In [ ]:

A.

1.

Can we use classification tree 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.

When the decision tree is trained on a data set it creates a sequence of if else statements based on the known feature data. Each concerning one feature with a split point. At the end of the decison tree is a classification label. When a prediction needs to be made about a label, the known features of the new observation are tested against the if else statements or so called split points. This happens all the way through the decision tree, starting at the root and finally after many decisions it will stop and the classification label will be assigned. I chose a decision tree because they are non parametric and scalable.

The outcome would be to input some feature data with an unknown classification or label and have the decision tree make decisions and assign it a label. In my example this would look like inputting some customer data and having the decision tree classify it's label as churn true or churn false.

2.

Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.

(Chauhan, n.d.)

3.

I have chosen python and the sklearn.tree.DecisionTreeClassifier.

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

2) sklearn.tree.DecisionTreeClassifier will be used to create the tree classifier and make classification predictions.

3) sklearn.model_selection import train_test_split to split data into train and test segments.

4) sklearn.metrics import accuracy_score to measure accuracy of the model.

5) sklearn.tree import DecisionTreeRegressor to measure regression performance.

6) sklearn.metrics import mean_squared_error to calculate the mean squared error.

C.

1.

One preprocessing goal for decision tree would be encoding categorical 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]:
```
# 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]:
```
# 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_per
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)
```

In [6]:
```
#write the prepared data to .csv file
x['Churn'] = y
x.to_csv('prepared-data.csv', index=False)
del x['Churn']
```

D.

1.

In [7]:
```
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,stratify=y, 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 sklearn decisionTreeClassifier to train a model that will predict which customers are more likely to churn.

I import the library from sklearn.tree import DecisionTreeClassifier. Then I instantiate the classification model with the random state of 50. 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 [8]:
```

from sklearn.tree import DecisionTreeClassifier

# Initialize the decision tree classifier
tree_classifier = DecisionTreeClassifier(random_state=42)
# Train the decision tree classifier
tree_classifier.fit(X_train, Y_train)
# Predict the labels for the test set
Y_pred = tree_classifier.predict(X_test)
```

In [9]:
```
from sklearn.metrics import accuracy_score

Y_pred = tree_classifier.predict(X_test)

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

```
Accuracy: 0.78
```

In [10]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
```

```
# Initialize the decision tree regressor
tree_regressor = DecisionTreeRegressor(random_state=50)
#map strings to numerics
Y_binary_test = Y_test.map({'No':0,'Yes':1})
Y_binary_train = Y_train.map({'No':0,'Yes':1})
# Train the decision tree regressor
tree_regressor.fit(X_train, Y_binary_train)

# Make predictions on the test data
Y_pred = tree_regressor.predict(X_test)
# Calculate the Mean Squared Error (MSE)
mse = mean_squared_error(Y_binary_test, Y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 0.225
```

E.

1.

The accuracy of my model is 0.78. This means that it correctly predicted 78% of of all predictions made.

The MSE of my model is 0.225.

The goal of a regression tree is to generate a line that best fits the data.

MSE is the average squared difference between the actual data values and where the data point would be on the proposed line. The tree runs an algorithm that finds the line that results in the smallest MSE.

(MSE And Variance Reduction in Regression Trees, n.d.)

2.

My classification model can predict if a customer will churn with 73% accuracy. It also has an MSE of .225. 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 decision tree model
identified as likely to churn, to mitigate churn.
```

citations

Chauhan, N. S. (n.d.). Decision Tree Algorithm, Explained - KDnuggets. KDnuggets. https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html#:~:text=Assumptions%20while%20creating%20Decision%20Tree&text=In%20the%20beginning%2C%20the%20whole,the%20b

MSE and variance reduction in regression trees. (n.d.). Cross Validated. https://stats.stackexchange.com/questions/479842/mse-and-variance-reduction-in-regression-trees#:~:text=The%20goal%20of%20a%20regression,results%20in%20the%20smallest%20MSE.

Decision Tree Regression. (n.d.). Scikit-learn. https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

In [ ]: