

This project is completed as part of the requirement for the Applied Statistics for Data Science course. It includes an analysis of the customer churn of an Iranian telecom company. An exploratory analysis of the data will be carried out from two perspectives, namely the quantitative analysis and the qualitative analysis. Subsets of the dataset will also be extracted to perform regression analysis and classification.

1.1 Dataset Selection

The dataset selected for this project is from the UCI archives. The link to the dataset is shown below.

link to dataset - https://archive.ics.uci.edu/dataset/563/iranian+churn+dataset

The dataset describes the churn data of customers in an Iranian telecom company over a 12 month period. There are thirteen variables with more than 3000 records in the dataset. Only some subsets of the dataset will be extracted for this analysis.

1.2 Definition of Variables

The following variables are included in the dataset.

Anonymous Customer ID
Call Failures: number of call failures
Complains: binary (0: No complaint, 1: complaint)
Subscription Length: total months of subscription
Charge Amount: Ordinal attribute (0: lowest amount, 9: highest amount)
Seconds of Use: total seconds of calls
Frequency of use: total number of calls
Frequency of SMS: total number of text messages
Distinct Called Numbers: total number of distinct phone calls
Age Group: ordinal attribute (1: younger age, 5: older age)
Tariff Plan: binary (1: Pay as you go, 2: contractual)
Status: binary (1: active, 2: non-active)
Churn: binary (1: churn, 0: non-churn) - Class label
Customer Value: The calculated value of customer

2.0 Data Analysis

In [1]:
```
# All libraries for the project will be imported here

#Data manipulation libraries
import pandas as pd
import numpy as np
import math

#Visualization libraries
import matplotlib.pyplot as plt
import plotly.express as px
from statsmodels.graphics.regressionplots import abline_plot

# Modeling Libraries
import statsmodels.formula.api as sm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
```

The whole data will be read to a dataframe and the required subset of the dataframe will be extracted to be used for quantitative, and then qualitative exploration.

In [2]:
```
df = pd.read_csv('Customer Churn.csv')
df
```

Out[2]:

	Call Failure	Complains	Subscription Length	Charge Amount	Seconds of Use	Frequency of use	Frequency of SMS	Distinct Called Numbers	Age Group	Tariff Plan	Status	Age	Customer Value	Churn
0	8	0	38	0	4370	71	5	17	3	1	1	30	197.640	0
1	0	0	39	0	318	5	7	4	2	1	2	25	46.035	0
2	10	0	37	0	2453	60	359	24	3	1	1	30	1536.520	0
3	10	0	38	0	4198	66	1	35	1	1	1	15	240.020	0
4	3	0	38	0	2393	58	2	33	1	1	1	15	145.805	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3145	21	0	19	2	6697	147	92	44	2	2	1	25	721.980	0
3146	17	0	17	1	9237	177	80	42	5	1	1	55	261.210	0
3147	13	0	18	4	3157	51	38	21	3	1	1	30	280.320	0
3148	7	0	11	2	4695	46	222	12	3	1	1	30	1077.640	0
3149	8	1	11	2	1792	25	7	9	3	1	1	30	100.680	1

3150 rows × 14 columns

2.1 Quantitative Analysis

The following columns will be used for the quantitative analysis:

1. Subscription Length
2. Seconds of Use
3. Frequency of Use
4. Frequency of SMS
5. Customer Value

The customer value is the response variable, while the other four (4) variables are the explanatory variables.

In [3]:
```
# Extractive the variables for quantitative analysis
df_quant = df[['Subscription  Length', 'Seconds of Use', 'Frequency of use', 'Frequency of SMS', 'Customer Value']]
# Removing the extra space in the column name for subscription length
df_quant = df_quant.rename(columns = {'Subscription  Length':'Subscription Length'})
df_quant
```

Out[3]:

	Subscription Length	Seconds of Use	Frequency of use	Frequency of SMS	Customer Value
0	38	4370	71	5	197.640
1	39	318	5	7	46.035
2	37	2453	60	359	1536.520
3	38	4198	66	1	240.020
4	38	2393	58	2	145.805
...	...	...	...	...	...
3145	19	6697	147	92	721.980
3146	17	9237	177	80	261.210
3147	18	3157	51	38	280.320
3148	11	4695	46	222	1077.640
3149	11	1792	25	7	100.680

3150 rows × 5 columns

In [4]:
```
# obtaining basic information on the extracted quantitative data
print(df_quant.shape, '\n')
df_quant.info()
```

```
(3150, 5)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3150 entries, 0 to 3149
Data columns (total 5 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Subscription Length  3150 non-null   int64  
 1   Seconds of Use       3150 non-null   int64  
 2   Frequency of use     3150 non-null   int64  
 3   Frequency of SMS     3150 non-null   int64  
 4   Customer Value       3150 non-null   float64
dtypes: float64(1), int64(4)
memory usage: 123.2 KB
```

2.1.1 Frequency Tables

First the frequency distribution of each variable will be examined and an histogram will be plotted for each variable. Since there are 5 different variables, a function will be defined to construct the frequency tables and plot the histograms. This will enhance code reusability.

For simplicity, we want 10 bins for each of the variable. Thus a bin of 10 is used in the function. I initially attempted to use a range as prescribed in the class notes but it did not capture all the frequencies for all the variables when I applied the function to each of them.

Although a hard coded bin size of 10 resulted in floats in the bins, rather than whole number integers, this is not really a problem as it does not diminish understandability. Additionally, the histograms are subsequently plotted to aid visualization.

In [5]:
```
def freq_table_gen(series):
    # Takes a variable as argument and returns the frequency distribution of elements in that series
    # Returns a frequency table
    smallest = series.min()
    largest = series.max()
    steps = math.floor((largest-smallest)/10)

    bins = 10
    categories = pd.cut(series, bins, include_lowest=True, right = False)
    Frequency = categories.value_counts().sort_index()
    Relative_Frequency = (Frequency/Frequency.sum()).round(3)

    freq_table = pd.DataFrame({'Frequency': Frequency,
                               'Relative_Frequency':Relative_Frequency})
    return freq_table
```

In [6]:
```
# Construct the frequency table for subscription length
sub_length_freq_table = freq_table_gen(df_quant['Subscription Length'])
sub_length_freq_table
```

Out[6]:

Subscription Length	Frequency	Relative_Frequency
[3.0, 7.4)	45	0.014
[7.4, 11.8)	76	0.024
[11.8, 16.2)	129	0.041
[16.2, 20.6)	120	0.038
[20.6, 25.0)	131	0.042
[25.0, 29.4)	286	0.091
[29.4, 33.8)	457	0.145
[33.8, 38.2)	1192	0.378
[38.2, 42.6)	577	0.183
[42.6, 47.044)	137	0.043

In [7]:
```
# Construct the frequency table for Seconds of use
sec_of_use_freq_table = freq_table_gen(df_quant['Seconds of Use'])
sec_of_use_freq_table
```

Out[7]:

Seconds of Use	Frequency	Relative_Frequency
[0.0, 1709.0)	935	0.297
[1709.0, 3418.0)	809	0.257
[3418.0, 5127.0)	395	0.125
[5127.0, 6836.0)	336	0.107
[6836.0, 8545.0)	184	0.058
[8545.0, 10254.0)	135	0.043
[10254.0, 11963.0)	99	0.031
[11963.0, 13672.0)	31	0.010
[13672.0, 15381.0)	110	0.035
[15381.0, 17107.09)	116	0.037

In [8]:
```
# Construct the frequency table for Frequency of use
freq_of_use_freq_table = freq_table_gen(df_quant['Frequency of use'])
freq_of_use_freq_table
```

Out[8]:

Frequency of use	Frequency	Relative_Frequency
[0.0, 25.5)	740	0.235
[25.5, 51.0)	753	0.239
[51.0, 76.5)	526	0.167
[76.5, 102.0)	435	0.138
[102.0, 127.5)	212	0.067
[127.5, 153.0)	128	0.041
[153.0, 178.5)	169	0.054
[178.5, 204.0)	61	0.019
[204.0, 229.5)	53	0.017
[229.5, 255.255)	73	0.023

In [9]:
```
# Construct the frequency table for Frequency of SMS
freq_of_SMS_freq_table = freq_table_gen(df_quant['Frequency of SMS'])
freq_of_SMS_freq_table
```

Out[9]:

Frequency of SMS	Frequency	Relative_Frequency
[0.0, 52.2)	2198	0.698
[52.2, 104.4)	214	0.068
[104.4, 156.6)	147	0.047
[156.6, 208.8)	223	0.071
[208.8, 261.0)	76	0.024
[261.0, 313.2)	101	0.032
[313.2, 365.4)	50	0.016
[365.4, 417.6)	77	0.024
[417.6, 469.8)	16	0.005
[469.8, 522.522)	48	0.015

In [10]:
```
# Construct the frequency table for Customer Value
cust_val_freq_table = freq_table_gen(df_quant['Customer Value'])
cust_val_freq_table
```

Out[10]:

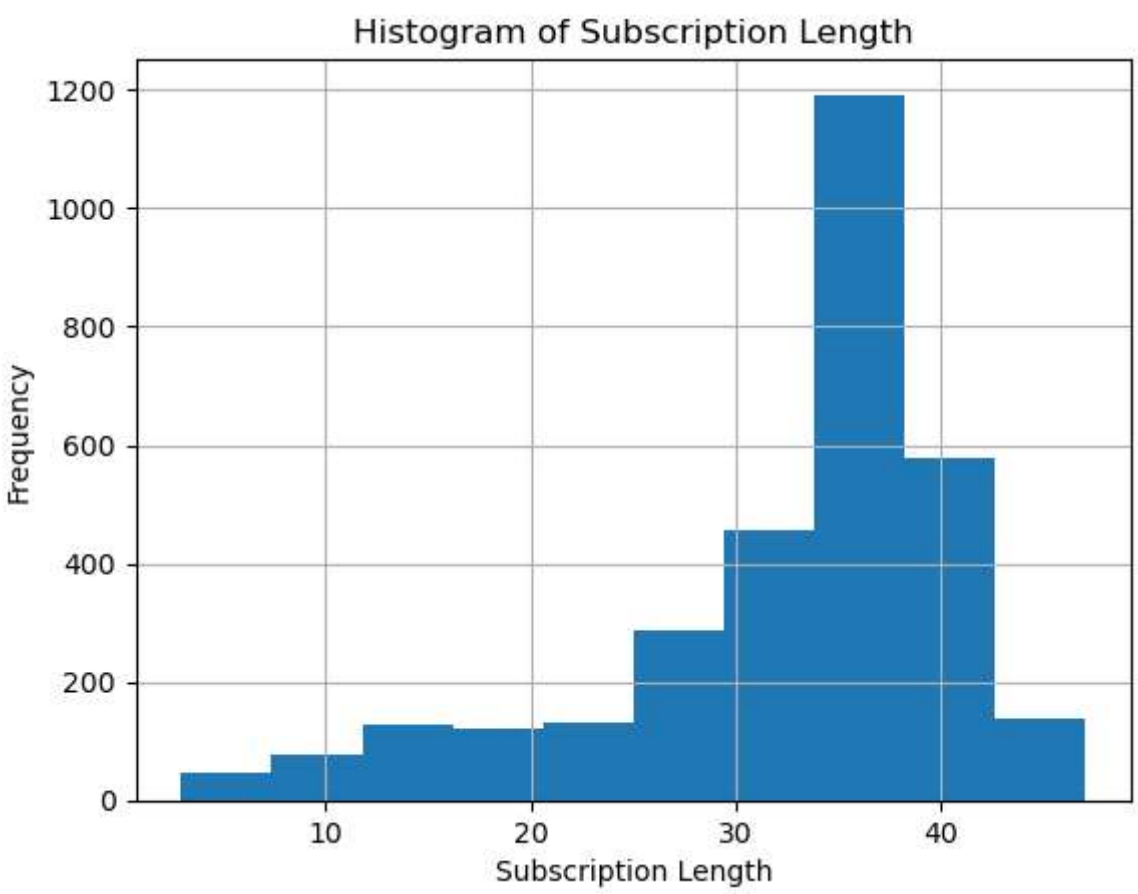
Customer Value	Frequency	Relative_Frequency
[0.0, 216.528)	1509	0.479
[216.528, 433.056)	588	0.187
[433.056, 649.584)	114	0.036
[649.584, 866.112)	273	0.087
[866.112, 1082.64)	272	0.086
[1082.64, 1299.168)	86	0.027
[1299.168, 1515.696)	111	0.035
[1515.696, 1732.224)	76	0.024
[1732.224, 1948.752)	39	0.012
[1948.752, 2167.445)	82	0.026

2.1.2 Histogram

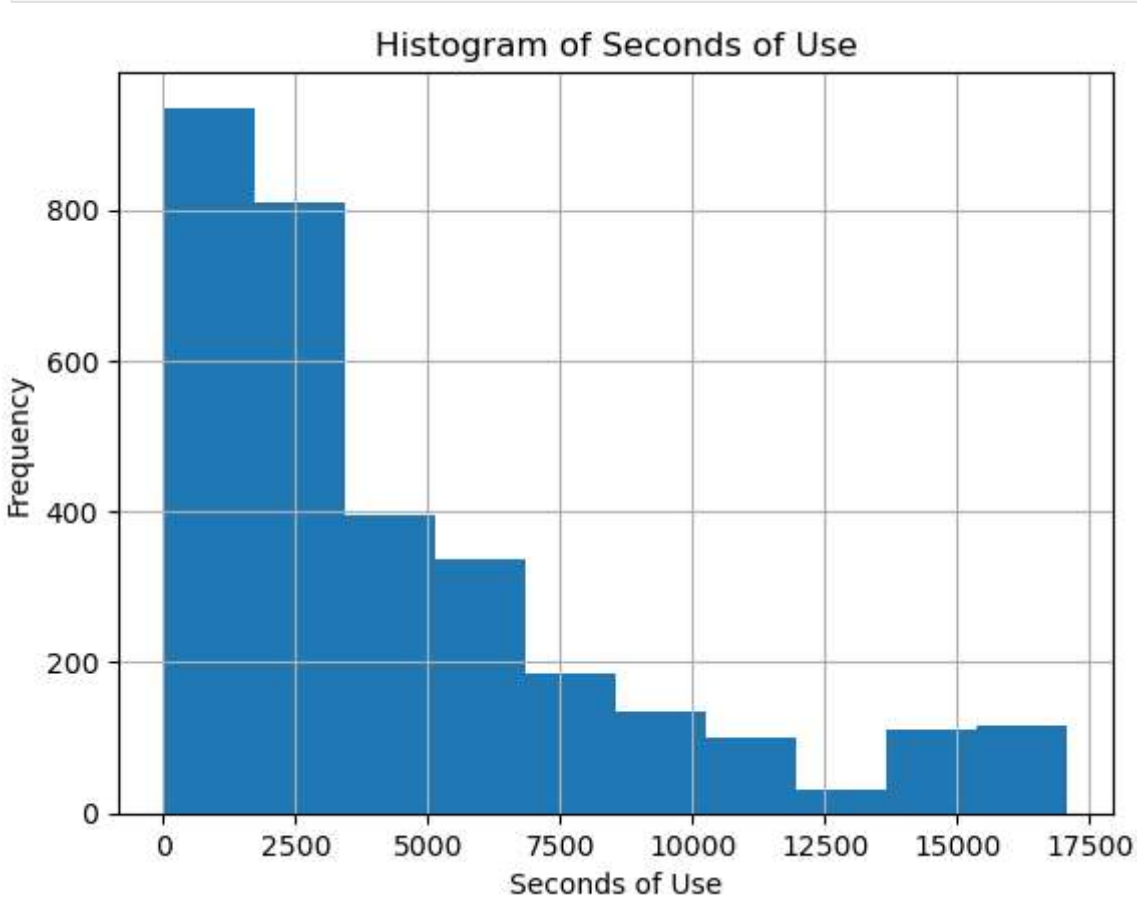
Again, because five different histograms will be plotted, a function will be used. I attempted using matplotlib.pyplot.subplot to plot the 5 histograms on a single plot. Each plot was too small to offer clarity of information they were meant to convey. Therefore, the histograms are plotted one after another.

In [11]:
```
def histogram_plotter(series, xlabel, title, bins = 10):
    series.hist(bins = bins)
    plt.ylabel('Frequency')
    plt.xlabel(xlabel)
    plt.title(title)
    plt.show()
```

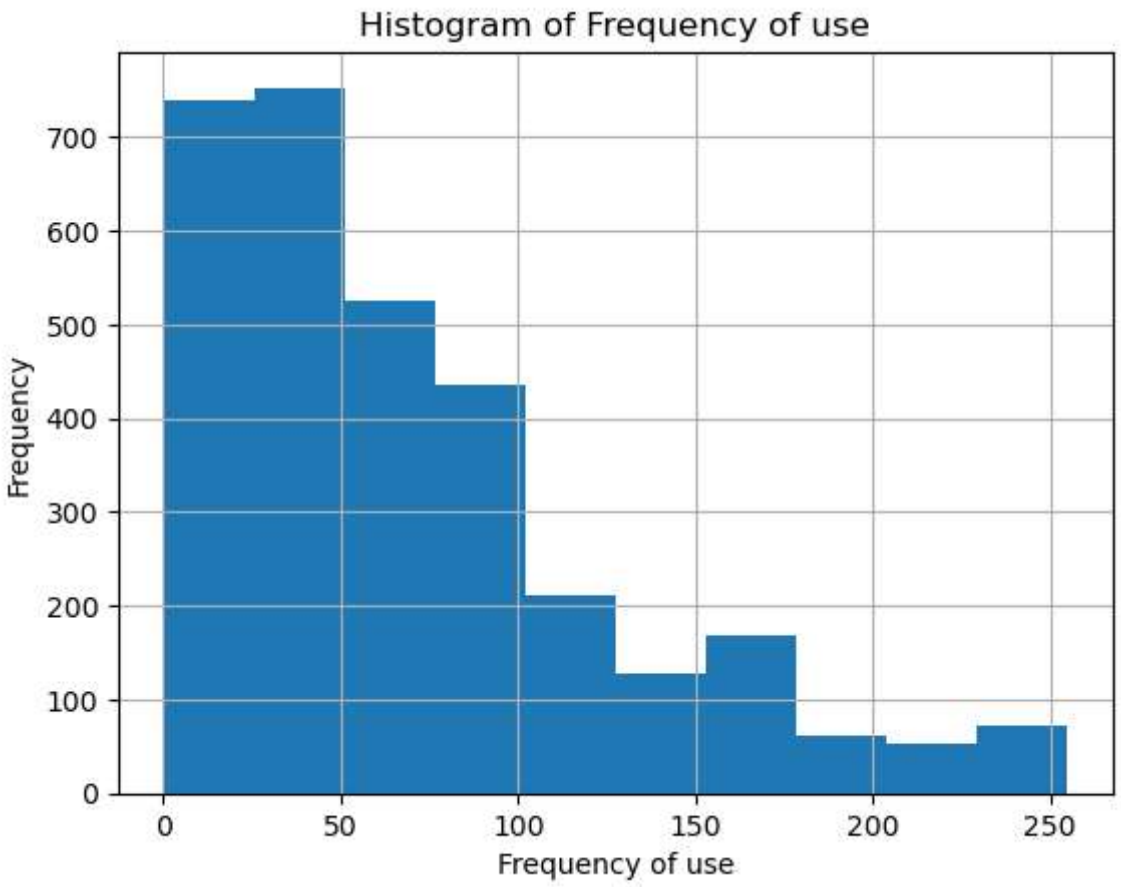
In [12]:
```
histogram_plotter(series = df_quant['Subscription Length'],
                  xlabel = 'Subscription Length',
                  title = 'Histogram of Subscription Length')
```



In [13]:
```
histogram_plotter(series = df_quant['Seconds of Use'],
                  xlabel = 'Seconds of Use',
                  title = 'Histogram of Seconds of Use')
```

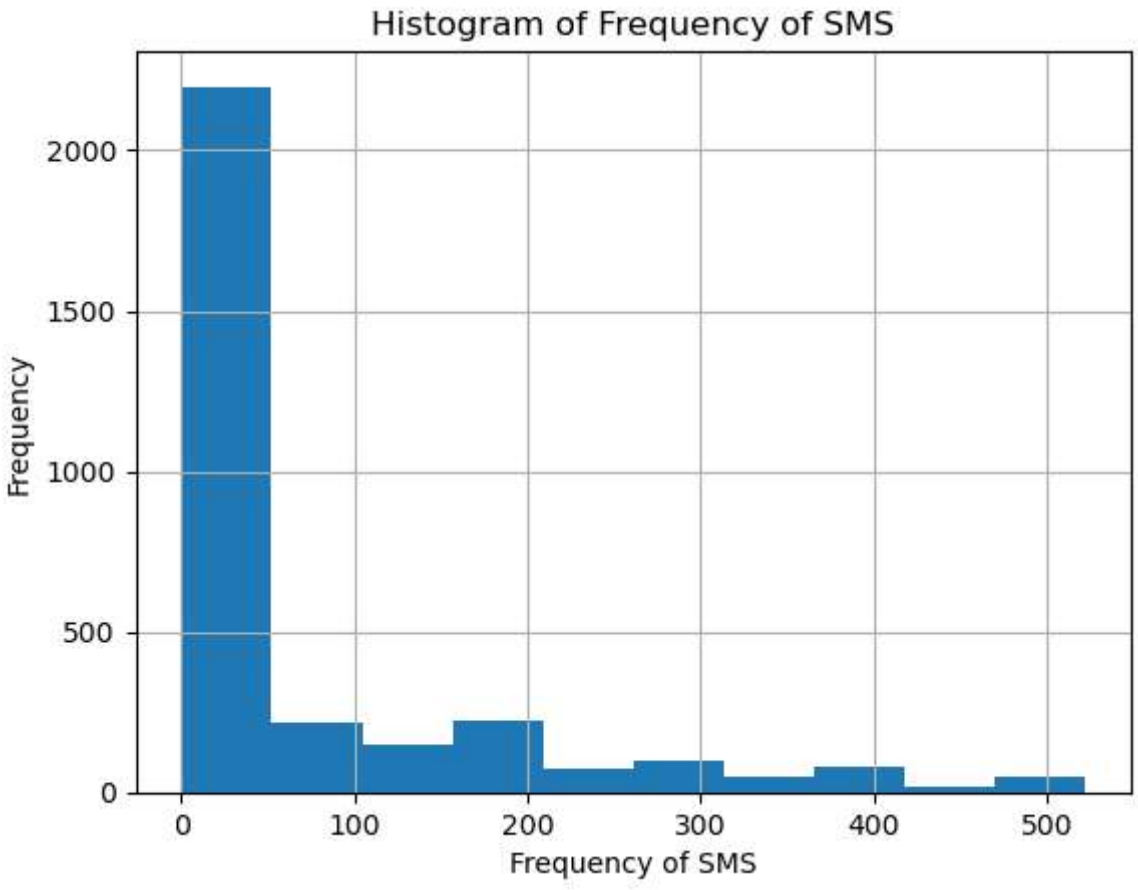


In [14]:
```
histogram_plotter(series = df_quant['Frequency of use'],
                  xlabel = 'Frequency of use',
                  title = 'Histogram of Frequency of use')
```

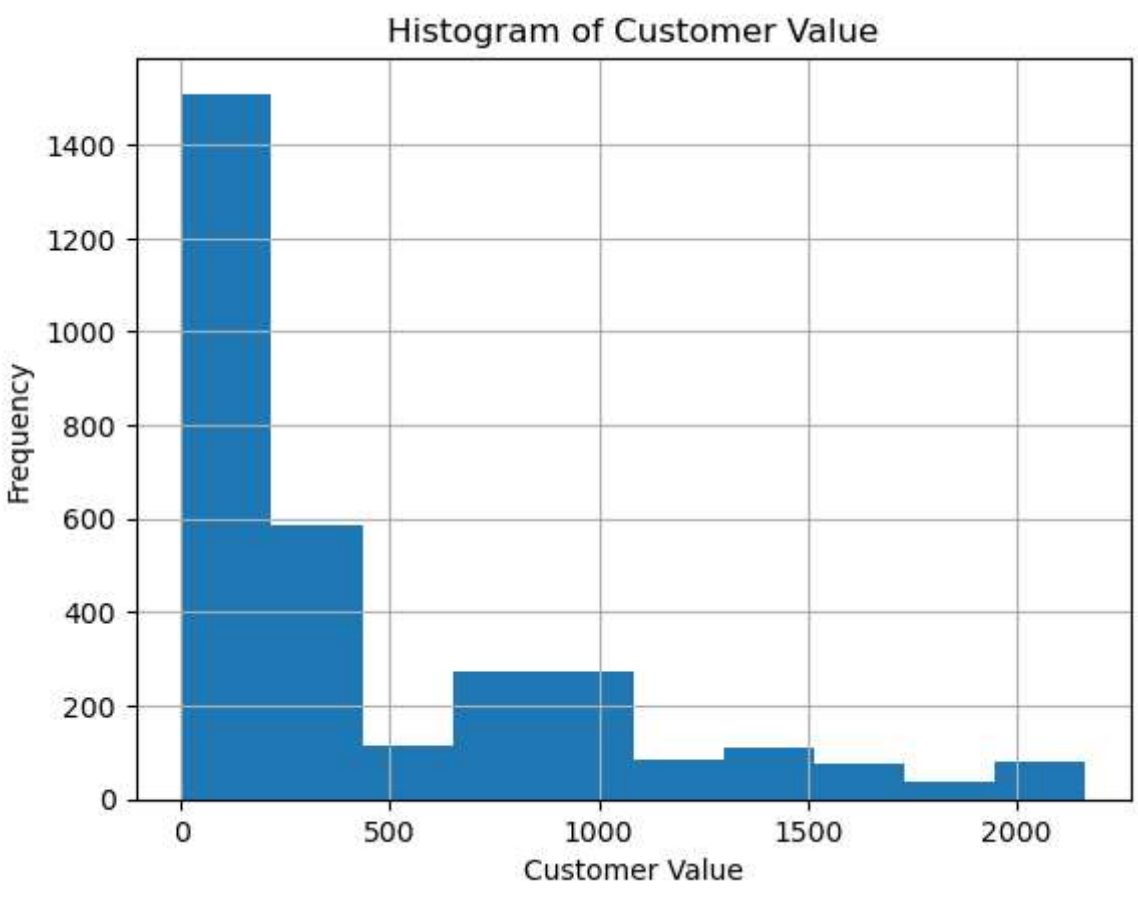


In [15]:
```
histogram_plotter(series = df_quant['Frequency of SMS'],
                  xlabel = 'Frequency of SMS',
```

```
                  title = 'Histogram of Frequency of SMS')
```



In [16]:
```
histogram_plotter(series = df_quant['Customer Value'],
                  xlabel = 'Customer Value',
                  title = 'Histogram of Customer Value')
```



2.1.3 Summary Measures

Rather than retrieving the summary statistics for each variable one at a time, the statistics across all five variables are tabulated below. This enhances comparison among the variables.

In [17]:
```
df_quant.describe()
```

Out[17]:

	Subscription Length	Seconds of Use	Frequency of use	Frequency of SMS	Customer Value
count	3150.000000	3150.000000	3150.000000	3150.000000	3150.000000
mean	32.541905	4472.459683	69.460635	73.174921	470.972916
std	8.573482	4197.908687	57.413308	112.237560	517.015433
min	3.000000	0.000000	0.000000	0.000000	0.000000
25%	30.000000	1391.250000	27.000000	6.000000	113.801250
50%	35.000000	2990.000000	54.000000	21.000000	228.480000
75%	38.000000	6478.250000	95.000000	87.000000	788.388750
max	47.000000	17090.000000	255.000000	522.000000	2165.280000

The table above concisely gives the summary measures for all the quantitative variables under consideration.

Count:

Clearly all the variables have the same number of records. There are no missing data in the dataframe.

Mean and Standard Deviation:

The mean for subscription length is 32.5 with a standard deviation of 8.5. On the far end of the spectrum, the mean for seconds of use is 4,472.5 with an equally sizeable standard deviation of 4,197. This is understandable since the range of seconds of use spreads from zero all the way to more than 17,000. I would consider converting seconds of use to hours or minutes of use to keep this variable from having an overbloated effect on the model. The other three variables also have large spreads around their mean with Frequency of SMS and Customer Value having standard deviations greater than their mean. This indicates that the data is widely dispersed.

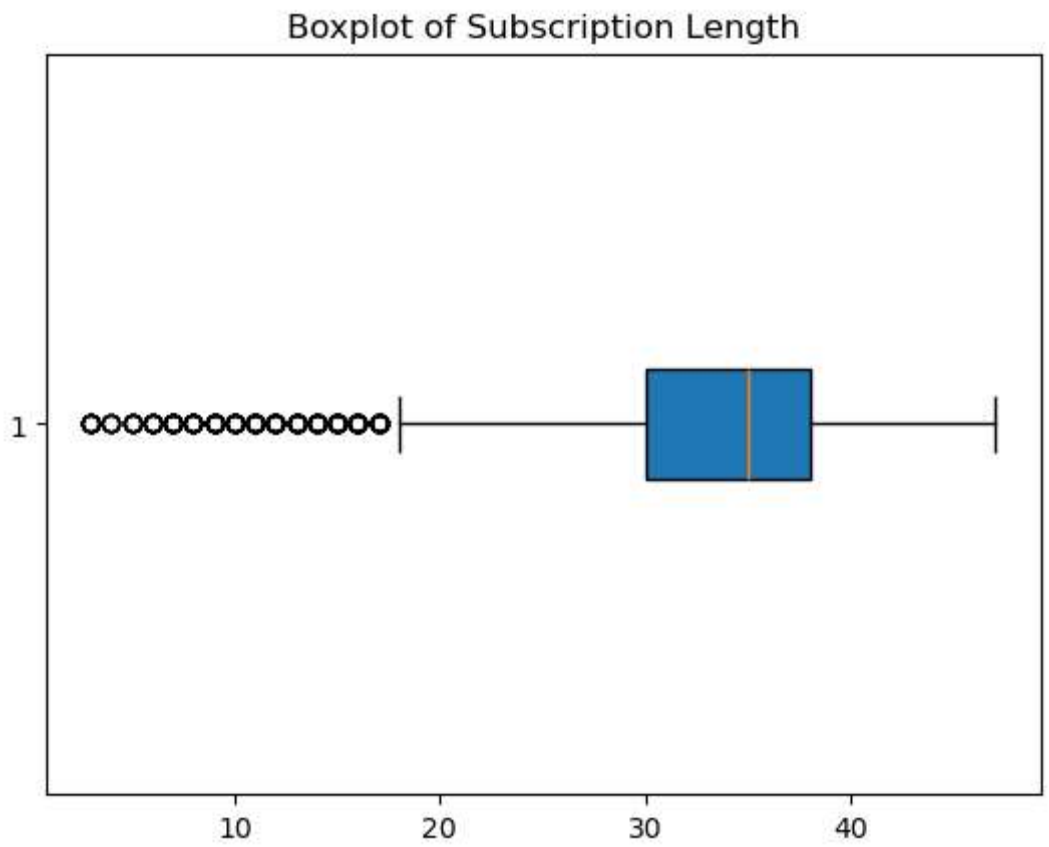
Five Point Summary:

It is noteworthy that 50% of the subsciption length is below 35 (Q2), while the remaining 12 units of length is spread above 50%. This gives some indication about the skewness of the data which will be explored with the boxplots. A converse distribution is observable for the 4 other fields, where more of the data is within the first quartile (Q1). Finally, only subscription length has a minimum value of 3, while the range of the other variables start from zero (0).

2.1.4 Boxplots and Skewness

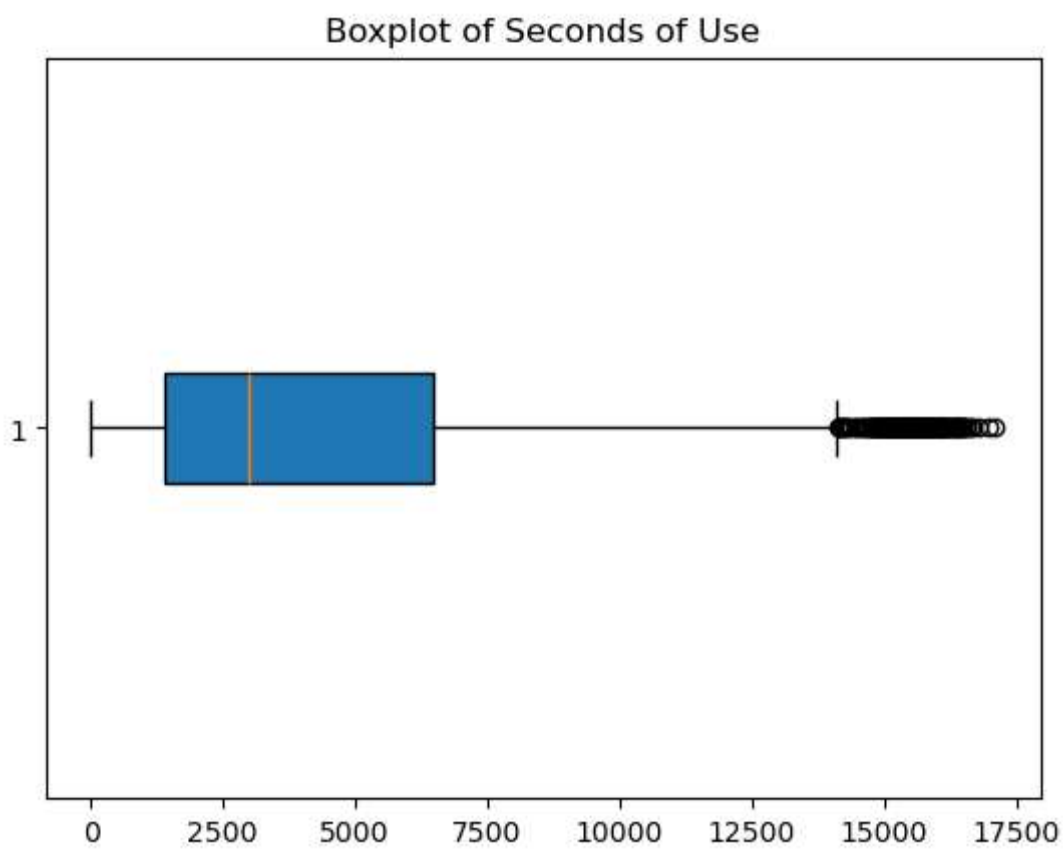
The histograms and the 5 point summaries for the variables already gave some indication about their skewness. This will be confirmed using the boxplots.

In [18]:
```
plt.boxplot(df_quant['Subscription Length'], patch_artist = True, vert = False)
plt.title('Boxplot of Subscription Length');
```



From the Subscription Length boxplot above, the first obvious item is that there are a number of outliers and they exist at the lower spectrum of the data range.
Also, the median line is to the right of the box plot, while the right tail appears longer than the left tail. This indicates that the subscription length is **left skewed.**

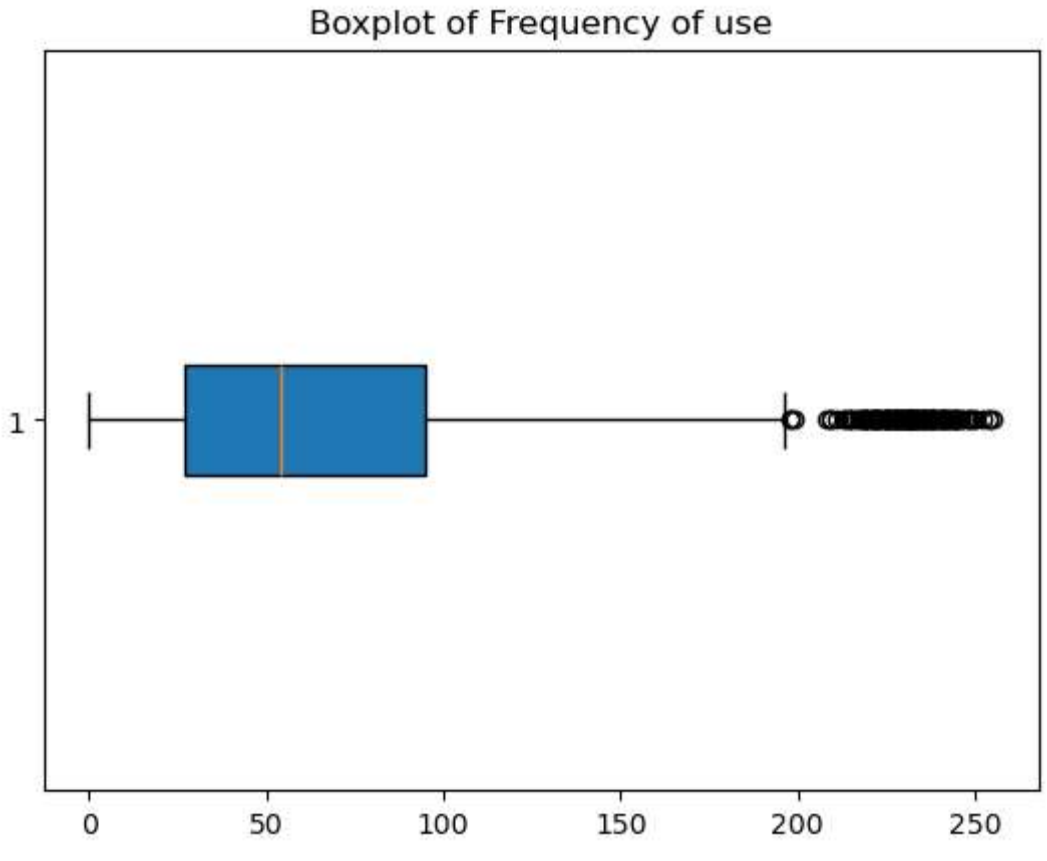
In [19]:
```
plt.boxplot(df_quant['Seconds of Use'], patch_artist = True, vert = False)
plt.title('Boxplot of Seconds of Use');
```



The Seconds of use variable has outliers at the higher spectrum. On the other hand, the right tail of its boxplot is longer than the left, while the median line is to the left. Thus the distribution is **right skewed.**

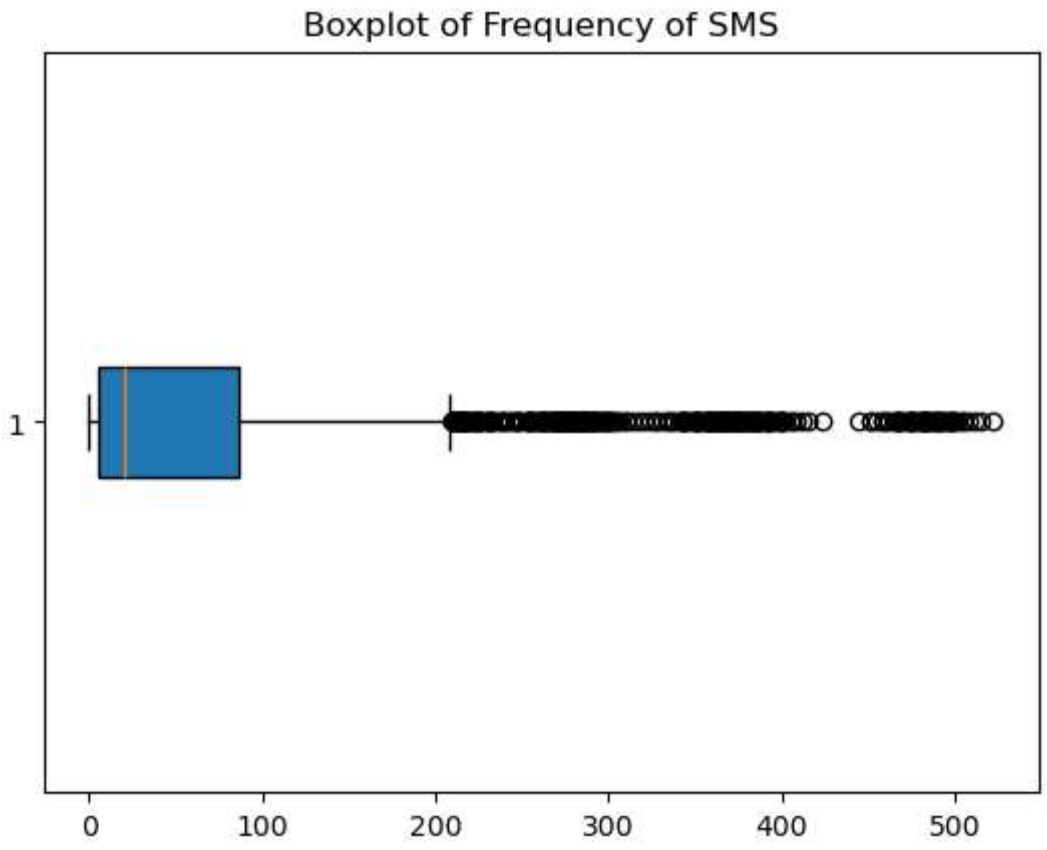
In [20]:
```
plt.boxplot(df_quant['Frequency of use'], patch_artist = True, vert = False)
```

```
plt.title('Boxplot of Frequency of use');
```



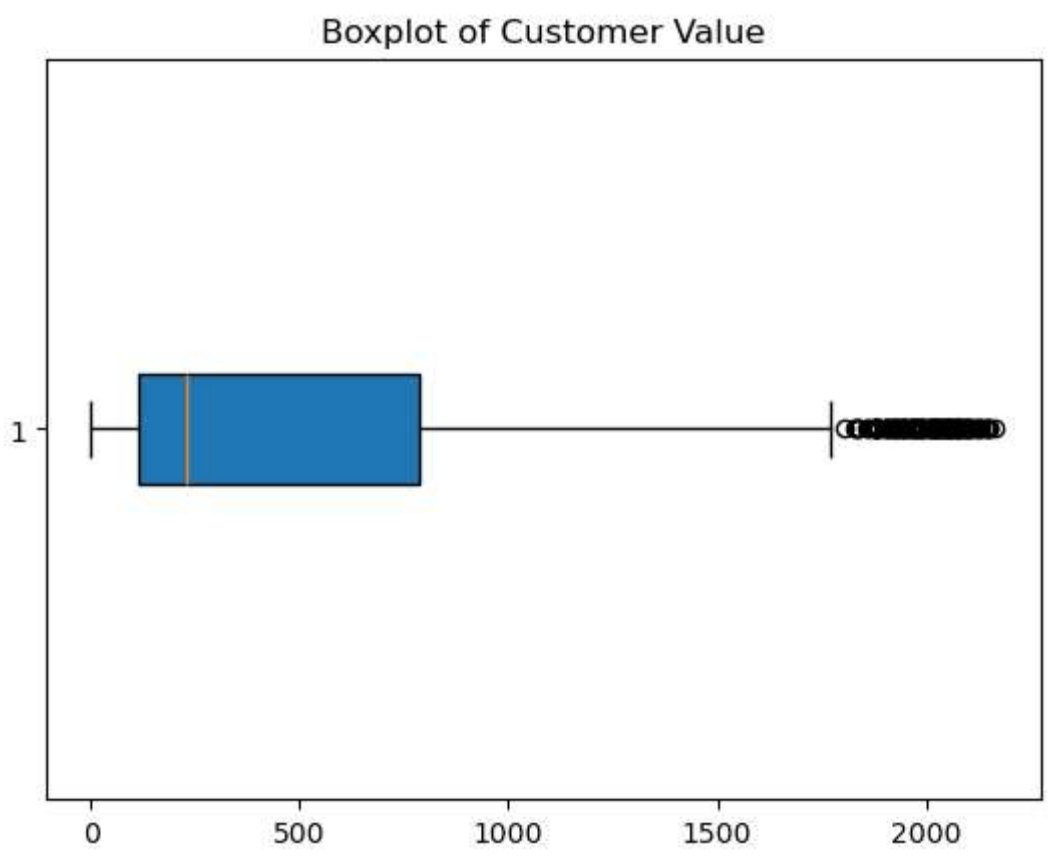
Similar to seconds of use, frequency of use is also **right skewed** considering the longer right tail and the median line that is closer to the left. Outliers exist at the upper extremes of the distribution.

In [21]:
```
plt.boxplot(df_quant['Frequency of SMS'], patch_artist = True, vert = False)
plt.title('Boxplot of Frequency of SMS');
```



frequency of SMS has noticeably more outliers at the upper edge of the distribution. The longer length of the right tail and the median line which is towards the right indicates that frequency of SMS is **right skewed.**

In [22]:
```
plt.boxplot(df_quant['Customer Value'], patch_artist = True, vert = False)
plt.title('Boxplot of Customer Value');
```



The right tail of customer value is much longer than the left tail and the median line aligns significantly to the left indicating a **right skewed** data distribution.
Some outliers are apparent in the customer value from the boxplot.

2.1.5 Outliers

From the previous visualizations (the boxplots), outliers exist in all five (5) of the fields. The outliers will be identified in this section and where appropriate, removed from the data. Outliers are removed so that they do not skew the model I intend to build. There are over 3000 records in the dataset, thus, removing the outliers where neccessary will not compromise the usability of the data for modeling as a result of volume.

In [23]:
```
def outlier_remover(df, series):
    # A function to remove outliers based on the inner fences
    # Returns a dataframe without outliers
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    cutoff = IQR * 1.5
    LIF = Q1 - cutoff
    UIF = Q3 + cutoff
    df = df[(series > LIF) & (series < UIF)]
    return df
```

In [24]:
```
df_quant = outlier_remover(df_quant, df_quant['Subscription Length'])
df_quant = outlier_remover(df_quant, df_quant['Seconds of Use'])
df_quant = outlier_remover(df_quant, df_quant['Frequency of use'])
df_quant = outlier_remover(df_quant, df_quant['Frequency of SMS'])
df_quant = outlier_remover(df_quant, df_quant['Customer Value'])
df_quant.shape
```

Out[24]: (1936, 5)

After removing all the outliers, we have more than 1,800 records left. This is sufficient for modelling.

2.1.6 Correlation

The correlation across the variables will be examined numerically, and then with visualization using a correlation matrix.

In [25]:
```
df_quant.corr()
```

Out[25]:

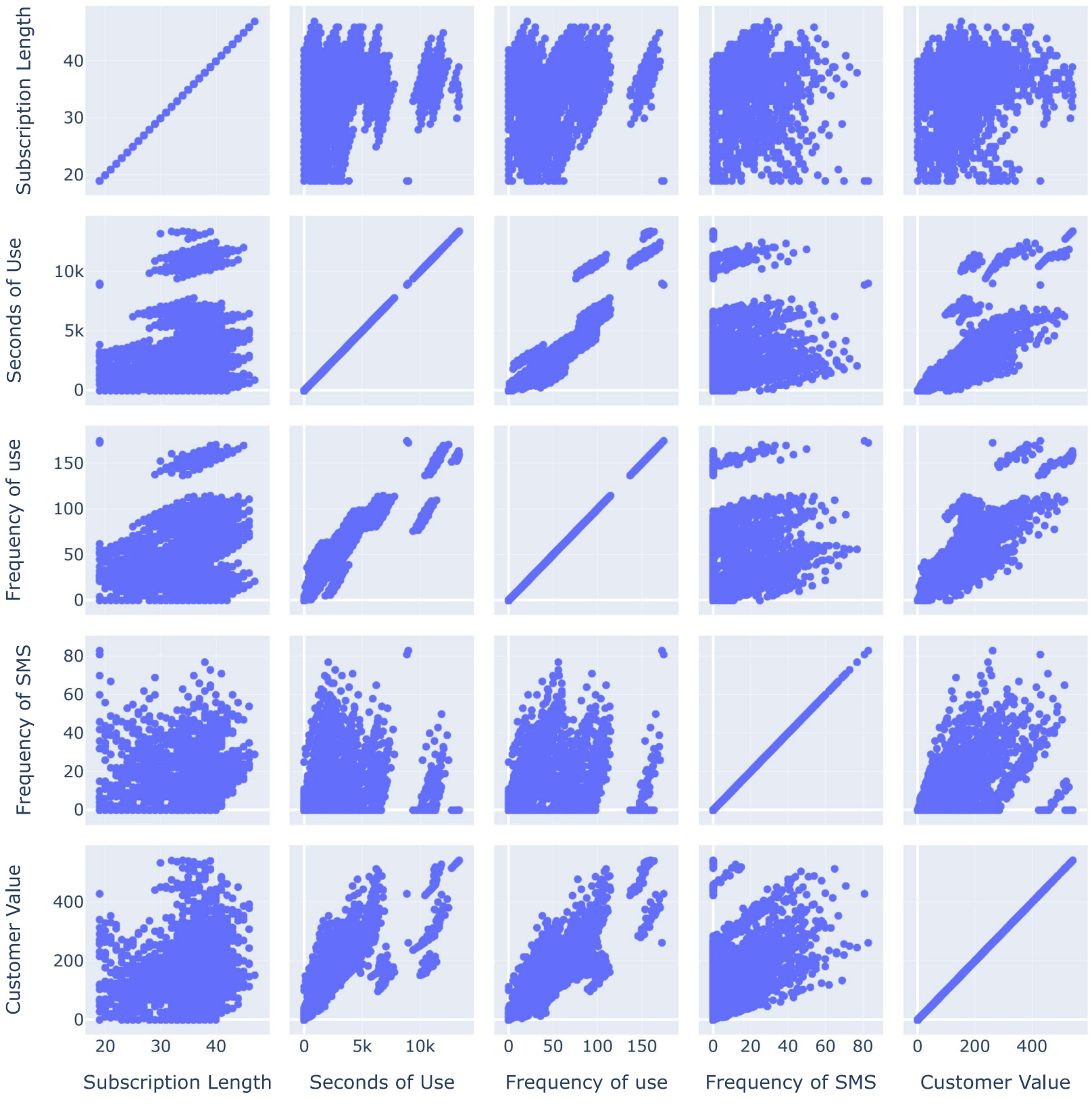
	Subscription Length	Seconds of Use	Frequency of use	Frequency of SMS	Customer Value
Subscription Length	1.000000	0.191050	0.225558	0.064501	0.219950
Seconds of Use	0.191050	1.000000	0.912530	0.027797	0.741500
Frequency of use	0.225558	0.912530	1.000000	0.174818	0.833685
Frequency of SMS	0.064501	0.027797	0.174818	1.000000	0.439215
Customer Value	0.219950	0.741500	0.833685	0.439215	1.000000

The great news from the correlation matrix above is that the Customer value which is the response variable is strongly correlated with frequency of use and seconds of use. This makes them suitable explanatory variables.
However, further examination reveals that frequency of use and seconds of use (the same variables) are strongly correlated with each other. This makes sense since the more frequently the services are used, the more time (seconds) will be spent on the service.

For this model, all 4 explanatory variables will still be used. However, for an optimized model, it is undesirable to use features that correlate. Further preprocessing will be required on all such features.

2.1.7 ScatterMatrix

In [26]:
```
# Resolve the error Datafram object has no attribute 'iteritems'
pd.DataFrame.iteritems = pd.DataFrame.items
# Plot scattermatrix for all 5 variablesm
px.scatter_matrix(df_quant, width = 900, height = 900)
```



The above visualization of the scatter matrix shows the visual relationship across the variables. As deduced from the correlation matrix, seconds of use and frequency of use are the variables who's scatter plot reflects a positive linear relationship with customer value and between each other.

2.1.8 Multiple Linear Regression

In [27]:
```
# Renaming the columns without spaces so that it can be used in statsmodel
df_quant.columns = ['Subscription_Length',
                    'Seconds_of_Use',
                    'Frequency_of_use',
                    'Frequency_of_SMS',
                    'Customer_Value']
# Scaling seconds of use to hours of use
df_quant['Hours_of_Use'] = df_quant.Seconds_of_Use/3600
df_quant.head()
```

Out[27]:

	Subscription_Length	Seconds_of_Use	Frequency_of_use	Frequency_of_SMS	Customer_Value	Hours_of_Use
0	38	4370	71	5	197.640	1.213889
1	39	318	5	7	46.035	0.088333
3	38	4198	66	1	240.020	1.166111
4	38	2393	58	2	145.805	0.664722
5	38	3775	82	32	282.280	1.048611

In [28]:
```
quant_model = sm.ols('Customer_Value ~ Subscription_Length + Hours_of_Use + Frequency_of_use + Frequency_of_SMS', df_quant).fit()
print(quant_model.summary2())
```

```
                 Results: Ordinary least squares
=================================================================
Model:               OLS             Adj. R-squared:     0.787
Dependent Variable:  Customer_Value  AIC:                20839.3190
Date:                2023-12-28 13:49 BIC:               20867.1609
No. Observations:    1936            Log-Likelihood:     -10415.
Df Model:            4               F-statistic:        1788.
Df Residuals:        1931            Prob (F-statistic): 0.00
R-squared:           0.787           Scale:              2761.8
-----------------------------------------------------------------
                     Coef.  Std.Err.    t    P>|t|   [0.025   0.975]
-----------------------------------------------------------------
Intercept           -5.1010   7.4031 -0.6890 0.4909 -19.6199  9.4178
Subscription_Length  0.5494   0.2168  2.5344 0.0113   0.1243  0.9746
Hours_of_Use        19.8040   3.8397  5.1577 0.0000  12.2736 27.3344
Frequency_of_use     1.9283   0.0831 23.2008 0.0000   1.7653  2.0914
Frequency_of_SMS     2.3817   0.0837 28.4674 0.0000   2.2176  2.5458
-----------------------------------------------------------------
Omnibus:               182.543       Durbin-Watson:          1.823
Prob(Omnibus):         0.000         Jarque-Bera (JB):       497.197
Skew:                  -0.513        Prob(JB):               0.000
Kurtosis:              5.261         Condition No.:          439
=================================================================
Notes:
[1] Standard Errors assume that the covariance matrix of the errors
is correctly specified.
```

Interpretation of Model Summary:

Intercept

The intercept coefficient is -0.0043. The standard error is 0.494 while its confidence interval is between -0.10 and 0.93 at 95% confidence level. The p-value of 0.93 which is greater than 0.05 indicates that we fail to reject the null hypothesis and conclude that the intercept has no effect on customer value. The test statistic is -0.087.

Subscription_Length

With a coefficient of -0.0013 and a p-value of 0.37, we likewise conclude that the effect of subscription length on customer value is zero. The 95% confidence interval is between -0.0042 and 0.0015 with a standard error of 0.0014 and test statistic of -0.90.

Hours of Use

Hours of use has the most significant effect on the response variable with a coefficient of 67.16 and a standard error of 1.36. The p-value of 0 indicates that we reject the null hypothesis and conclude that hours of use is related to customer value. The 95% confidence interval is from 64.49 to 69.84. The value of the test statistic is 49.24.

Frequency of Use

The cross correlation with seconds of use (now hours of use) may have been accountable for the coefficient of frequency of use to be as low as -0.0081 and a 95% confidence interval range from -0.0093 to -0.0070.The p-value of 0 albeit indicates that the relationship with customer value is still statistically worthy and we fail to reject the null hypothesis. The test statistic is -13.82.

Frequency of SMS

The coefficient is 0.0004 and the standard error is 0.0001. We reject the null hypothesis based on the p-value of 0.0018 which is less than 0.05 and conclude that frequency of sms relates to customer value. The test statistic is 3.12 and the 95% confidence interval ranges from 0.0002 to 0.0007.

Coefficient of Determination

The coefficient of determination of the model is 0.84. This indicates that more than 84% of the variability in the response variable is explained by the 4 features and the intercept.

2.1.9 Predictions

In [29]:
```
# Create random arrays with min and max values of each columns as lower and upper limits respectively
Psubscription_length = np.random.randint(df_quant['Subscription_Length'].min(), df_quant['Subscription_Length'].max(), 10)
PFrequency_of_Use = np.random.randint(df_quant['Frequency_of_use'].min(), df_quant['Frequency_of_use'].max(), 10)
PFrequency_of_SMS = np.random.randint(df_quant['Frequency_of_SMS'].min(), df_quant['Frequency_of_SMS'].max(), 10)
PHours_of_Use = np.random.uniform(df_quant['Hours_of_Use'].min(), df_quant['Hours_of_Use'].max(), 10)
# Create a new dataframe from the random values
new = pd.DataFrame({'Subscription_Length': Psubscription_length,
                    'Frequency_of_use': PFrequency_of_Use,
                    'Frequency_of_SMS': PFrequency_of_SMS,
                    'Hours_of_Use': PHours_of_Use})
new['Predicted_Customer_Value'] = quant_model.predict(new)
new
```

Out[29]:

	Subscription_Length	Frequency_of_use	Frequency_of_SMS	Hours_of_Use	Predicted_Customer_Value
0	24	30	63	1.277344	241.280125
1	30	96	23	0.781115	266.752077
2	37	18	64	3.579949	273.265176
3	29	49	10	1.197692	152.857929
4	42	146	68	0.482708	471.030010
5	23	8	28	1.315859	115.709957
6	32	23	40	1.372770	179.287702
7	34	138	73	0.209495	457.705543
8	19	138	35	3.583308	425.774130
9	37	19	36	1.692041	171.117500

2.2 Qualitative Analysis

From the main dataframe (df) of Iranian customer churn, the following columns will be extracted for qualitative analysis.

1. Churn
2. Subscription Length
3. Seconds of Use
4. Frequency of Use
5. Frequency of SMS

Churn is the field that will be used as the response variable when building the classification model.

In [30]:
```
qual = df[['Churn', 'Subscription  Length', 'Seconds of Use', 'Frequency of use', 'Frequency of SMS']]
qual.columns = ['Churn', 'Subscription_Length', 'Seconds_of_Use', 'Frequency_of_use', 'Frequency_of_SMS']
qual
```

Out[30]:

	Churn	Subscription_Length	Seconds_of_Use	Frequency_of_use	Frequency_of_SMS
0	0	38	4370	71	5
1	0	39	318	5	7
2	0	37	2453	60	359
3	0	38	4198	66	1
4	0	38	2393	58	2
...	...	...	...	...	...
3145	0	19	6697	147	92
3146	0	17	9237	177	80
3147	0	18	3157	51	38
3148	0	11	4695	46	222
3149	1	11	1792	25	7

3150 rows × 5 columns

2.2.1 Frequency table

The frequency tables and histogram for the explanatory variables have been constructed previously. The frequency table for the response variable (Churn) is constructed below.

In [31]:
```
freq = qual.Churn.value_counts()
freq = pd.DataFrame({'Churn':freq.keys(), 'frequency': freq.values})
freq['relative frequency'] = freq.frequency/(freq.frequency.sum())
freq
```

Out[31]:

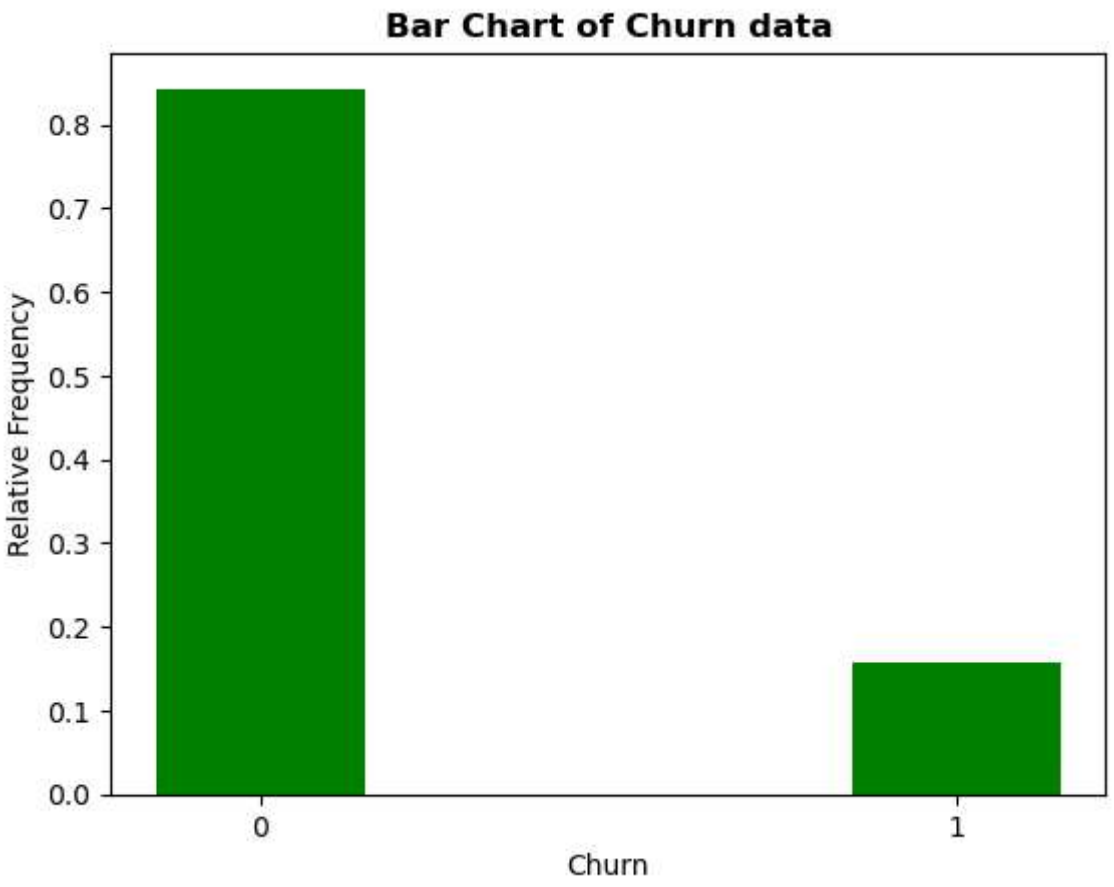
	Churn	frequency	relative frequency
0	0	2655	0.842857
1	1	495	0.157143

2.2.2 Bar Chart

In [32]:
```
# Convert the churn column to string so that the x axis is not marked in decimals from 0 to 1
churn = freq.Churn.astype(str)

plt.bar(churn, freq['relative frequency'], color = 'green', width = 0.3)
plt.xlabel('Churn')
```

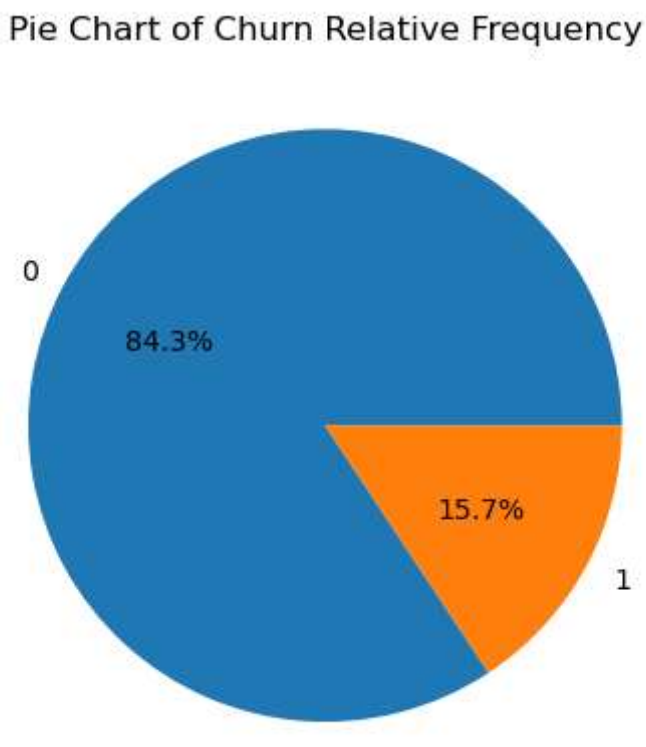
```
plt.ylabel("Relative Frequency")
plt.title('Bar Chart of Churn data', fontweight = 'bold');
```



As observable from both the frequency table and the histogram, the '0' (which represents non churn) is more than five times the '1' (the churn data). This may indicate some class imbalance and in practise, may require data resampling.

2.2.3 Pie Chart

In [35]:
```
plt.pie(freq['relative frequency'], labels = churn, autopct = '%1.1f%%')
plt.title('Pie Chart of Churn Relative Frequency');
```



The pie chart above basically shows similar information to the bar chart.

2.2.4 Train Test Split

In [36]:
```
# Preprocessing - outlier removal, seconds to hours conversion before model development
#Outlier remover using the function defined in section 2.1.5
qual = outlier_remover(qual, qual['Subscription_Length'])
qual = outlier_remover(qual, qual['Seconds_of_Use'])
qual = outlier_remover(qual, qual['Frequency_of_use'])
qual = outlier_remover(qual, qual['Frequency_of_SMS'])

# Convert seconds of use to hours of use
qual['Hours_of_Use'] = qual['Seconds_of_Use']/3600

qual.shape
```

Out[36]:
```
(2160, 6)
```

In [37]:
```
y = qual.Churn
x = qual.drop(columns = ['Churn'])
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 42)

qual_train = pd.concat([x_train, y_train], axis = 1, join = 'inner')
qual_test = pd.concat([x_test, y_test], axis = 1, join = 'inner')
print('train data shape:', qual_train.shape, '\n test data shape', qual_test.shape)
```

```
train data shape: (1512, 6)
 test data shape (648, 6)
```

2.2.5 Multiple Logistic Regression

In [38]:
```
qual.columns
```

Out[38]:
```
Index(['Churn', 'Subscription_Length', 'Seconds_of_Use', 'Frequency_of_use',
       'Frequency_of_SMS', 'Hours_of_Use'],
      dtype='object')
```

In [39]:
```
qual_model = sm.logit(formula = 'Churn ~ Subscription_Length + Seconds_of_Use + Frequency_of_use + Frequency_of_SMS', data = qual_train)
result = qual_model.fit()

result.summary()
```

```
Optimization terminated successfully.
        Current function value: 0.434331
        Iterations 7
```

Out[39]:

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	1512
Model:	Logit	**Df Residuals:**	1507
Method:	MLE	**Df Model:**	4
Date:	Thu, 28 Dec 2023	**Pseudo R-squ.:**	0.1513
Time:	13:50:28	**Log-Likelihood:**	-656.71
converged:	True	**LL-Null:**	-773.75
Covariance Type:	nonrobust	**LLR p-value:**	1.741e-49

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-2.4687	0.454	-5.440	0.000	-3.358	-1.579
Subscription_Length	0.0683	0.013	5.216	0.000	0.043	0.094
Seconds_of_Use	-0.0004	9e-05	-4.112	0.000	-0.001	-0.000
Frequency_of_use	-0.0043	0.005	-0.783	0.434	-0.015	0.006
Frequency_of_SMS	-0.0063	0.003	-1.935	0.053	-0.013	8.33e-05

Interpretation of Model Summary:

Intercept

The intercept coefficient is -2.47. The standard error is 0.45 while its confidence interval is between -3.36 and -1.58 at 95% confidence level. The p-value of 0 which is less than 0.05 indicates that we reject the null hypothesis and conclude that the intercept has an effect on churn. The test statistic is -5.44.

Subscription_Length

With a coefficient of 0.068 and a p-value of 0, we likewise conclude that the effect of subscription length on churn is significant. The 95% confidence interval is between 0.043 and 0.094 with a standard error of 0.013 and test statistic of 5.22.

Hours of Use

Hours of use also has a negative effect on the response variable with a coefficient of -1.33 and a standard error of 0.32. The p-value of 0 indicates that we reject the null hypothesis and conclude that hours of use is related to churn. The 95% confidence interval is from -1.97 to -0.698. The value of the test statistic is -4.11.

Frequency of Use

The coefficient of frequency of use is -0.0043 and a 95% confidence interval range from -0.015 to 0.006.The p-value of 0.43 indicates that the relationship with customer value is not statistically significant and we fail to reject the null hypothesis. The test statistic is -0.78.

Frequency of SMS

The coefficient is -0.0063 and the standard error is 0.003. We reject the null hypothesis based on the p-value of 0 which is greater than 0.05 and conclude that frequency of sms does influence the customer churn. The test statistic is -1.94 and the 95% confidence interval ranges from -0.013 to 8.33e-05.

2.2.6 Predictions

In [40]:
```
predictions = result.predict(qual_test)
predicted_labels = (predictions > 0.5).astype(int)
true_labels = qual_test['Churn']
pred = pd.DataFrame({'True Class': true_labels, 'Predicted Probability':predictions, 'predicted class': predicted_labels})
pred.head(10)
```

Out[40]:

	True Class	Predicted Probability	predicted class
3031	1	0.492148	0
973	0	0.013676	0
3025	0	0.258928	0
1627	1	0.196226	0
1154	0	0.248445	0
2607	0	0.012159	0
2442	0	0.155235	0
2510	0	0.067729	0
1351	0	0.413318	0
3143	0	0.089970	0

In [41]:
```
print('\n True Classes \n', true_labels.value_counts(),'\n')
print('\n Predicted Classes \n', predicted_labels.value_counts())
```

```
 True Classes
 Churn
0    536
1    112
Name: count, dtype: int64

 Predicted Classes
 0    619
1     29
Name: count, dtype: int64
```

The model predicted only 29 customer churn while there is an actual 112. The confusion matrix will be used to check the model's performance.

2.2.7 Model Performance

In [42]:
```
true_labels = qual_test['Churn']
conf_matrix = confusion_matrix(true_labels, predicted_labels)
print('Confusion Matrix: \n', conf_matrix)
test_acc = accuracy_score(true_labels, predicted_labels)
print('The accuracy for the Test set is ', test_acc * 100)
print('Test Error Rate is ', 100 - test_acc*100)
```

```
Confusion Matrix:
 [[517  19]
 [102  10]]
The accuracy for the Test set is  81.32716049382715
Test Error Rate is  18.67283950617285
```

The model accuracy is 81%. From the confusion matrix, the model predicted 517 true positives and 10 true negatives. The false positives of 102 and false negatives of 19 accounted for about 19% error rate in the model.

3.0 Conclusion

Riding on the knowledge gained from the applied statistics for data science class, two different models have been developed and used for prediction. A linear regression model was used to predict customer value, and a R-squared score of 84% was obtained. Furthermore, a logistic model was used to classify customers based on their chances of churn and an accuracy score of 81% was obtained.

4.0 References

1. Iranian Churn Dataset. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C5JW3Z.