Part 1: Research Question

A1) Research Question

The question that will be answered is if we can create a model to accurately predict what a customer’s monthly charge will be based on various features. We will answer this using a decision tree.

A2) Defined Goal

The goal is to create a viable model to help determine which features are most important when it comes to what a new customer’s monthly charge will be.

Part 2: Method Justification

B1) Prediction Method

A decision tree analyzes data by breaking it down into smaller subsets with similar values. Each of these has a leaf node and decision node that contains more branches. They can handle both categorical and numeric data, require less computation time and not sensitive to outliers. Therefore, using a decision tree will be a good fit for the data set.

B2) Summary of Assumption

One assumption of using a decision tree is that it is non-parametric, being that it does not make distribution assumptions on the data. Therefore, decisions can deal with most types of data efficiently.

B3) Packages and Libraries

|  |  |
| --- | --- |
| Pandas | Used for importing data into a DataFrame and manipulations |
| Numpy | Provides arrays for required calculations |
| Seaborn | Better visualizations |
| Matplotlib.pyplot | Graphs and visualizations |
| Sklearn.model\_selection  Import train\_test\_split | Splitting the data into a training and a testing set |
| Sklearn.model\_selection  Import GridSearchCV | Allows us to use gridsearch cross-validation |
| Sklearn.tree  Import DecisionTreeRegressor | Allows us to use a decision tree |
| Sklearn.metrics  Imoprt mean\_squared\_error | Lets us find the MSE |

Part 3: Data Summary and Implications

C1) Preprocessing Goal

Two goals for preprocessing are to check for any and all missing or null values, then impute any found. As well as encoding any categorical (yes / no) variables into 1’s and 0’s using dummy variables.

C2) Identifying Dataset Variables

For this analysis I will be using the following variables:

|  |  |
| --- | --- |
| Variable | Type |
| Children | Continuous |
| Outage\_sec\_perweek | Continuous |
| Yearly\_equip\_failure | Continuous |
| Monthly\_charge | Continuous |
| Bandwidth\_GB\_Year | Continuous |
| Tenure | Continuous |
| Churn | Categorical |
| Techie | Categorical |
| Contract | Categorical |
| Port\_modem | Categorical |
| Tablet | Categorical |
| InternetService | Categorical |
| Phone | Categorical |
| Multiple | Categorical |
| OnlineSecurity | Categorical |
| OnlineBackup | Categorical |
| DeviceProtection | Categorical |
| TechSupport | Categorical |
| StreamingTV | Categorical |
| StreamingMovies | Categorical |

C3) Steps for Analysis and Code

**Step 1** – Load dataset into Python and describe the data

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*#Import all packages*

*import pandas as pd*

*import numpy as np*

*from pandas import Series, DataFrame*

*import seaborn as sns*

*import matplotlib.pyplot as plt*

*%matplotlib inline*

*import sklearn*

*from sklearn.tree import DecisionTreeRegressor*

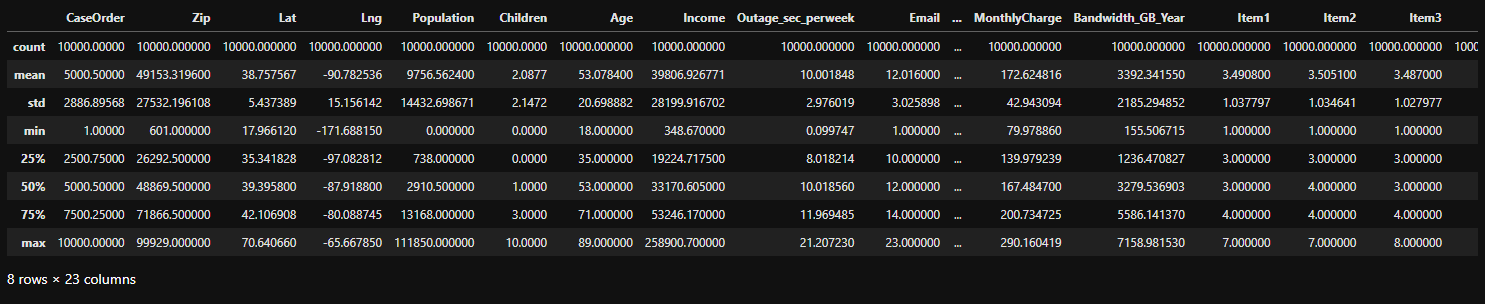
*from sklearn.model\_selection import train\_test\_split, GridSearchCV*

*from sklearn.metrics import mean\_squared\_error*

*#Load the data set into Pandas*

*df = pd.read\_csv('churn\_clean.csv')*

*df.describe()*

**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step 2** – Delete less meaningful columns from the dataset

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*#Drop the less meaningul columns from the data set*

*df = df.drop(columns = ['CaseOrder', 'Customer\_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Age', 'Marital', 'Gender', 'Email', 'PaperlessBilling', 'PaymentMethod', 'Contacts', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'])*

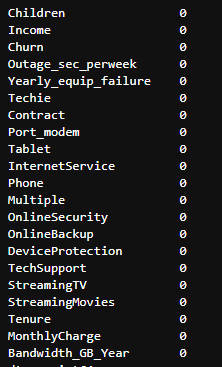
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Step 3** – Check for any missing data, impute any found

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*#Search for missing data*

*df.isnull().sum()*

**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

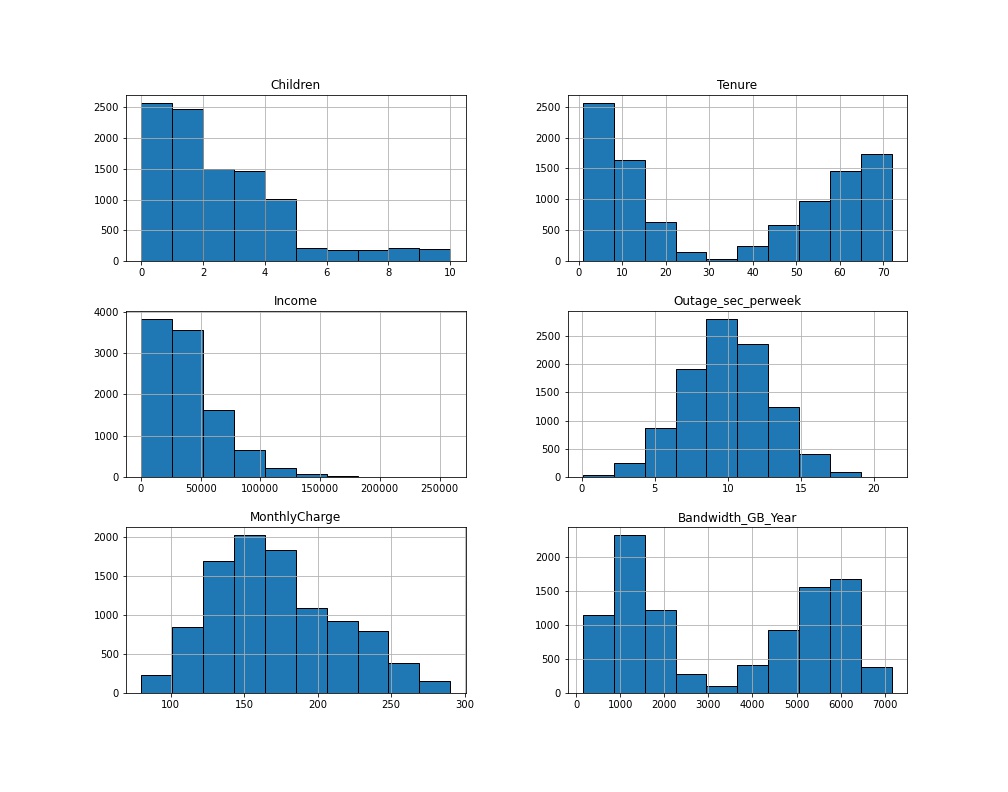
**Step 4** – Create some graphs for data visualization

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*#Create histograms for some continuous variables*

*df[['Children', 'Tenure', 'Income', 'Outage\_sec\_perweek', 'MonthlyCharge', 'Bandwidth\_GB\_Year']].hist(ec = "black", figsize = (14, 11))*

*#plt.savefig('Histogram1.jpg')*

**

*#For bivariate statistics create some scatterplots with a few variables with our response variable as the y-axis*

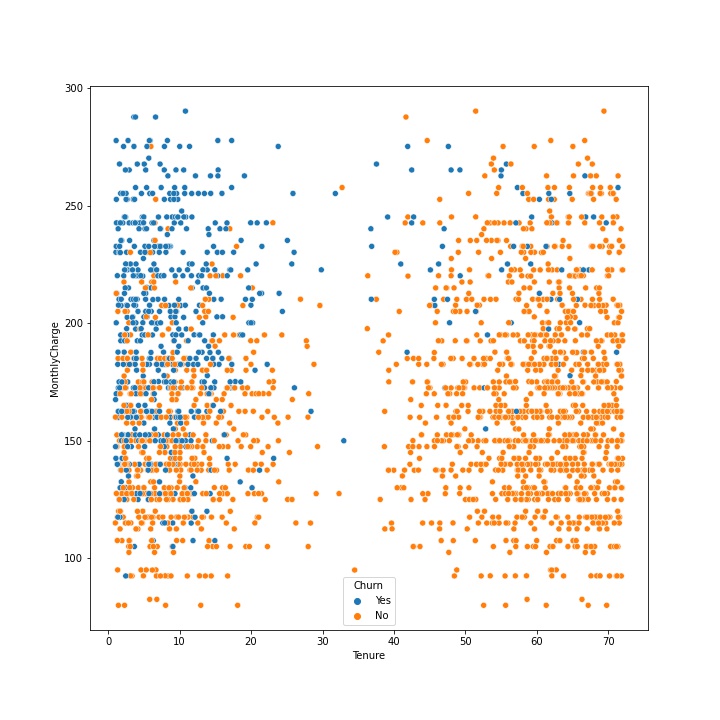
*#First, creating a random sampling of 25% of the data set for the scatterplots to improve visibility[3]*

*subset = df.sample(frac = 0.25)*

*ax = plt.subplots(figsize = (10, 7))*

*ax = sns.scatterplot(data = subset, x = "Tenure", y = "MonthlyCharge", hue = "Churn")*

*#plt.savefig('Tenure.jpg')*

**

*ax = plt.subplots(figsize = (10, 10))*

*ax = sns.scatterplot(data = subset, x = "Bandwidth\_GB\_Year", y = "MonthlyCharge", hue = "Churn")*

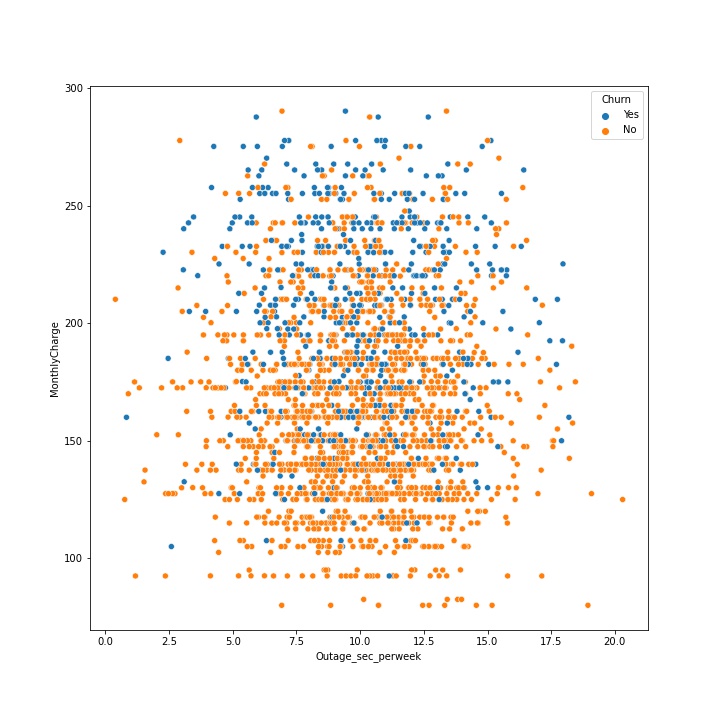
*#plt.savefig('Bandwidth.jpg')*

**

*ax = plt.subplots(figsize = (10, 10))*

*ax = sns.scatterplot(data = subset, x = "Outage\_sec\_perweek", y = "MonthlyCharge", hue = "Churn")*

*#plt.savefig('Outage.jpg')*

**

*ax = plt.subplots(figsize = (10, 10))*

*ax = sns.scatterplot(data = subset, x = "Income", y = "MonthlyCharge", hue = "Churn")*

*#plt.savefig('Income.jpg')*

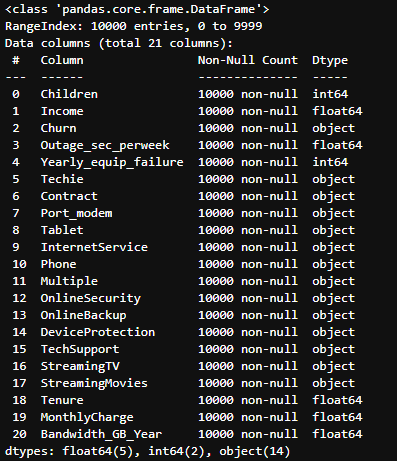
**

**Step 5** – Encode categorical variables to numeric ones using dummy variables, then extract the prepared dataset

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*#See which variables are categorical*

*df.info()*

**

*#No missing data, now to use ordinal encoding to replace the categorical values with numeric ones*

*#Yes to 1, No to 0*

*df['Churn\_num'] = df['Churn']*

*df['Techie\_num'] = df['Techie']*

*df['Port\_modem\_num'] = df['Port\_modem']*

*df['Tablet\_num'] = df['Tablet']*

*df['Phone\_num'] = df['Phone']*

*df['Multiple\_num'] = df['Multiple']*

*df['OnlineSecurity\_num'] = df['OnlineSecurity']*

*df['OnlineBackup\_num'] = df['OnlineBackup']*

*df['DeviceProtection\_num'] = df['DeviceProtection']*

*df['TechSupport\_num'] = df['TechSupport']*

*df['StreamingTV\_num'] = df['StreamingTV']*

*df['StreamingMovies\_num'] = df['StreamingMovies']*

*df['Contract\_num'] = df['Contract']*

*df['InternetService\_num'] = df['InternetService']*

*#Set up dictionary for converting to numeric values*

*dict\_churn = {"Churn\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_techie = {"Techie\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_modem = {"Port\_modem\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_tablet = {"Tablet\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_phone = {"Phone\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_multiple = {"Multiple\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_security = {"OnlineSecurity\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_backup = {"OnlineBackup\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_protection = {"DeviceProtection\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_tech = {"TechSupport\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_tv = {"StreamingTV\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_movie = {"StreamingMovies\_num" : {"Yes" : 1, "No" : 0}}*

*dict\_contract = {"Contract\_num" : {"Month-to-month" : 0, "One year" : 1, "Two Year" : 2}}*

*dict\_isp = {"InternetService\_num" : {"None" : 0, "DSL" : 1, "Fiber Optic" : 2}}*

*#Replace the variables values*

*df.replace(dict\_churn, inplace = True)*

*df.replace(dict\_techie, inplace = True)*

*df.replace(dict\_modem, inplace = True)*

*df.replace(dict\_tablet, inplace = True)*

*df.replace(dict\_phone, inplace = True)*

*df.replace(dict\_multiple, inplace = True)*

*df.replace(dict\_security, inplace = True)*

*df.replace(dict\_backup, inplace = True)*

*df.replace(dict\_protection, inplace = True)*

*df.replace(dict\_tech, inplace = True)*

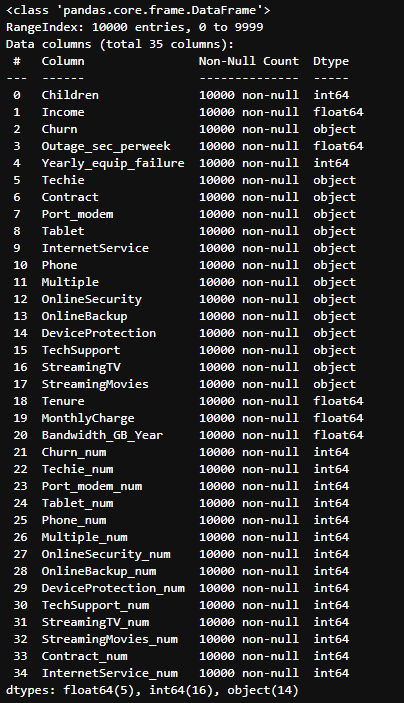
*df.replace(dict\_tv, inplace = True)*

*df.replace(dict\_movie, inplace = True)*

*df.replace(dict\_contract, inplace = True)*

*df.replace(dict\_isp, inplace = True)*

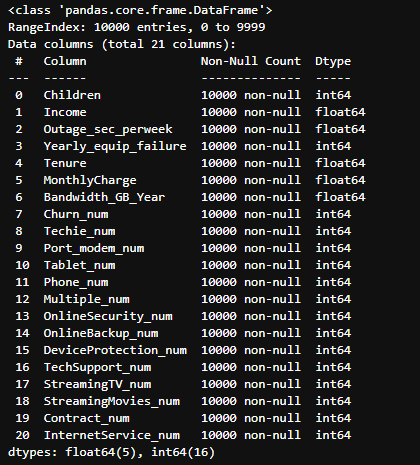
*df.info()*

**

*#Now that we have those as numeric, we can drop the original columns*

*df = df.drop(columns = ['Churn', 'Techie', 'Port\_modem', 'Tablet', 'Phone', 'Multiple', 'OnlineSecurity','OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'InternetService'])*

*df.info()*

**

*#Extract prepared dataset*

*df.to\_csv('churn\_prepared.csv')*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Part 4: Analysis

D1) Train / Test Split

Attached.

D2) Describe Technique

The way I will go about this analysis will first be to split the data into 80% training and 20% testing sets. I’ll use GridSearchCV to find the best max\_depth and min\_sample\_leaf to use for the decision tree. Finally I’ll fit the model using the best parameters I found through hyperparameter tuning, then find the mean squared error, root mean squared error and accuracy score.

D3) Code

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*#Import all packages*

*import pandas as pd*

*import numpy as np*

*from pandas import Series, DataFrame*

*import seaborn as sns*

*import matplotlib.pyplot as plt*

*%matplotlib inline*

*import sklearn*

*from sklearn.tree import DecisionTreeRegressor*

*from sklearn.model\_selection import train\_test\_split, GridSearchCV*

*from sklearn.metrics import mean\_squared\_error*

*#Load the data set into Pandas*

*df = pd.read\_csv('churn\_prepared.csv', index\_col = 0)*

*df.describe()*

*#Set up for train - test split*

*X = df.drop('MonthlyCharge', axis = 1).values*

*y = df['MonthlyCharge'].values*

*#Split the data set with an 80/20 split*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size = 0.8, test\_size = 0.2, random\_state = 25)*

*#Save the training and testing sets as csv files*

*pd.DataFrame(X\_train).to\_csv('X\_train.csv')*

*pd.DataFrame(X\_test).to\_csv('X\_test.csv')*

*pd.DataFrame(y\_train).to\_csv('y\_train.csv')*

*pd.DataFrame(y\_test).to\_csv('y\_test.csv')*

*#Run gridsearch cv to find best max depth and leaves [1][2]*

*dt = DecisionTreeRegressor()*

*param\_grid = {"max\_depth": range(1, 8),*

*"min\_samples\_leaf": range(1, 5)}*

*grid = GridSearchCV(dt, param\_grid, cv = 5)*

*grid.fit(X\_train, y\_train)*

*print('The best parameters for this model: {}'.format(grid.best\_params\_))*

**

*#Fit the decision tree model with the found parameters*

*dt = DecisionTreeRegressor(max\_depth = 7, min\_samples\_leaf = 1)*

*dt.fit(X\_train, y\_train)*

*y\_pred = dt.predict(X\_test)*

*mse\_dt = mean\_squared\_error(y\_test, y\_pred)*

*rmse\_dt = mse\_dt\*\*(1/2)*

*print("Accuracy: ", dt.score(X\_test, y\_test))*

*print("Mean squared error: ", mse\_dt)*

*print("Root mean squared error: ", rmse\_dt)*

**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Part 5: Data Summary and Implications

E1) Accuracy and MSE

As seen above, after fitting the decision tree to the data set the accuracy, being the performance of the model found by dividing the number of corrected predictions by the total number of predictions was 0.98. The mean squared error, which is a measure of how close a regression line is to a set of points, is 22.58. [4] The root mean squared error, which is the standard deviation of the residuals, is 4.75. [5]

E2) Results and Implications

With the accuracy score shown above we can determine that it seems to be a pretty strong model meaning that with the predictor variables we used we can make a good determination of what a customers monthly charge would be.

E3) Limitation

One limitation is that for large data sets, decision trees aren’t as good for regression and can easily cause over-fitting so the accuracy might not be as good as it seems.

E4) Course of Action

From here the best course of action would be to try training a model of another prediction method and compare the results.

Part 6: Demonstration

F) Panopto Video

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=305bcb44-9248-412d-83db-aef20124a200

G) Third-Party Code Sources

[1] Saini, B. (2021, January 28). *Hyperparameter tuning of decision tree classifier using GRIDSEARCHCV*. Medium. Retrieved August 5, 2022, from https://ai.plainenglish.io/hyperparameter-tuning-of-decision-tree-classifier-using-gridsearchcv-2a6ebcaffeda

[2] *Decision tree regression with hyper parameter tuning*. Decision Tree Regression With Hyper Parameter Tuning In Python. (n.d.). Retrieved August 5, 2022, from https://www.nbshare.io/notebook/312837011/Decision-Tree-Regression-With-Hyper-Parameter-Tuning-In-Python/

[3] Duca, Angelica Lo. “How to Sample a Dataframe in Python Pandas.” *Medium*, Towards Data Science, 7 July 2021, https://towardsdatascience.com/how-to-sample-a-dataframe-in-python-pandas-d18a3187139b.

J) Sources

[4] *Mean squared error: Definition and example*. Statistics How To. (2021, June 26). Retrieved August 5, 2022, from https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/mean-squared-error/

[5] *RMSE: Root mean square error*. Statistics How To. (2021, May 31). Retrieved August 5, 2022, from https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/