In [ ]:

```
#Lut Lat Aung,  6511163,  542
```

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import uniform
from statsmodels.formula.api import ols

from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
```

In [72]:

```
# Q2 (2.1)


telecom_churn = pd.read_csv("telecom_churn.csv")
telecom_churn.head()

telecom_churn.isna().sum()

#There are no missing values
```

Out[72]:

```
Unnamed: 0                0
account_length            0
area_code                 0
international_plan        0
voice_mail_plan           0
number_vmail_messages     0
total_day_minutes         0
total_day_calls           0
total_day_charge          0
total_eve_minutes         0
total_eve_calls           0
total_eve_charge          0
total_night_minutes       0
total_night_calls         0
total_night_charge        0
total_intl_minutes        0
total_intl_calls          0
total_intl_charge         0
customer_service_calls    0
churn                     0
dtype: int64
```

In [77]:

```
# Q2 (2.2)

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split




X = telecom_churn.drop("churn", axis = 1)
y = telecom_churn["customer_service_calls"]
#[["total_day_charge"], ["total_eve_charge"], ["total_night_charge"], ["custom


knn = KNeighborsClassifier(n_neighbors = 6)
X_train, X_test, y_train, y_test =train_test_split(X,y, test_size=0.3,random_s

knn.fit(X_train, y_train)
print(knn.score(X_test, y_test))
```

```
0.271
```

In [83]:

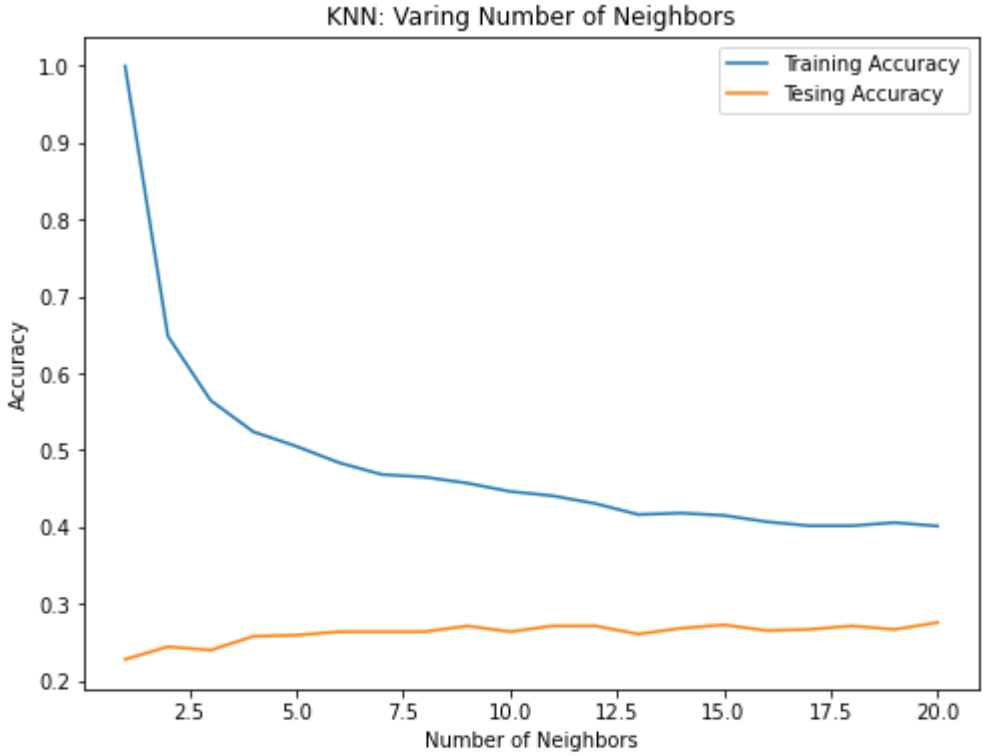
```
# Q2 (2.3)

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split




X = telecom_churn.drop("churn", axis = 1)
y = telecom_churn["customer_service_calls"]
#[["total_day_charge"], ["total_eve_charge"], ["total_night_charge"], ["custom


knn = KNeighborsClassifier(n_neighbors = 6)
X_train, X_test, y_train, y_test =train_test_split(X,y, test_size=0.25,random_

knn.fit(X_train, y_train)
print("This is 25% -",knn.score(X_test, y_test))

#---------------------------------------------------------


from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split




X = telecom_churn.drop("churn", axis = 1)
y = telecom_churn["customer_service_calls"]
#[["total_day_charge"], ["total_eve_charge"], ["total_night_charge"], ["custom


knn = KNeighborsClassifier(n_neighbors = 6)
X_train, X_test, y_train, y_test =train_test_split(X,y, test_size=0.20,random_

knn.fit(X_train, y_train)
print("This is 20% -",knn.score(X_test, y_test))
```

```
This is 25% - 0.2637889688249401
This is 20% - 0.263868065967016
```

In [85]:

```
# Q2 (2.4)

import matplotlib.pyplot as plt


train_accuracies = {}
test_accuracies = {}
neighbors = np.arange(1,21)
print(neighbors)
for neighbor in neighbors:
    knn = KNeighborsClassifier(n_neighbors = neighbor)
    knn.fit(X_train, y_train)
    train_accuracies[neighbor] = knn.score(X_train, y_train)
    test_accuracies[neighbor] = knn.score(X_test, y_test)

#print(train_accuracies.values())
my_train = list(train_accuracies.values())
my_test = list(test_accuracies.values())

plt.figure(figsize=(8,6))
plt.title("KNN: Varing Number of Neighbors")
plt.plot(neighbors, my_train, label="Training Accuracy")
plt.plot(neighbors, my_test, label="Tesing Accuracy")
plt.legend()
plt.xlabel("Number of Neighbors")
plt.ylabel("Accuracy")
plt.show()
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
```


KNN: Varing Number of Neighbors
Training Accuracy
Tesing Accuracy
Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Number of Neighbors

In [96]:

```
# Q2 (2.5)

X_new = np.array([[35.0,17.5,10.1,1],
                  [107.0,19.0,24.1,0],
                  [13.0,10.9,11.2,2],
                  [67.9,45.7,34.5,1],
                  ])


knn = KNeighborsClassifier(n_neighbors = 6)
X_train, X_test, y_train, y_test =train_test_split(X,y, test_size=0.3,random_s

knn.fit(X_train, y_train)
print(knn.score(X_test, y_test))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [96], in <cell line: 13>()
     10 knn = KNeighborsClassifier(n_neighbors = X_new)
     11 X_train, X_test, y_train, y_test =train_test_split(X,y, test_size=0.
3,random_state = 63)
---> 13 knn.fit(X_train, y_train)
     14 print(knn.score(X_test, y_test))

File ~\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:198,
in KNeighborsClassifier.fit(self, X, y)
    179 """Fit the k-nearest neighbors classifier from the training dataset.
    180 
    181 Parameters
   (...)
    194     The fitted k-nearest neighbors classifier.
    195 """
    196 self.weights = _check_weights(self.weights)
--> 198 return self._fit(X, y)

File ~\anaconda3\lib\site-packages\sklearn\neighbors\_base.py:568, in Neighbo
rsBase._fit(self, X, y)
    565     raise ValueError("algorithm = '%s' not recognized" % self.algorit
hm)
    567 if self.n_neighbors is not None:
--> 568     if self.n_neighbors <= 0:
    569         raise ValueError("Expected n_neighbors > 0. Got %d" % self.n_
neighbors)
    570     elif not isinstance(self.n_neighbors, numbers.Integral):

ValueError: The truth value of an array with more than one element is ambiguo
us. Use a.any() or a.all()
```

In [88]:

```
# Q2 (2.6)

CustName = pd.read_csv("newData.csv")
CustName

X_new = np.array([[35.0,17.5,10.1,1],
                 [107.0,19.0,24.1,0],
                 [13.0,10.9,11.2,2],
                 [67.9,45.7,34.5,1],
                 ])
```

Out[88]:

	cust_name	day_charge	eve_charge	night_charge	cust_service
0	Tom	13.4	11.50	29.4	35
1	Peter	65.3	23.10	34.2	4
2	John	14.7	41.20	29.1	3
3	Jack	90.2	9.21	32.1	1
4	Lin	51.3	2.31	45.6	0
5	Levy	10.1	10.80	13.5	9
6	Kim	78.4	9.70	8.5	8
7	Worch	30.4	63.70	64.2	6
8	Gorge	90.1	10.10	10.2	1
9	Jack	30.1	10.60	15.4	2

# Q2 (2.7)

From the predictions, in my opinion, the feature of the most likely condition do contribute to a churn. Because it is most likely to happen.

In [ ]:

```
#Lut Lat Aung,  6511163,  542
```