A.

1.

Can we use KNN models to learn which customers are more likely to churn based on the data we have?

2.

One goal of the data analysis is to identify customers that are more likely to churn, so the sales team can pro-actively offer them monthly sales specials.

B.

1.

KNN classifies unlabeled observations based on the nearest labeled observation's data points. By nearness we mean the most similar predictor variable values. An expected outcome would be: If the nearest K (chosen by analyst) number of data points to the new observation have churned, then the new unclassified observations will be classified to churn. If the nearest neighbors have not churned, the new observation will be classified as not churn.
This should happen for all predictions.

2.

A core assumption of KNN is:
The closer two given points are to each other, the more related and similar they are.

(Hachcham, 2023)

3.

I have chosen python and the sklearn.neighbors.KNeighborsClassifier.

1) pandas will be used for transforming, manipulating, and cleaning data.

2) sklearn.neighbors.KNeighborsClassifier will be used to create the KNN classifier and make classification predictions.

3) sklearn.preprocessing import MinMaxScaler will be used to normalize numeric data.

```
4) sklearn.model_selection import train_test_split will be used to split the training and test data.

5) sklearn.metrics import accuracy_score will be used to calculate accuracy of the model.

6) sklearn.metrics import roc_auc_score will be used to calculate the area under the curve score.
```

C.

1.

```
One preprocessing goal for KNN would be normalization of data.
```

2.

```
Predictors:
I will be using
'Age','Income','Outage_sec_perweek','Contacts','Yearly_equip_failure','Tenure','Bandwidth_GB_Year','Age'
which are continuous.

I will also be using
'Gender','Area','InternetService','Phone','OnlineSecurity','DeviceProtection','StreamingMovies',and
'OnlineBackup' which are categorical.

Predicted:
Predicted variable will be churn which is categorical.
```

3.

read in data

In [1]:

```
#import libraries and read in the data from file.
import pandas as pd
# read in the data
file_path = '/home/dj/skewl/d209/churn_clean.csv'
pd.set_option('display.max_columns', None)
# Read the data from the CSV file into a DataFrame
df = pd.read_csv(file_path)
#drop index column
df = df.loc[:, ~df.columns.str.contains('Unnamed')]
```

find duplicate rows

In [2]:
```python
# Find duplicate rows
duplicate_rows = df.duplicated(["CaseOrder"]).sum()

# Print duplicate rows   # found NO duplicate rows here!
print(duplicate_rows)
```

```
0
```

identify missing values

In [3]:
```python
# Identify missing values using isna() method
missing_values = df.isna().sum()
# Print DataFrame with True for missing values and False for non-missing values
print(missing_values)

# no missing values here!
```

```
CaseOrder               0
Customer_id             0
Interaction             0
UID                     0
City                    0
State                   0
County                  0
Zip                     0
Lat                     0
Lng                     0
Population              0
Area                    0
TimeZone                0
Job                     0
Children                0
Age                     0
Income                  0
Marital                 0
Gender                  0
Churn                   0
Outage_sec_perweek      0
Email                   0
Contacts                0
Yearly_equip_failure    0
Techie                  0
Contract                0
Port_modem              0
Tablet                  0
InternetService         0
Phone                   0
Multiple                0
OnlineSecurity          0
OnlineBackup            0
DeviceProtection        0
TechSupport             0
StreamingTV             0
StreamingMovies         0
PaperlessBilling        0
PaymentMethod           0
Tenure                  0
MonthlyCharge           0
Bandwidth_GB_Year       0
Item1                   0
Item2                   0
Item3                   0
Item4                   0
Item5                   0
Item6                   0
```

```
Item7                0
Item8                0
dtype: int64
```

Check for outliers

In [4]:
```
df.describe()
```

Out[4]:

	CaseOrder	Zip	Lat	Lng	Population	Children	Age	Income	Outage_sec_pe
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.0000	10000.000000	10000.000000	10000.0
mean	5000.50000	49153.319600	38.757567	-90.782536	9756.562400	2.0877	53.078400	39806.926771	10.0
std	2886.89568	27532.196108	5.437389	15.156142	14432.698671	2.1472	20.698882	28199.916702	2.9
min	1.00000	601.000000	17.966120	-171.688150	0.000000	0.0000	18.000000	348.670000	0.0
25%	2500.75000	26292.500000	35.341828	-97.082812	738.000000	0.0000	35.000000	19224.717500	8.0
50%	5000.50000	48869.500000	39.395800	-87.918800	2910.500000	1.0000	53.000000	33170.605000	10.0
75%	7500.25000	71866.500000	42.106908	-80.088745	13168.000000	3.0000	71.000000	53246.170000	11.9
max	10000.00000	99929.000000	70.640660	-65.667850	111850.000000	10.0000	89.000000	258900.700000	21.2

encode data

In [5]:
```
#split continuous and categorical variables into separate dataframes
dfcon = df[['Age','Income','Outage_sec_perweek','Contacts','Yearly_equip_failure','Tenure','Bandwidth_GB_Year','Age']]
dfcat = df[['Gender','Area','InternetService','Phone','OnlineSecurity','DeviceProtection','StreamingMovies','OnlineBack
#split data into x and y
y = df['Churn']
#one-hot encode categorical data and drop first level of each
dfcat_encoded = pd.get_dummies(dfcat,drop_first=True)
#concatenate the columns
x = pd.concat([dfcon, dfcat_encoded], axis=1)
```

normalize data

In [6]:
```python
#normalize data after encoding
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_normalized = pd.DataFrame(scaler.fit_transform(x), columns=x.columns)
```

In [7]:
```python
#write the prepared data to .csv file
x['Churn'] = y
x.to_csv('prepared-data.csv', index=False)
del x['Churn']
```

D.

1.

In [8]:
```python
from sklearn.model_selection import train_test_split
## split into training and test
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.2, random_state=35)
# write to csv
X_train.to_csv('x_train.csv', index=False)
X_test.to_csv('x_test.csv', index=False)
Y_train.to_csv('y_train.csv', index=False)
Y_test.to_csv('y_test.csv', index=False)
```

2.

I am using KNN classification with a K value of 5 to train a model that will predict which customers are more likely to churn.

I import the library from sklearn.neighbors import KNeighborsClassifier. Then I instantiate the classification model with the k=5. Then I fit the model with the training predictor variables, and training dependent variable data. Lastly I predict the classifications for the test data.

3.

In [9]:
```python
from sklearn.neighbors import KNeighborsClassifier


# Initialize the KNN classifier
k = 5  # Number of neighbors
knn = KNeighborsClassifier(n_neighbors=k)

# Train the KNN classifier
knn.fit(X_train, Y_train)
```

```python
# Predict the labels for the test set
y_pred = knn.predict(X_test)
```

In [10]:
```python
from sklearn.metrics import accuracy_score

Y_pred = knn.predict(X_test)

# Evaluate the accuracy of the model
accuracy = accuracy_score(Y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7125
```

In [11]:
```python
from sklearn.metrics import roc_auc_score
# Predict the probabilities of the positive class for the test data
Y_probs = knn.predict_proba(X_test)[:, 1]

# Compute the AUC score
auc_score = roc_auc_score(Y_test, Y_probs)
print("AUC Score:", auc_score)
```

```
AUC Score: 0.7380209872510746
```

E.

1.

```
 The accuracy of my model is  0.731. This means that it correctly predicted 73% of of all predictions
made.

 The AUC of my model is 0.7602. This means that across all probability threshold values it was able
to distinguish between negative and positive classes at a score of 0.76.
 The AUC measurement of pure chance is 0.5.
```

2.

```
 My classification model can predict if a customer will churn with 73% accuracy. It also has an AUC
score of 76% which is 26% better than random chance. The implications are that we can with some level
of statistical certainty predict which customers are going to churn.
```

3.

```
One limit of this data analysis is that the model isn't as accurate as I would like it to be.
```

4.

```
I think we should use this model to predict which customers are going to churn and have the sales
team reach out and offer sales and promotions to the customers that the KNN model identified as
likely to churn to mitigate churn.
```

citations

Hachcham, A. (2023, August 11). The KNN Algorithm – Explanation, Opportunities, Limitations. neptune.ai. https://neptune.ai/blog/knn-algorithm-explanation-opportunities-limitations#:~:text=A%20core%20assumption%20of%20KNN,related%20and%20similar%20they%20are.&text=Typically%20used%20with%20d

K-nearest Neighbors (KNN) Classification Model. (n.d.). ritchieng.github.io. https://www.ritchieng.com/machine-learning-k-nearest-neighbors-knn/

Srivastava, T. (2024, January 4). A Complete Guide to K-Nearest Neighbors (Updated 2024). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

In [ ]: