In [1]:
```
#!pip install pymatch
```

In [2]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
#from plotnine import *
import matplotlib.pyplot as plt
import seaborn as sns
from pymatch.Matcher import Matcher
from sklearn.utils import shuffle
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold
import warnings
import seaborn as sns
from scipy import stats
from scipy.stats import ttest_ind
warnings.filterwarnings('ignore')


pd.set_option('display.max_columns', 300)
pd.set_option('display.max_rows', 300)
```

All the data sets were downloaded from IBM Cognos Analytics https://community.ibm.com/community/user/businessanalytics/blogs/steven-macko/2019/07/10/new-base-samples-for-ibm-cognos-analytics-1113 (https://community.ibm.com/community/user/businessanalytics/blogs/steven-macko/2019/07/10/new-base-samples-for-ibm-cognos-analytics-1113) .

In [3]:
```
#Demo - demographics data set with information about customer's gender,
 age, marital status and number of dependents.
demo = pd.read_excel('../00_raw_data/Telco_customer_churn_demographics.x
lsx')
#Loca - information about customer's location: Country, State, City, and
Zip code of the area.
loca = pd.read_excel('../00_raw_data/Telco_customer_churn_location.xlsx'
)
#Pop - number of residents in a particular zip code area.
pop = pd.read_excel('../00_raw_data/Telco_customer_churn_population.xls
x')
#Serv - all information related to a phone service: tenure (in months),
 phone service, internet type, monthly charge, etc.
serv = pd.read_excel('../00_raw_data/Telco_customer_churn_services.xlsx'
)
#Churn - information about customer satisfaction and churn (if churned,
 also includes a reason for churn)
churn = pd.read_excel('../00_raw_data/Telco_customer_churn_status.xlsx')
```

All of the data sets have a common unique ID: Customer ID which we will use to merge the datasets together.

In [4]:
```
#Merging and dropping columns that will not be used in the analysis.
df1 = pd.merge(demo, loca, how='inner', on='Customer ID',indicator=True)
df1 = df1.drop(['Count_x', 'Count_y', 'Country', 'State',
               'Under 30','Senior Citizen','Lat Long',
               'Latitude','Longitude','_merge'], axis=1)
df2 = pd.merge(df1, serv, how='inner', on='Customer ID',indicator=True)
df2 = df2.drop(['Count', 'Quarter','_merge'], axis=1)

df3 = pd.merge(df2, churn, how='inner', on='Customer ID',indicator=True)
df3 = df3.drop(['Count', 'Quarter','_merge','Churn Label'], axis=1)

df = pd.merge(df3, pop, how='inner', on='Zip Code',indicator=True)
df = df.drop(['ID','_merge','Churn Score'], axis=1)
```

Let's look more closely at our data now.

In [5]:
```
print ('Dimensions of the data set', df.shape)
df.head(5)
```

Dimensions of the data set (7043, 42)

Out[5]:

	Customer ID	Gender	Age	Married	Dependents	Number of Dependents	City	Zip Code	Referred a Friend	Numb Referra
0	8779-QRDMV	Male	78	No	No	0	Los Angeles	90022	No	
1	4737-AQCPU	Male	39	Yes	No	0	Los Angeles	90022	Yes	
2	5043-TRZWM	Female	32	No	No	0	Los Angeles	90022	No	
3	8165-CBKXO	Male	35	Yes	Yes	3	Los Angeles	90022	Yes	
4	9979-RGMZT	Female	20	No	No	0	Los Angeles	90022	No	

In [6]:
```
null=pd.DataFrame(df.isnull().sum()/df.shape[0])
null[null[0]>0]
```

Out[6]:

	0
Churn Category	0.73463
Churn Reason	0.73463

Two categories that have missing data is, as expected, churn category and churn reason. In this case, missing values indicate that a user has not churned.

In [7]:
```
df=df.fillna('No churn')
```

In [8]:
```
df.describe()
```

Out[8]:

	Age	Number of Dependents	Zip Code	Number of Referrals	Tenure in Months	Avg Monthly Long Distance Charges	Avg Mont Downl
count	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000
mean	46.509726	0.468692	93486.070567	1.951867	32.386767	22.958954	20.515
std	16.750352	0.962802	1856.767505	3.001199	24.542061	15.448113	20.418
min	19.000000	0.000000	90001.000000	0.000000	1.000000	0.000000	0.000
25%	32.000000	0.000000	92101.000000	0.000000	9.000000	9.210000	3.000
50%	46.000000	0.000000	93518.000000	0.000000	29.000000	22.890000	17.000
75%	60.000000	0.000000	95329.000000	3.000000	55.000000	36.395000	27.000
max	80.000000	9.000000	96150.000000	11.000000	72.000000	49.990000	85.000

Data set has some outliers for instance, for variables "Total refunds" or "Total Extra Data Charges", but in general the data looks reasonable without any obvious errors such as negative values or extremely large entries.

In [9]:
```
#Creating the clean list of cities in the set (all written in the lower
 case without word "City" in the name).
temp = np.sort(df['City'].unique()).tolist()
cnt = 0

for item in temp:
    if 'city' in item.lower():
        cnt += 1

temp_remove_city = [a.replace('city', '').replace('City', '').strip() fo
r a in temp]
temp_remove_city = [a.lower() for a in temp_remove_city]
```

Income is an essential variable for customer churn question. Our data set did not have a variable estimating customer's income. We, however, found another set with information about an average income in each city.

In [10]:
```
#read in mean income data
mean_income = pd.read_csv('../00_raw_data/ACSST5Y2018.S1902_data_with_ov
erlays_2020-04-21T172347.csv',
encoding = "ISO-8859-1", skiprows=(1,))
mean_income = mean_income[['City','S1902_C03_019E']]
mean_income.columns = ['City','mean_income']
mean_income.head(5)
```

Out[10]:

	City	mean_income
0	Acalanes Ridge	53484
1	Acampo	70161
2	Acton	45253
3	Adelanto	12445
4	Adin	28639

In [11]:
```
print (f'The dataset has some {len([a for a in mean_income.mean_income.t
olist() if not a.isnumeric()])} unusual entries in the "mean_income" col
umn such as N or - that we want to remove.')
```

```
The dataset has some 54 unusual entries in the "mean_income" column suc
h as N or - that we want to remove.
```

In [12]:
```
#delete city which has "N" & '-' for mean_income
def preproc_cityname(city_name):
    return city_name.replace('City', '').replace('city', '').replace('CD
P', '').lower().split('(')[0].strip()

temp2 = np.array(mean_income).tolist()
temp2 = [[preproc_cityname(a[0]), a[1]] for a in temp2]
temp2 = [[a[0], int(a[1])] for a in temp2 if a[1] not in ['N', '-']]
mean_income_df = pd.DataFrame(data=temp2, columns=["City", "Mean_Income"
])
```

Now we merge our main dataset with the income information.

In [13]:
```
df['City'] = df['City'].str.lower()
final = pd.merge(df,mean_income_df, how = 'inner', on = 'City')
```

Data Preprocessing

Missing values

There is around 73% of missing data out of total records in *Churn Category* and *Churn Reason*. As the main project objective is to see what causal factors would lead to customer churn or not, the two variables are left for later analysis of identification of potential leaving customers if have time.

Binary and multi-category variables

The often-occuring "yes/no" binary response across different variables are modified to 1 and 0. For multi-category variables, dummy variables are created for each subcategory with 0 or 1 input.

Normalization on numeric variables

As the numeric variables have different scales, normalization is performed on each numeric predictors by substracting the minimum value of that predictor and then divided by range value of that predictor. All numeric variables are adjusted to around the same scale this way.

In [14]:
```
# Creating treatment variable with/without offer
final['treated'] = 1
final['treated'][final['Offer'] == 'None'] = 0

#drop useless columns
final_drop = final.drop(['CLTV','City','Zip Code','Total Revenue', 'Cust
omer Status'],axis=1)


#replace all "Yes" to "1", "No" to "0"
final_drop = final_drop.replace('Yes',1)
final_drop = final_drop.replace('No',0)

#Categorical variables transform (Female='1', Male='0')
final_drop = final_drop.replace('Female',1)
final_drop = final_drop.replace('Male',0)
```

In [15]:
```
#continous: normalization
def normalization(data, normalization_list):

    for i in normalization_list:
        data[i] = (data[i] - data[i].min())/(data[i].max()-data[i].min
())
    return data
```

In [16]:
```
#Creating a normalized final data set.
normalization_list = ['Tenure in Months','Avg Monthly Long Distance Char
ges',
                      'Avg Monthly GB Download','Monthly Charge','Total
 Charges',
                      'Total Long Distance Charges','Population', 'Mean_
Income']
normalization(final_drop, normalization_list).head(3)
```

Out[16]:

	Customer ID	Gender	Age	Married	Dependents	Number of Dependents	Referred a Friend	Number of Referrals	Tenure in Months	Offe
0	8779-QRDMV	0	78	0	0	0	0	0	0.0	Nor
1	4737-AQCPU	0	39	1	0	0	1	5	1.0	Nor
2	5043-TRZWM	1	32	0	0	0	0	0	0.0	Nor

In [17]:
```
final_drop=final_drop.fillna('No churn')
```

EDA

In [18]:
```
dummies=final_drop[['Internet Type','Contract','Payment Method','Churn C
ategory', 'Offer']]
```

In [19]:
```
temp=pd.get_dummies(dummies)
final_drop=final_drop.assign(**temp)
```

In [20]:
```
#Check collinearity. Note: we display only the highly collinear values.
 To see the full correlation matrix, use df.corr() command.
corr = final_drop[['Tenure in Months', 'Monthly Charge', 'Total Charges'
, 'Total Long Distance Charges','Contract_Month-to-Month']].corr()
def color_negative_red(val):
    if val > 0.5 and val!=1:
        color = 'red'
    elif val<-0.5:
        color='red'
    else:
        color='black'
    return 'color: %s' % color
highlited = corr.style.applymap(color_negative_red)
highlited
```

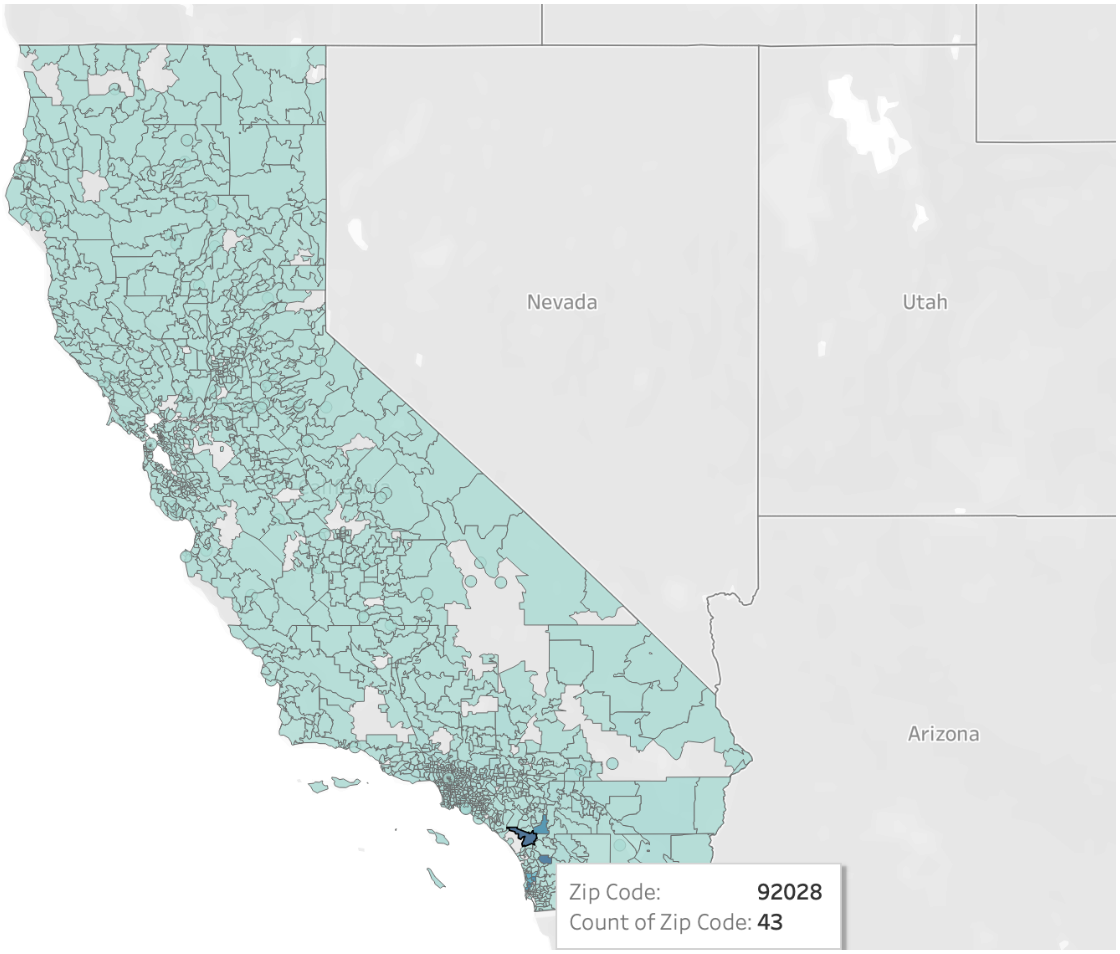
Out[20]:

	Tenure in Months	Monthly Charge	Total Charges	Total Long Distance Charges	Contract_Month-to-Month
Tenure in Months	1	0.245947	0.826428	0.670744	-0.627825
Monthly Charge	0.245947	1	0.648663	0.245187	0.0273106
Total Charges	0.826428	0.648663	1	0.608563	-0.444965
Total Long Distance Charges	0.670744	0.245187	0.608563	1	-0.416184
Contract_Month-to-Month	-0.627825	0.0273106	-0.444965	-0.416184	1

We also would like to know from what region data comes from. Note: for the sake of simplicity, we used Tableau for this visualization.

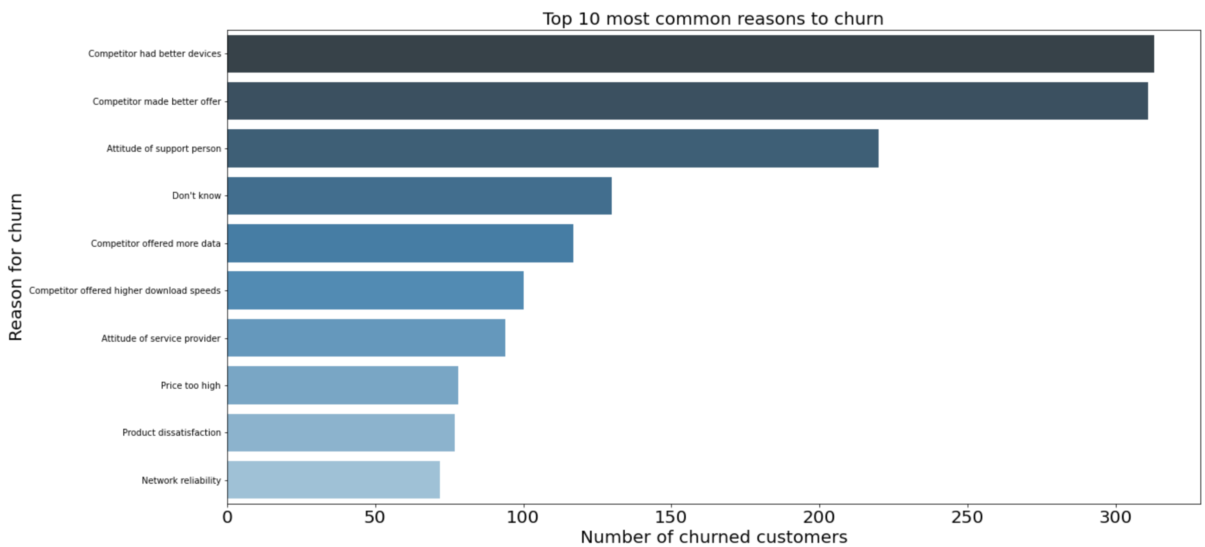
In [21]:
```
from IPython.display import Image
Image(filename = "california.png", width=700, height=600)
```

Out[21]:



It is clear that all data was collected in California on average with 4 respondents per zip code. Fallbrook, CA, however, has the largest value of 43 respondents. Let's start from investigating the top reasons for customers to churn.

In [22]:
```
plt.figure(figsize=(20,10))
reasons=pd.DataFrame(churn.groupby('Churn Reason')['Customer ID'].count
().nlargest(10))
y=reasons.index
x=reasons['Customer ID']
ax = sns.barplot(x=x, y=y,palette="Blues_d")
plt.xticks(fontsize=20)
plt.title('Top 10 most common reasons to churn',fontsize=20)
ax.set_xlabel('Number of churned customers', fontsize=20)
ax.set_ylabel('Reason for churn', fontsize=20)
plt.show()
```



The company has a variety of discount offers. Let's first check whether any of these offers are more effective than the others.

In [23]:
```
#final_drop['Offer']=serv['Offer'] #length of final_drop and serv is dif
ferent, cannot directly assign Offer or Customer ID
#final_drop['Customer ID']=serv['Customer ID']
offers=pd.DataFrame(final_drop.groupby('Offer')['Customer ID'].count())
no_churn=pd.DataFrame(final_drop.groupby('Offer')['Churn Category_No chu
rn'].sum())
offers=pd.DataFrame(pd.DataFrame(offers['Customer ID'] - no_churn['Churn
Category_No churn'])[0]/offers['Customer ID']*100)
```

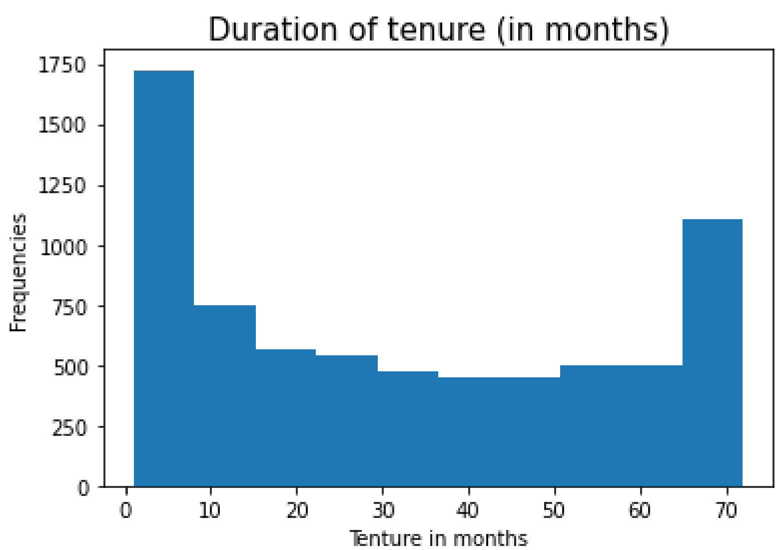
In [24]:
```
offers.columns=['% churned']
offers.sort_values(by='% churned', ascending=False)
```

Out[24]:

	% churned
Offer	
Offer E	52.269400
None	28.151515
Offer D	26.219512
Offer C	22.928177
Offer B	12.797619
Offer A	6.004619

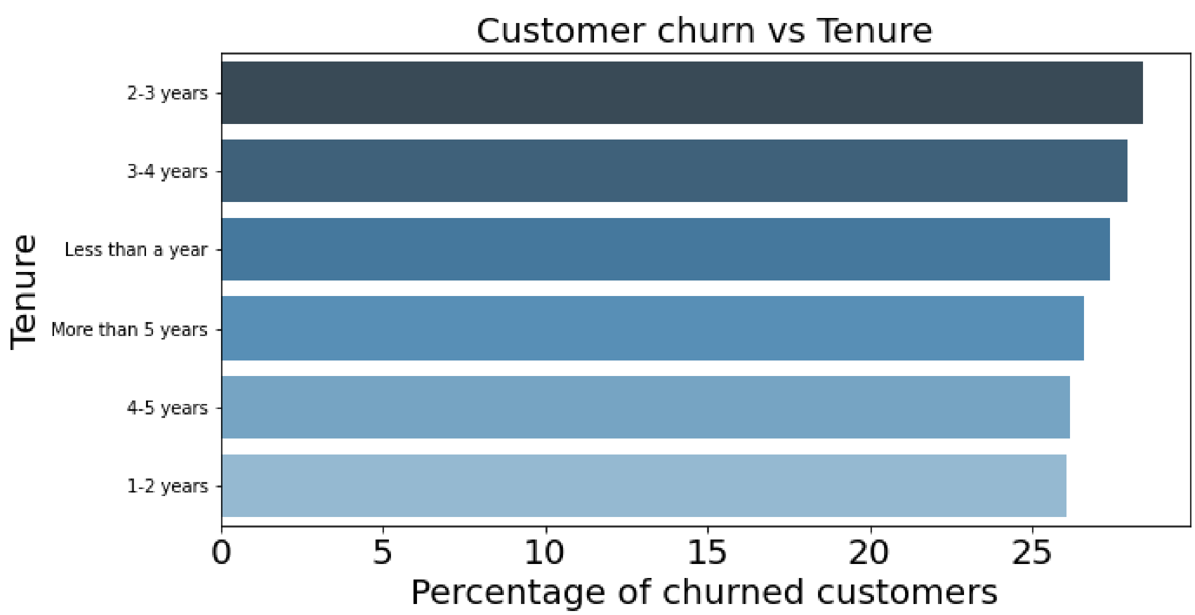
It seems like offer C is the least efficient and Offer D the most efficient as it has the lowest proportion of churned customers. Let's check now how long the customers utilize the service.

In [25]:
```
plt.hist(serv['Tenure in Months'])
plt.title('Duration of tenure (in months)',fontsize=15)
ax.set_xlabel('Months of tenure', fontsize=15)
plt.xlabel('Tenture in months')
plt.ylabel('Frequencies')
plt.show()
```



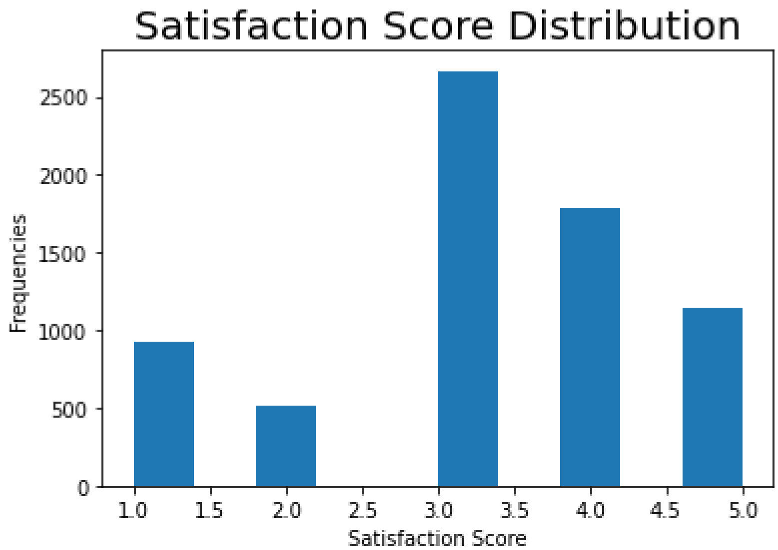
Let's compare the difference in churn rates that used service for a year; 1-2 years; 2-3 years; 4-5 years; or more than five years.

In [26]:
```
df=final_drop.copy()
df['Tenure']='Less than a year'
mask = (serv['Tenure in Months'] >=12) & (serv['Tenure in Months']<24)
df['Tenure'][mask] = '1-2 years'
mask = (serv['Tenure in Months'] >=24) & (serv['Tenure in Months']<36)
df['Tenure'][mask] = '2-3 years'
mask = (serv['Tenure in Months'] >=36) & (serv['Tenure in Months']<48)
df['Tenure'][mask] = '3-4 years'
mask = (serv['Tenure in Months'] >=48) & (serv['Tenure in Months']<60)
df['Tenure'][mask] = '4-5 years'
mask = (serv['Tenure in Months'] >=60)
df['Tenure'][mask] = 'More than 5 years'
churn_tenure=pd.DataFrame((pd.DataFrame(df.groupby(['Tenure'])['Customer ID'].count())['Customer ID']-pd.DataFrame(df.groupby(['Tenure'])['Churn Category_No churn'].sum())['Churn Category_No churn'])/pd.DataFrame(df.groupby(['Tenure'])['Customer ID'].count())['Customer ID']*100)
churn_tenure=churn_tenure.sort_values(by=[0], ascending=False)
plt.figure(figsize=(10,5))
y=churn_tenure.index
x=churn_tenure[0]
ax = sns.barplot(x=x, y=y,palette="Blues_d")
plt.xticks(fontsize=20)
plt.title('Customer churn vs Tenure',fontsize=20)
ax.set_xlabel('Percentage of churned customers', fontsize=20)
ax.set_ylabel('Tenure', fontsize=20)
plt.show()
```



Interestingly, the churn rate does not change significantly over time. Let's monitor customer satisfaction rates now.

In [27]:
```
plt.hist(churn['Satisfaction Score'])
plt.title ('Satisfaction Score Distribution',fontsize=20)
plt.xlabel('Satisfaction Score')
plt.ylabel('Frequencies')
plt.show()
```



In [28]:
```
treatment=final_drop[final_drop['treated']==1]
control=final_drop[final_drop['treated']==0]
trt_sat=np.mean(treatment['Satisfaction Score'])
cntl_sat=np.mean(control['Satisfaction Score'])
print (f'Average satistdaction score for churned customers: {trt_sat:.2
f} \nSatisfaction score for remained customers {cntl_sat:.2f}.')
```

```
Average satistdaction score for churned customers: 3.26
Satisfaction score for remained customers 3.21.
```

Is there any difference in premium features that remained and churned users have?

In [ ]:

In [29]:
```
#df['Churn Reason']=churn['Churn Reason'] #cannot do as len(churn) and l
en(df) does not fit

premium=df[['Customer ID','Online Security','Premium Tech Support','Stre
aming TV','Streaming Movies','Streaming Music','Unlimited Data','Churn V
alue','Churn Reason']]
premium_total=premium.drop(['Customer ID','Churn Reason','Churn Value'],
axis=1)
premium['total']=np.sum(premium_total, axis=1)
treat_premium=premium[premium['Churn Value']==1]
control_premium=premium[premium['Churn Value']==0]
premium_features=['Online Security','Premium Tech Support','Streaming T
V','Streaming Movies','Streaming Music','Unlimited Data']

for feature in premium_features:
    print (f'Feature: {feature}.\nAvg value for treatment group {np.mean
(treat_premium[feature])}.\nAvg value for control group {np.mean(control
_premium[feature])}')
    print (ttest_ind(treat_premium[feature],control_premium[feature]))
    print ('\n\n')
```

```
Feature: Online Security.
Avg value for treatment group 0.15527950310559005.
Avg value for control group 0.3331024930747922
Ttest_indResult(statistic=-13.706544317291769, pvalue=4.036523767226141
e-42)



Feature: Premium Tech Support.
Avg value for treatment group 0.16459627329192547.
Avg value for control group 0.3340258541089566
Ttest_indResult(statistic=-12.995876844123266, pvalue=4.253748771622332
e-38)



Feature: Streaming TV.
Avg value for treatment group 0.44161490683229815.
Avg value for control group 0.36334256694367495
Ttest_indResult(statistic=5.525433469679423, pvalue=3.426432696626884e-
08)



Feature: Streaming Movies.
Avg value for treatment group 0.4403726708074534.
Avg value for control group 0.37119113573407203
Ttest_indResult(statistic=4.868422762457452, pvalue=1.154025694833617e-
06)



Feature: Streaming Music.
Avg value for treatment group 0.3869565217391304.
Avg value for control group 0.34233610341643583
Ttest_indResult(statistic=3.1980784265583146, pvalue=0.0013907277723094
34)



Feature: Unlimited Data.
Avg value for treatment group 0.801863354037267.
Avg value for control group 0.6281163434903048
Ttest_indResult(statistic=12.885310757977328, pvalue=1.724584242797852e
-37)
```

In [30]:

```
final_drop = final_drop.drop(['Customer ID','Churn Reason','Churn Catego
ry'],axis=1)
```

The difference is significant for all categories. However, the proportion is higher for churned users across all premium features except online security and tech support - we should pay additional attention to these features when do the analysis.

Logistic regression: detect impactful features

In [31]:
```
final_drop2 = final_drop.copy()
final_drop2 = final_drop2.drop(['Offer','Internet Type','Contract',
                                'Payment Method', 'Churn Category_Attitude',
                                'Churn Category_Competitor','Churn Category_Dissatisfaction',
                                'Churn Category_No churn','Churn Category_Other',
                                'Churn Category_Price', 'Offer_None','Offer_Offer A',
                                'Offer_Offer B','Offer_Offer C','Offer_Offer D','Offer_Offer E'],axis=1)
```

In [32]:
```
X = final_drop2.drop('Churn Value',axis=1)
X['intercept'] =1
y = final_drop2['Churn Value']

logit_model=sm.Logit(y,X)
result=logit_model.fit()
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.089389
         Iterations: 35
```

In [33]:
```
print(result.summary2())
```

```
                                           Results: Logit
========================================================================
============================
Model:                            Logit                           Pseudo R-s
quared:             0.847
Dependent Variable:               Churn Value                     AIC:
1138.3015
Date:                             2020-04-28 22:31                BIC:
1392.5139
No. Observations:                 5942                            Log-Likeli
hood:               -531.15
Df Model:                         37                              LL-Null:
-3471.3
Df Residuals:                     5904                            LLR p-valu
e:                  0.0000
Converged:                        0.0000                          Scale:
1.0000
No. Iterations:                   35.0000
------------------------------------------------------------------------
----------------------------
                                    Coef.    Std.Err.      z    P>|z|
[0.025         0.975]
------------------------------------------------------------------------
----------------------------
Gender                               0.1365      0.1562  0.8736 0.3823
-0.1697        0.4427
Age                                  0.0183      0.0061  2.9932 0.0028
0.0063        0.0303
Married                             -0.0170      0.4487 -0.0379 0.9698
-0.8964        0.8624
Dependents                          -2.9971      0.6012 -4.9853 0.0000
-4.1754       -1.8188
Number of Dependents                 0.4885      0.2222  2.1989 0.0279
0.0531        0.9240
Referred a Friend                    2.3714      0.5170  4.5865 0.0000
1.3580        3.3847
Number of Referrals                 -0.9167      0.1346 -6.8089 0.0000
-1.1805       -0.6528
Tenure in Months                    -3.2356      0.9266 -3.4919 0.0005
-5.0517       -1.4195
Phone Service                       -2.0422      0.9443 -2.1626 0.0306
-3.8930       -0.1914
Avg Monthly Long Distance Charges   -0.3812      0.4007 -0.9513 0.3414
-1.1665        0.4041
Multiple Lines                      -0.0633      0.2936 -0.2156 0.8293
-0.6387        0.5121
Internet Service                    10.4983         nan     nan    nan
nan           nan
Avg Monthly GB Download              0.4074      0.4687  0.8692 0.3847
-0.5112        1.3261
Online Security                     -3.7533      0.4295 -8.7392 0.0000
-4.5951       -2.9116
Online Backup                       -0.7420      0.2813 -2.6374 0.0084
-1.2934       -0.1906
Device Protection Plan              -0.0855      0.2966 -0.2885 0.7730
-0.6668        0.4957
Premium Tech Support                -1.0777      0.2940 -3.6653 0.0002
```

```
-1.6540       -0.5014
Streaming TV                         -0.7064       0.4627 -1.5265 0.1269
-1.6133        0.2006
Streaming Movies                     -0.9725       0.5295 -1.8368 0.0662
-2.0102        0.0652
Streaming Music                       0.5247       0.3078  1.7047 0.0883
-0.0786        1.1281
Unlimited Data                        0.0389       0.3635  0.1071 0.9147
-0.6736        0.7514
Paperless Billing                     0.3471       0.1826  1.9010 0.0573
-0.0108        0.7050
Monthly Charge                        8.8433       4.2630  2.0744 0.0380
0.4878       17.1987
Total Charges                         1.1260       1.2980  0.8675 0.3857
-1.4180        3.6700
Total Refunds                        -0.0207       0.0109 -1.8908 0.0586
-0.0421        0.0008
Total Extra Data Charges              0.0022       0.0049  0.4416 0.6588
-0.0074        0.0117
Total Long Distance Charges           1.5183       0.9359  1.6222 0.1048
-0.3161        3.3527
Satisfaction Score                  -20.9776     341.8662 -0.0614 0.9511
-691.0229      649.0678
Population                            1.0746       0.4032  2.6650 0.0077
0.2843        1.8648
Mean_Income                          -0.2503       0.7047 -0.3552 0.7225
-1.6316        1.1310
treated                               0.1051       0.1575  0.6675 0.5044
-0.2035        0.4137
Internet Type_Cable                   4.4433 3759870.4160  0.0000 1.0000
-7369206.1585  7369215.0452
Internet Type_DSL                     4.2129 3758584.7146  0.0000 1.0000
-7366686.4607  7366694.8864
Internet Type_Fiber Optic             2.4486 3542497.3804  0.0000 1.0000
-6943164.8324  6943169.7296
Internet Type_None                   16.6523 2658754.1275  0.0000 1.0000
-5211045.6813  5211078.9858
Contract_Month-to-Month               9.9041 6127262.2770  0.0000 1.0000
-12009203.4827 12009223.2908
Contract_One Year                     9.1247 6060295.6512  0.0000 1.0000
-11877952.0874 11877970.3367
Contract_Two Year                     8.2732 6060295.6512  0.0000 1.0000
-11877952.9389 11877969.4852
Payment Method_Bank Withdrawal        9.2318 9400661.0972  0.0000 1.0000
-18424947.9496 18424966.4132
Payment Method_Credit Card            8.7899 9400661.0972  0.0000 1.0000
-18424948.3915 18424965.9713
Payment Method_Mailed Check           9.2804 9400661.0972  0.0000 1.0000
-18424947.9010 18424966.4618
intercept                            27.3021          nan     nan    nan
nan           nan
=======================================================================
============================
```

Propensity score matching

In [34]:
```
#change variables' names: space to underscore
propensity = final_drop2.copy()
cols=[]
for col in propensity.columns:
    if col=='Contract_Month-to-Month':
        col='Contract_Month_to_Month'
        cols.append(col)
    else:
        cols.append(col.replace(" ", "_"))

propensity.columns=cols

#Treatment = customers with offer; control = customers without offer.
control = propensity[propensity['treated'] == 0]
treatment = propensity[propensity['treated'] == 1]
```

In [35]:
```
propensity.columns
```

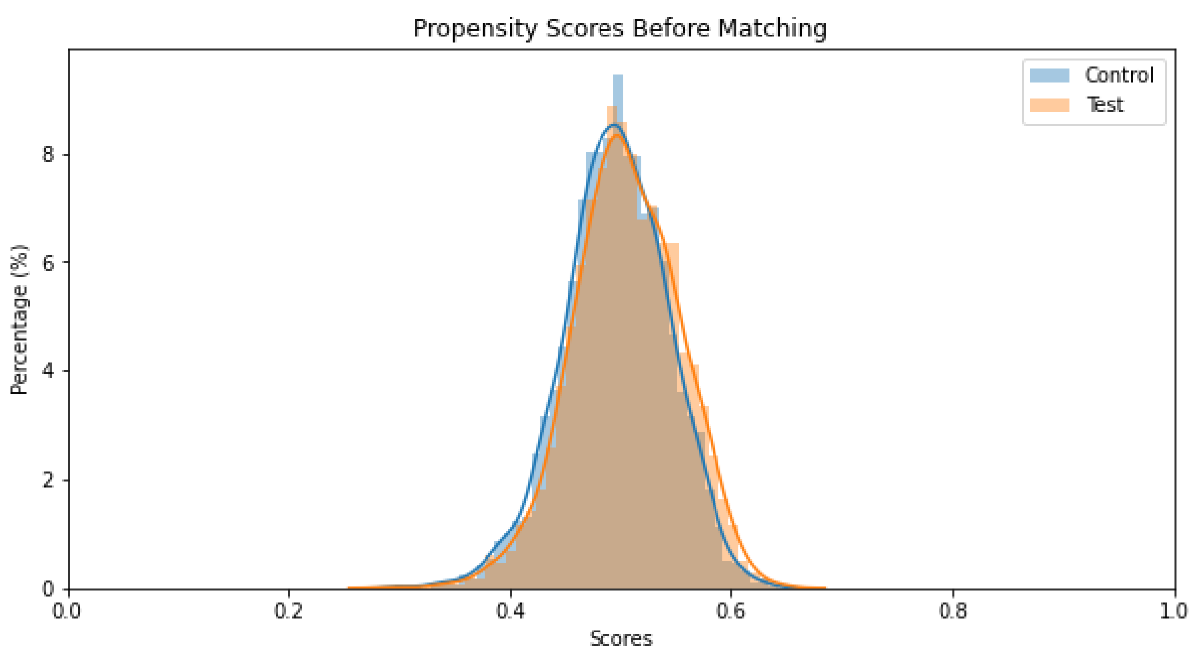
Out[35]:
```
Index(['Gender', 'Age', 'Married', 'Dependents', 'Number_of_Dependent
s',
       'Referred_a_Friend', 'Number_of_Referrals', 'Tenure_in_Months',
       'Phone_Service', 'Avg_Monthly_Long_Distance_Charges', 'Multiple_
Lines',
       'Internet_Service', 'Avg_Monthly_GB_Download', 'Online_Securit
y',
       'Online_Backup', 'Device_Protection_Plan', 'Premium_Tech_Suppor
t',
       'Streaming_TV', 'Streaming_Movies', 'Streaming_Music', 'Unlimite
d_Data',
       'Paperless_Billing', 'Monthly_Charge', 'Total_Charges', 'Total_R
efunds',
       'Total_Extra_Data_Charges', 'Total_Long_Distance_Charges',
       'Satisfaction_Score', 'Churn_Value', 'Population', 'Mean_Incom
e',
       'treated', 'Internet_Type_Cable', 'Internet_Type_DSL',
       'Internet_Type_Fiber_Optic', 'Internet_Type_None',
       'Contract_Month_to_Month', 'Contract_One_Year', 'Contract_Two_Ye
ar',
       'Payment_Method_Bank_Withdrawal', 'Payment_Method_Credit_Card',
       'Payment_Method_Mailed_Check'],
      dtype='object')
```

In [36]:
```
# Propensity score matching
from pymatch.Matcher import Matcher
m = Matcher(control, treatment, yvar = 'treated',
            exclude = ['Churn_Value'])


np.random.seed(20)
m.fit_scores(balance=True, nmodels=200)
```

```
Formula:
treated ~ Gender+Age+Married+Dependents+Number_of_Dependents+Referred_a
_Friend+Number_of_Referrals+Tenure_in_Months+Phone_Service+Avg_Monthly_
Long_Distance_Charges+Multiple_Lines+Internet_Service+Avg_Monthly_GB_Do
wnload+Online_Security+Online_Backup+Device_Protection_Plan+Premium_Tec
h_Support+Streaming_TV+Streaming_Movies+Streaming_Music+Unlimited_Data+
Paperless_Billing+Monthly_Charge+Total_Charges+Total_Refunds+Total_Extr
a_Data_Charges+Total_Long_Distance_Charges+Satisfaction_Score+Populatio
n+Mean_Income+Internet_Type_Cable+Internet_Type_DSL+Internet_Type_Fiber
_Optic+Internet_Type_None+Contract_Month_to_Month+Contract_One_Year+Con
tract_Two_Year+Payment_Method_Bank_Withdrawal+Payment_Method_Credit_Car
d+Payment_Method_Mailed_Check
n majority: 3300
n minority: 2642
Fitting Models on Balanced Samples: 200\200
Average Accuracy: 53.68%
```

In [37]:
```
#Evaluate the common support of the treated and control groups
m.predict_scores()
m.plot_scores()
```



In [38]:
```
#Obtain a matched sample using k:1 nearest neighbor method
m.match(method = "min", nmatches = 1)
records = m.record_frequency()
records
```

Out[38]:

	freq	n_records
0	1	3588
1	2	403
2	3	155
3	4	66
4	5	17
5	6	9
6	7	2
7	8	1

In [39]:
```
m.assign_weight_vector()
matched = m.matched_data.sort_values("match_id")
matched.head(10)
```

Out[39]:

	record_id	weight	Gender	Age	Married	Dependents	Number_of_Dependents	Referred_
2642	3300	1.000000	1	20	0	0	0	
1263	1612	1.000000	0	42	0	0	0	
2643	3301	1.000000	1	74	1	1	1	
1282	1628	1.000000	1	31	0	0	0	
358	444	0.500000	1	74	0	0	0	
2644	3302	1.000000	0	71	0	1	3	
2645	3303	1.000000	0	34	0	0	0	
2316	2906	1.000000	1	31	0	0	0	
382	476	0.333333	0	41	0	0	0	
2646	3304	1.000000	1	65	0	0	0	

In [40]:
```
# save matched dataset csv
matched.to_csv('../20_analysis_datasets/matched.csv')
matched.describe().to_csv('../20_analysis_datasets/matched_summary.csv')
```

In [41]:
```
#Conduct a t-test between the treatment and control group using the matc
hed data.
from scipy.stats import ttest_ind

matched_market_offer = matched[matched['treated'] == 1]
matched_no_market_offer = matched[matched['treated'] == 0]
```

In [42]:
```
covar_list = matched.drop(['record_id','weight','treated','scores','matc
h_id'],axis=1).columns.tolist()
for covar in covar_list:
    ttest_p = ttest_ind(matched_market_offer[covar],matched_no_market_of
fer[covar])[1]
    print(f'P-value from t-test for {covar} between treatment and contro
l group is {ttest_p:.5f}.')
```

```
P-value from t-test for Gender between treatment and control group is
0.93420.
P-value from t-test for Age between treatment and control group is 0.51
312.
P-value from t-test for Married between treatment and control group is
0.63984.
P-value from t-test for Dependents between treatment and control group
is 0.20367.
P-value from t-test for Number_of_Dependents between treatment and cont
rol group is 0.28314.
P-value from t-test for Referred_a_Friend between treatment and control
group is 0.42293.
P-value from t-test for Number_of_Referrals between treatment and contr
ol group is 0.38048.
P-value from t-test for Tenure_in_Months between treatment and control
group is 0.72572.
P-value from t-test for Phone_Service between treatment and control gro
up is 0.96320.
P-value from t-test for Avg_Monthly_Long_Distance_Charges between treat
ment and control group is 0.81316.
P-value from t-test for Multiple_Lines between treatment and control gr
oup is 0.61817.
P-value from t-test for Internet_Service between treatment and control
group is 0.43696.
P-value from t-test for Avg_Monthly_GB_Download between treatment and c
ontrol group is 0.59913.
P-value from t-test for Online_Security between treatment and control g
roup is 0.51180.
P-value from t-test for Online_Backup between treatment and control gro
up is 0.66656.
P-value from t-test for Device_Protection_Plan between treatment and co
ntrol group is 0.88588.
P-value from t-test for Premium_Tech_Support between treatment and cont
rol group is 0.27364.
P-value from t-test for Streaming_TV between treatment and control grou
p is 0.65108.
P-value from t-test for Streaming_Movies between treatment and control
group is 0.06399.
P-value from t-test for Streaming_Music between treatment and control g
roup is 0.05330.
P-value from t-test for Unlimited_Data between treatment and control gr
oup is 0.57487.
P-value from t-test for Paperless_Billing between treatment and control
group is 0.91053.
P-value from t-test for Monthly_Charge between treatment and control gr
oup is 0.24226.
P-value from t-test for Total_Charges between treatment and control gro
up is 0.53653.
P-value from t-test for Total_Refunds between treatment and control gro
up is 0.38223.
P-value from t-test for Total_Extra_Data_Charges between treatment and
control group is 0.46237.
P-value from t-test for Total_Long_Distance_Charges between treatment a
nd control group is 0.95593.
P-value from t-test for Satisfaction_Score between treatment and contro
l group is 0.04436.
P-value from t-test for Churn_Value between treatment and control group
```

```
is 0.22427.
P-value from t-test for Population between treatment and control group
is 0.85955.
P-value from t-test for Mean_Income between treatment and control group
is 0.86583.
P-value from t-test for Internet_Type_Cable between treatment and contr
ol group is 0.04097.
P-value from t-test for Internet_Type_DSL between treatment and control
group is 0.02906.
P-value from t-test for Internet_Type_Fiber_Optic between treatment and
control group is 0.27958.
P-value from t-test for Internet_Type_None between treatment and contro
l group is 0.43696.
P-value from t-test for Contract_Month_to_Month between treatment and c
ontrol group is 0.47430.
P-value from t-test for Contract_One_Year between treatment and control
group is 0.89453.
P-value from t-test for Contract_Two_Year between treatment and control
group is 0.49273.
P-value from t-test for Payment_Method_Bank_Withdrawal between treatmen
t and control group is 0.89000.
P-value from t-test for Payment_Method_Credit_Card between treatment an
d control group is 0.65126.
P-value from t-test for Payment_Method_Mailed_Check between treatment a
nd control group is 0.52888.
```

Great! There is no significant difference between treatment and control groups.

At first, we fit a logistic regression using all variables; then select only statistically significant variables with the significance level<0.05. After that we used propensity score matching to ensure no baseline difference between treatment and control groups. Now we fit a weighted linear regression to evaluate the effect of all features on customers' churn.

Weighted Linear regression

In [43]:
```
matched['Churn_Value']=churn['Churn Value']
```

In [44]:
```
print(final_drop2.columns)
```

```
Index(['Gender', 'Age', 'Married', 'Dependents', 'Number of Dependent
s',
       'Referred a Friend', 'Number of Referrals', 'Tenure in Months',
       'Phone Service', 'Avg Monthly Long Distance Charges', 'Multiple
Lines',
       'Internet Service', 'Avg Monthly GB Download', 'Online Securit
y',
       'Online Backup', 'Device Protection Plan', 'Premium Tech Suppor
t',
       'Streaming TV', 'Streaming Movies', 'Streaming Music', 'Unlimite
d Data',
       'Paperless Billing', 'Monthly Charge', 'Total Charges', 'Total R
efunds',
       'Total Extra Data Charges', 'Total Long Distance Charges',
       'Satisfaction Score', 'Churn Value', 'Population', 'Mean_Incom
e',
       'treated', 'Internet Type_Cable', 'Internet Type_DSL',
       'Internet Type_Fiber Optic', 'Internet Type_None',
       'Contract_Month-to-Month', 'Contract_One Year', 'Contract_Two Ye
ar',
       'Payment Method_Bank Withdrawal', 'Payment Method_Credit Card',
       'Payment Method_Mailed Check'],
      dtype='object')
```

In [45]:
```
# drop high correlated columns
final_drop2 = final_drop2.drop('Total Charges',axis=1)
```

In [46]:
```
new_name = []
for name in final_drop2.columns:
    new_name.append(name.replace(' ','_'))

#change variables' names: space to underscore
ori_name = final_drop2.columns.tolist()

final_drop2.columns = new_name
```

Covariates taking into modeling (removed correlated and unsignificant items)

In [47]:
```
ans=''
for name in final_drop2.columns:
    ans += '+'+name
print(ans)
```

```
+Gender+Age+Married+Dependents+Number_of_Dependents+Referred_a_Friend+N
umber_of_Referrals+Tenure_in_Months+Phone_Service+Avg_Monthly_Long_Dist
ance_Charges+Multiple_Lines+Internet_Service+Avg_Monthly_GB_Download+On
line_Security+Online_Backup+Device_Protection_Plan+Premium_Tech_Support
+Streaming_TV+Streaming_Movies+Streaming_Music+Unlimited_Data+Paperless
_Billing+Monthly_Charge+Total_Refunds+Total_Extra_Data_Charges+Total_Lo
ng_Distance_Charges+Satisfaction_Score+Churn_Value+Population+Mean_Inco
me+treated+Internet_Type_Cable+Internet_Type_DSL+Internet_Type_Fiber_Op
tic+Internet_Type_None+Contract_Month-to-Month+Contract_One_Year+Contra
ct_Two_Year+Payment_Method_Bank_Withdrawal+Payment_Method_Credit_Card+P
ayment_Method_Mailed_Check
```

In [54]:
```
weight = matched['weight'].values
model=smf.wls('Churn_Value~Gender+Age+Married+Dependents+Number_of_Depen
dents+Referred_a_Friend+Number_of_Referrals+Tenure_in_Months+Phone_Servi
ce+Avg_Monthly_Long_Distance_Charges+Multiple_Lines+Internet_Service+Avg
_Monthly_GB_Download+Online_Security+Online_Backup+Device_Protection_Pla
n+Premium_Tech_Support+Streaming_TV+Streaming_Movies+Streaming_Music+Unl
imited_Data+Paperless_Billing+Monthly_Charge+Total_Refunds+Total_Extra_D
ata_Charges+Total_Long_Distance_Charges+Satisfaction_Score+Churn_Value+P
opulation+Mean_Income+treated+Internet_Type_Cable+Internet_Type_DSL+Inte
rnet_Type_Fiber_Optic+Internet_Type_None+Contract_One_Year+Contract_Two_
Year+Payment_Method_Bank_Withdrawal+Payment_Method_Credit_Card+Payment_M
ethod_Mailed_Check',
        matched, weights = weight).fit()
```

In [55]:
```
final=pd.DataFrame(np.exp(pd.read_html(model.summary().tables[1].as_html
(),header=0,index_col=0)[0]['coef'])).head(22)
#To exclude intercept coefficient
final=final[1:]
```

In [56]:
```
final.sort_values(by='coef').head(3)
```

Out[56]:

	coef
Gender	1.0
Number_of_Referrals	1.0
Avg_Monthly_Long_Distance_Charges	1.0

In [57]:
```
final.sort_values(by='coef').tail(3)
```

Out[57]:

	coef
Referred_a_Friend	1.0
Phone_Service	1.0
Unlimited_Data	1.0

Offering customers discounted options has a great effect on the chances of a customer to churn. Customers who received a discount offer are 51% less likely to churn compare to all others. Offer D is one of the most effective solutions. We have not obtained any evidence that any other variables impact churn rates.