In [73]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [74]:
```
import pandas as pd
from fancyimpute import KNN
import numpy as np
```

In [75]:
```
data = pd.read_csv("churn_raw_data.csv")
data.head()
```

Out[75]:

	Unnamed: 0	CaseOrder	Customer_id	Interaction	City	State	County	Zip	
0	1	1	K409198	aa90260b-4141-4a24-8e36-b04ce1f4f77b	Point Baker	AK	Prince of Wales-Hyder	99927	56
1	2	2	S120509	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	West Branch	MI	Ogemaw	48661	44
2	3	3	K191035	344d114c-3736-4be5-98f7-c72c281e2d35	Yamhill	OR	Yamhill	97148	45
3	4	4	D90850	abfa2b40-2d43-4994-b15a-989b8c79e311	Del Mar	CA	San Diego	92014	32
4	5	5	K662701	68a861fd-0d20-4e51-a587-8a90407ee574	Needville	TX	Fort Bend	77461	29

5 rows × 52 columns

In [76]:
```
data_Job_testing = data.Job.str.split(",", expand = True)
data_Job_testing.head()
```

Out[76]:

	0	1
0	Environmental health practitioner	None
1	Programmer	multimedia
2	Chief Financial Officer	None
3	Solicitor	None
4	Medical illustrator	None

In [77]:
```
data.Job = data_Job_testing[0]
```

In [78]:
```
convert_yes_to_one = ["Churn","Phone",  "Techie", "Port_modem","Tablet","Multiple",
data[convert_yes_to_one].head()
```

Out[78]:

	Churn	Phone	Techie	Port_modem	Tablet	Multiple	OnlineSecurity	OnlineBackup	D
0	No	Yes	No	Yes	Yes	No	Yes	Yes	
1	Yes	Yes	Yes	No	Yes	Yes	Yes	No	
2	No	Yes	Yes	Yes	No	Yes	No	No	
3	No	Yes	Yes	No	No	No	Yes	No	
4	Yes	No	No	Yes	No	No	No	No	

In [79]:
```
convert_yes_to_one = ["Churn","Phone",  "Techie", "Port_modem","Tablet","Multiple",
data[convert_yes_to_one] = data[convert_yes_to_one].replace(["Yes","No"],[1,0])
data[convert_yes_to_one].head()
```

Out[79]:

	Churn	Phone	Techie	Port_modem	Tablet	Multiple	OnlineSecurity	OnlineBackup	D
0	0	1.0	0.0	1	1	0	1	1	
1	1	1.0	1.0	0	1	1	1	0	
2	0	1.0	1.0	1	0	1	0	0	
3	0	1.0	1.0	0	0	0	1	0	
4	1	0.0	0.0	1	0	0	0	0	

In [80]:
```
test1 = data.copy(deep = True)
```

In [81]:
```
zip = test1[test1.Population == 0].Zip.unique()
zip
```

Out[81]:
```
array([ 4228, 48397, 11359, 38132, 28019, 10020, 82646, 28629, 85726,
       19109, 99164,  5481, 70451,  7961, 10271, 41848, 95364, 90831,
       64102, 84515, 11351, 10177, 11451, 17120, 10153, 66760, 10154,
       56593, 20202, 89826, 47907, 48242, 19710, 13290, 73019, 86433,
       10170, 40231, 20045, 10169, 40434, 86443, 96850,  1199, 21105,
       98562, 98174, 38131, 13138, 35074, 76957, 29912, 98559, 90747,
       14893, 63902, 53031, 96155, 76523, 20701, 92338, 20053, 90506,
       95915,  2643, 89022, 89831, 99903, 21240, 78029, 21031, 37243,
       54561, 25002, 53792, 11424, 32399, 41762, 89446, 10152, 56741,
       30475, 30164], dtype=int64)
```

In [82]:
```
zip = test1[test1.Population == 0].Zip.unique()
# https://stackoverflow.com/questions/19966018/filling-missing-values-by-mean-in-ea
for x in zip:
    test1.loc[test1.Zip == x, "Population"] = np.nan
    test1.loc[test1.Zip == x, "Population"] = test1.groupby("Area").transform(lambd
```

In [83]:
```
test1[data.Population == 0].Population
```

Out[83]:
```
13       9545.084185
422      9547.954024
428      9996.750673
434      9550.824726
446      9553.696291
            ...
9216    10041.673982
9441    10083.783681
9657     9814.620712
9702    10086.798265
9944     9817.571590
Name: Population, Length: 97, dtype: float64
```

In [84]:
```
data.Population = test1.Population
```

In [85]:
```
df_knn = data.copy(deep = True)
```

In [86]:
```
# drop unusable strings
drop_columns = ["CaseOrder", "Customer_id", "Interaction","County","Timezone"]
#makes values dummies
dummy_columns = ["Area", "Employment","Education" ,"Marital", "Gender", "Contract",
#Cat Variables
cat_columns = ["Job","City","State"]
```

In [87]:
```
df_knn[cat_columns] = df_knn[cat_columns].astype('category')
for x in cat_columns:
    df_knn[x] = df_knn[x].cat.codes
```

In [88]:
```
df_knn = df_knn.drop(columns = drop_columns, axis = 1)
```

In [89]:
```
df_knn = pd.get_dummies(df_knn, columns = dummy_columns)
```

In [90]:
```
df_knn.dtypes.head(30)
```

Out[90]:
```
Unnamed: 0                int64
City                      int16
State                      int8
Zip                       int64
Lat                     float64
Lng                     float64
Population              float64
Job                       int16
Children                float64
Age                     float64
Income                  float64
Churn                     int64
Outage_sec_perweek      float64
Email                     int64
Contacts                  int64
Yearly_equip_failure      int64
Techie                  float64
Port_modem                int64
Tablet                    int64
Phone                   float64
Multiple                  int64
OnlineSecurity            int64
OnlineBackup              int64
DeviceProtection          int64
TechSupport             float64
StreamingTV               int64
StreamingMovies           int64
PaperlessBilling          int64
Tenure                  float64
MonthlyCharge           float64
dtype: object
```

In [91]:
```
knn_imputer = KNN(k =5)
```

In [92]:
```
df_knn.iloc[:,:] = knn_imputer = knn_imputer.fit_transform(df_knn)
```

```
Imputing row 1/10000 with 1 missing, elapsed time: 26.348
Imputing row 101/10000 with 2 missing, elapsed time: 26.355
Imputing row 201/10000 with 1 missing, elapsed time: 26.361
Imputing row 301/10000 with 2 missing, elapsed time: 26.367
Imputing row 401/10000 with 1 missing, elapsed time: 26.373
Imputing row 501/10000 with 1 missing, elapsed time: 26.379
Imputing row 601/10000 with 1 missing, elapsed time: 26.385
Imputing row 701/10000 with 0 missing, elapsed time: 26.390
Imputing row 801/10000 with 0 missing, elapsed time: 26.397
Imputing row 901/10000 with 0 missing, elapsed time: 26.403
Imputing row 1001/10000 with 0 missing, elapsed time: 26.409
Imputing row 1101/10000 with 0 missing, elapsed time: 26.416
Imputing row 1201/10000 with 3 missing, elapsed time: 26.421
Imputing row 1301/10000 with 1 missing, elapsed time: 26.427
Imputing row 1401/10000 with 3 missing, elapsed time: 26.434
Imputing row 1501/10000 with 2 missing, elapsed time: 26.441
Imputing row 1601/10000 with 1 missing, elapsed time: 26.447
Imputing row 1701/10000 with 2 missing, elapsed time: 26.452
Imputing row 1801/10000 with 0 missing, elapsed time: 26.459
Imputing row 1901/10000 with 1 missing, elapsed time: 26.465
Imputing row 2001/10000 with 2 missing, elapsed time: 26.471
Imputing row 2101/10000 with 2 missing, elapsed time: 26.477
Imputing row 2201/10000 with 3 missing, elapsed time: 26.483
Imputing row 2301/10000 with 2 missing, elapsed time: 26.488
Imputing row 2401/10000 with 1 missing, elapsed time: 26.494
Imputing row 2501/10000 with 0 missing, elapsed time: 26.501
Imputing row 2601/10000 with 2 missing, elapsed time: 26.507
Imputing row 2701/10000 with 1 missing, elapsed time: 26.514
Imputing row 2801/10000 with 0 missing, elapsed time: 26.520
Imputing row 2901/10000 with 1 missing, elapsed time: 26.526
Imputing row 3001/10000 with 5 missing, elapsed time: 26.533
Imputing row 3101/10000 with 0 missing, elapsed time: 26.539
Imputing row 3201/10000 with 1 missing, elapsed time: 26.546
Imputing row 3301/10000 with 0 missing, elapsed time: 26.552
Imputing row 3401/10000 with 1 missing, elapsed time: 26.558
Imputing row 3501/10000 with 2 missing, elapsed time: 26.565
Imputing row 3601/10000 with 2 missing, elapsed time: 26.572
Imputing row 3701/10000 with 0 missing, elapsed time: 26.578
Imputing row 3801/10000 with 1 missing, elapsed time: 26.584
Imputing row 3901/10000 with 2 missing, elapsed time: 26.589
Imputing row 4001/10000 with 4 missing, elapsed time: 26.595
Imputing row 4101/10000 with 3 missing, elapsed time: 26.602
Imputing row 4201/10000 with 3 missing, elapsed time: 26.608
Imputing row 4301/10000 with 1 missing, elapsed time: 26.615
Imputing row 4401/10000 with 1 missing, elapsed time: 26.621
Imputing row 4501/10000 with 1 missing, elapsed time: 26.627
Imputing row 4601/10000 with 0 missing, elapsed time: 26.633
Imputing row 4701/10000 with 1 missing, elapsed time: 26.640
Imputing row 4801/10000 with 3 missing, elapsed time: 26.645
Imputing row 4901/10000 with 1 missing, elapsed time: 26.652
Imputing row 5001/10000 with 1 missing, elapsed time: 26.658
Imputing row 5101/10000 with 0 missing, elapsed time: 26.664
Imputing row 5201/10000 with 2 missing, elapsed time: 26.669
Imputing row 5301/10000 with 2 missing, elapsed time: 26.676
Imputing row 5401/10000 with 1 missing, elapsed time: 26.682
Imputing row 5501/10000 with 3 missing, elapsed time: 26.688
```

```
Imputing row 5601/10000 with 2 missing, elapsed time: 26.695
Imputing row 5701/10000 with 1 missing, elapsed time: 26.701
Imputing row 5801/10000 with 0 missing, elapsed time: 26.707
Imputing row 5901/10000 with 1 missing, elapsed time: 26.713
Imputing row 6001/10000 with 3 missing, elapsed time: 26.719
Imputing row 6101/10000 with 2 missing, elapsed time: 26.725
Imputing row 6201/10000 with 3 missing, elapsed time: 26.730
Imputing row 6301/10000 with 2 missing, elapsed time: 26.736
Imputing row 6401/10000 with 3 missing, elapsed time: 26.742
Imputing row 6501/10000 with 0 missing, elapsed time: 26.748
Imputing row 6601/10000 with 2 missing, elapsed time: 26.754
Imputing row 6701/10000 with 1 missing, elapsed time: 26.762
Imputing row 6801/10000 with 0 missing, elapsed time: 26.768
Imputing row 6901/10000 with 1 missing, elapsed time: 26.775
Imputing row 7001/10000 with 1 missing, elapsed time: 26.782
Imputing row 7101/10000 with 3 missing, elapsed time: 26.789
Imputing row 7201/10000 with 1 missing, elapsed time: 26.795
Imputing row 7301/10000 with 0 missing, elapsed time: 26.801
Imputing row 7401/10000 with 1 missing, elapsed time: 26.806
Imputing row 7501/10000 with 1 missing, elapsed time: 26.813
Imputing row 7601/10000 with 2 missing, elapsed time: 26.819
Imputing row 7701/10000 with 2 missing, elapsed time: 26.825
Imputing row 7801/10000 with 1 missing, elapsed time: 26.832
Imputing row 7901/10000 with 0 missing, elapsed time: 26.838
Imputing row 8001/10000 with 2 missing, elapsed time: 26.843
Imputing row 8101/10000 with 1 missing, elapsed time: 26.850
Imputing row 8201/10000 with 2 missing, elapsed time: 26.857
Imputing row 8301/10000 with 1 missing, elapsed time: 26.863
Imputing row 8401/10000 with 2 missing, elapsed time: 26.869
Imputing row 8501/10000 with 3 missing, elapsed time: 26.875
Imputing row 8601/10000 with 0 missing, elapsed time: 26.881
Imputing row 8701/10000 with 1 missing, elapsed time: 26.888
Imputing row 8801/10000 with 0 missing, elapsed time: 26.894
Imputing row 8901/10000 with 0 missing, elapsed time: 26.900
Imputing row 9001/10000 with 1 missing, elapsed time: 26.906
Imputing row 9101/10000 with 3 missing, elapsed time: 26.913
Imputing row 9201/10000 with 2 missing, elapsed time: 26.918
Imputing row 9301/10000 with 1 missing, elapsed time: 26.924
Imputing row 9401/10000 with 2 missing, elapsed time: 26.931
Imputing row 9501/10000 with 3 missing, elapsed time: 26.937
Imputing row 9601/10000 with 2 missing, elapsed time: 26.944
Imputing row 9701/10000 with 3 missing, elapsed time: 26.949
Imputing row 9801/10000 with 1 missing, elapsed time: 26.957
Imputing row 9901/10000 with 0 missing, elapsed time: 26.964
```

In [93]:
```
missing = ["Children", "Age", "Income", "Techie", "Phone", "TechSupport", "Tenure",
```

In [94]:
```
df_knn[missing].head()
```

Out[94]:

	Children	Age	Income	Techie	Phone	TechSupport	Tenure	Bandwidth_GB_Yea
0	0.897025	68.0	28561.990000	0.0	1.0	0.0	6.795513	904.53611
1	1.000000	27.0	21704.770000	1.0	1.0	0.0	1.156681	800.98276
2	4.000000	50.0	58483.698806	1.0	1.0	0.0	15.754144	2054.70696
3	1.000000	48.0	18925.230000	1.0	1.0	0.0	17.087227	2164.57941
4	0.000000	83.0	40074.190000	0.0	0.0	1.0	1.670972	271.49343

In [95]:
```
df_knn[missing_data].isna().sum()
```

Out[95]:
```
Children            0
Age                 0
Income              0
Techie              0
Phone               0
TechSupport         0
Tenure              0
Bandwidth_GB_Year   0
dtype: int64
```

In [96]:
```
data[missing_data] = df_knn[missing_data]
```

In [97]:
```
data.dtypes
```

Out[97]:
```
Unnamed: 0                int64
CaseOrder                 int64
Customer_id              object
Interaction              object
City                     object
State                    object
County                   object
Zip                       int64
Lat                     float64
Lng                     float64
Population              float64
Area                     object
Timezone                 object
Job                      object
Children                float64
Age                     float64
Education                object
Employment               object
Income                  float64
Marital                  object
Gender                   object
Churn                     int64
Outage_sec_perweek      float64
Email                     int64
Contacts                  int64
Yearly_equip_failure      int64
Techie                  float64
Contract                 object
Port_modem                int64
Tablet                    int64
InternetService          object
Phone                   float64
Multiple                  int64
OnlineSecurity            int64
OnlineBackup              int64
DeviceProtection          int64
TechSupport             float64
StreamingTV               int64
StreamingMovies           int64
PaperlessBilling          int64
PaymentMethod            object
Tenure                  float64
MonthlyCharge           float64
Bandwidth_GB_Year       float64
item1                     int64
item2                     int64
item3                     int64
item4                     int64
item5                     int64
item6                     int64
item7                     int64
item8                     int64
dtype: object
```

In [98]:
```
missing.remove("Income")
missing.remove("Tenure")
```

```
missing.remove("Bandwidth_GB_Year")
```

In [99]:
```
data[missing] = round(data[missing]).astype(int)
data[missing]
```

Out[99]:

	Children	Age	Techie	Phone	TechSupport
0	1	68	0	1	0
1	1	27	1	1	0
2	4	50	1	1	0
3	1	48	1	1	0
4	0	83	0	0	1
...	...	...	...	...	...
9995	3	43	0	1	0
9996	4	48	0	1	0
9997	0	87	0	1	0
9998	1	39	0	0	1
9999	1	28	0	1	0

10000 rows × 5 columns

In [100…
```
data.to_csv("CleanData.Csv")
```