Data Mining

D209: Task 2

Shantel Johnson

April 6, 2023

Contents

[Section A 2](#_Toc132056872)

[Part 1: Research Question 2](#_Toc132056873)

[Part 2: Analytical Goal 2](#_Toc132056874)

[Section B 2](#_Toc132056875)

[Part 1: Classification Method 2](#_Toc132056876)

[Part 2: Assumption 2](#_Toc132056877)

[Part 3: Packages 2](#_Toc132056878)

[Section C 3](#_Toc132056879)

[Part 1: Preprocessing Goal 3](#_Toc132056880)

[Part 2: Variables 3](#_Toc132056881)

[Part 3: Steps 3](#_Toc132056882)

[Part 4: Preprocessing Output 3](#_Toc132056883)

[Section D 3](#_Toc132056884)

[Part 1: Data Partition Output 3](#_Toc132056885)

[Part 2: Analysis 3](#_Toc132056886)

[Part 3: Code 4](#_Toc132056887)

[Section E 4](#_Toc132056888)

[Part 1: Model Evaluation 4](#_Toc132056889)

[Part 2: Results 4](#_Toc132056890)

[Part 3: Limitation 4](#_Toc132056891)

[Part 4: Recommendation 4](#_Toc132056892)

[Section F 4](#_Toc132056893)

[Part 1: Data Preparation Code 4](#_Toc132056894)

[Part 2: Decision Tree Regressor Model Code 9](#_Toc132056895)

[Decision Tree Regressor 9](#_Toc132056896)

[Section G 12](#_Toc132056897)

[Part 1: Demonstration 12](#_Toc132056898)

[Section H 12](#_Toc132056899)

[Part 1: Web Sources 12](#_Toc132056900)

[Part 2: References 12](#_Toc132056901)

# Section A

## Part 1: Research Question

The dataset consists of AnyTelecom’s observations on customer demographics, their telecom services, and their associated monthly charges. As the project’s data analyst, it is my job to answer the research question: What will a customer’s monthly charges be? To answer this question, I will use decision tree regression.

## Part 2: Analytical Goal

The goal of this project is to build a machine learning model that predicts a hypothetical customer’s monthly charges based on their telecom package and service subscriptions. The information for this hypothetical customer is included in the following table:

|  |  |
| --- | --- |
| **Variable** | **Value** |
| Bandwidth\_GB\_Year | 1000 |
| Port\_modem | Yes |
| Tablet | Yes |
| Phone | Yes |
| Multiple | Yes |
| OnlineSecurity | Yes |
| OnlineBackup | No |
| DeviceProtection | Yes |
| TechSupport | Yes |
| StreamingTV | Yes |
| StreamingMovies | Yes |
| PaperlessBilling | Yes |

# Section B

## Part 1: Classification Method

A decision tree (or tree model) is a set of if-then-else rules that discover hidden patterns corresponding to complex interactions in data. Simple tree models can be expressed in terms of predictor relationships that are easily interpretable, and the expected outcome of a decision tree regressor is a continuous value (Bruce, Bruce, & Gedeck, 2019).

## Part 2: Assumption

One assumption of a decision tree regressor is that the predictive performance is judged by the root mean squared error (RMSE) of each partition (Bruce, Bruce, & Gedeck, 2019).

## Part 3: Packages

The following libraries will be used in support of the analysis:

|  |  |  |
| --- | --- | --- |
| **Library** | **Module** | **Purpose** |
| pandas | DataFrame | Data manipulation & analysis |
| pandas | cut | Feature transformation |
| scipy | stats | Calculate z-scores |
| sklearn | model\_selection | Partition data |
| sklearn | tree | Classification model |
| sklearn | metrics | Evaluate model |

# Section C

## Part 1: Preprocessing Goal

Sklearn’s implementation of decision trees does not currently support categorical variables (Pedregosa, 2011). One preprocessing goal is to convert categorical predictor variables into continuous variables using ordinal encoding.

## Part 2: Variables

Data descriptions for all predictor variables used in the analysis, including whether a particular variable is considered continuous or categorical, can be viewed in Section F, Part 1.

## Part 3: Steps

The data was prepared according to the following steps:

1. Data quality issues were identified.
   1. Find duplicates using the pandas.DataFrame.duplicated() function.
   2. Find missing values using the pandas.DataFrame.isnull().sum() function.
   3. Find outliers by applying the scipy.stats.zscore() function on continuous variables in the dataset.
2. Since no data quality issues were identified, none had to be remediated.
3. Categorical variables were converted to continuous variables using the pandas.DataFrame.replace() function.

To view the associated data preparation code, refer to Section F, Part 1.

## Part 4: Preprocessing Output

Refer to the attachment “churn prepped2.csv” to view the output of the data cleaning process.

# Section D

## Part 1: Data Partition Output

Refer to the attachments “train\_data2.csv” and “test\_data2.csv” to view the output of partitioning the data.

## Part 2: Analysis

As stated in Section A, Part 1, the dataset was analyzed using decision tree regression. The analysis was conducted according to the following steps:

1. The data was partitioned into train and test sets using the sklearn.model\_selction.train\_test\_split() function.
2. The decision tree regressor was fit with the predictor and target variables in the training set using the sklearn.tree.DecisionRegressor() function.
3. The model was evaluated.
   1. The model’s mean squared error (MSE) was calculated with the predictor and target variables in the test set using the sklearn.metrics.accuracy\_mean\_squared\_error() function.
   2. The model’s accuracy was determined by the RMSE. To obtain the RMSE, python’s internal math.sqrt() function was applied on the MSE (part a of this step).

## Part 3: Code

Refer to Section F, Part 2 to view the code associated with the analysis classification.

# Section E

## Part 1: Model Evaluation

The decision tree regression model was evaluated for MSE and accuracy (RMSE). According to the output, the model’s MSE was approximately 247.96, while the accuracy was approximately 15.75.

## Part 2: Results

The predictor variables used in the decision tree regressor’s decision path were StreamingMovies, Multiple (whether a customer has multiple phone lines), StreamingTV, and OnlineBackup. Using these 4 variables, the model can predict a customer’s monthly charges.

Recall that the information for the hypothetical customer was defined in Section A, Part 2. After plugging this information into the model, the predicted monthly charges for this customer is. $238.86. Refer to Section F, Part 2 for more.

## Part 3: Limitation

To reduce the complexity of the model, the max depth of the decision tree regressor was set to 4. It is possible that increasing the max depth could improve the performance of the model (at the risk of increasing the model’s complexity and overfitting the results).

## Part 4: Recommendation

As a next step, I recommend experimenting with the model to find the max depth that yields the best performance with the lowest complexity. Once this step is complete, the data engineering team can scale the model for production use.

# Section F

## Part 1: Data Preparation Code

In [2]:

# read data into DataFrame using panda

import pandas as pd

# Title: panda-dev/pandas

# Author: The pandas development team

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7741580

df = pd.read\_csv('source\_output/churn\_clean.csv', header='infer')

df = df[['MonthlyCharge',

'Bandwidth\_GB\_Year',

'Port\_modem',

'Tablet',

'Phone',

'Multiple',

'OnlineSecurity',

'OnlineBackup',

'DeviceProtection',

'TechSupport',

'StreamingTV',

'StreamingMovies',

'PaperlessBilling']]

df.head(5)

Out[2]:

|  | **MonthlyCharge** | **Bandwidth\_GB\_Year** | **Port\_modem** | **Tablet** | **Phone** | **Multiple** | **OnlineSecurity** | **OnlineBackup** | **DeviceProtection** | **TechSupport** | **StreamingTV** | **StreamingMovies** | **PaperlessBilling** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 172.455519 | 904.536110 | Yes | Yes | Yes | No | Yes | Yes | No | No | No | Yes | Yes |
| **1** | 242.632554 | 800.982766 | No | Yes | Yes | Yes | Yes | No | No | No | Yes | Yes | Yes |
| **2** | 159.947583 | 2054.706961 | Yes | No | Yes | Yes | No | No | No | No | No | Yes | Yes |
| **3** | 119.956840 | 2164.579412 | No | No | Yes | No | Yes | No | No | No | Yes | No | Yes |
| **4** | 149.948316 | 271.493436 | Yes | No | No | No | No | No | No | Yes | Yes | No | No |

In [3]:

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10000 entries, 0 to 9999

Data columns (total 13 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 MonthlyCharge 10000 non-null float64

1 Bandwidth\_GB\_Year 10000 non-null float64

2 Port\_modem 10000 non-null object

3 Tablet 10000 non-null object

4 Phone 10000 non-null object

5 Multiple 10000 non-null object

6 OnlineSecurity 10000 non-null object

7 OnlineBackup 10000 non-null object

8 DeviceProtection 10000 non-null object

9 TechSupport 10000 non-null object

10 StreamingTV 10000 non-null object

11 StreamingMovies 10000 non-null object

12 PaperlessBilling 10000 non-null object

dtypes: float64(2), object(11)

memory usage: 1015.8+ KB

#### Data Cleaning[¶](#Data-Cleaning)

##### Duplicates[¶](#Duplicates)

In [4]:

# check for duplication

df[df.duplicated()]

Out[4]:

|  | **MonthlyCharge** | **Bandwidth\_GB\_Year** | **Port\_modem** | **Tablet** | **Phone** | **Multiple** | **OnlineSecurity** | **OnlineBackup** | **DeviceProtection** | **TechSupport** | **StreamingTV** | **StreamingMovies** | **PaperlessBilling** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

##### Missing Values[¶](#Missing-Values)

In [5]:

# check for missing values

df.isnull().sum()

Out[5]:

MonthlyCharge 0

Bandwidth\_GB\_Year 0

Port\_modem 0

Tablet 0

Phone 0

Multiple 0

OnlineSecurity 0

OnlineBackup 0

DeviceProtection 0

TechSupport 0

StreamingTV 0

StreamingMovies 0

PaperlessBilling 0

dtype: int64

##### Outliers[¶](#Outliers)

In [6]:

# check for outliers

# import scipy.stats to calculate z-scores

# Title: scipy/scipy: Scipy

# Author: Gommers, et al.

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7655153

from scipy import stats

outliers = df.select\_dtypes(include='number')

outliers = stats.zscore(outliers)

outliers[outliers.abs() >= 3].count()

Out[6]:

MonthlyCharge 0

Bandwidth\_GB\_Year 0

dtype: int64

#### Data Preparation[¶](#Data-Preparation)

##### Data Description: Categorical Variables[¶](#Data-Description:-Categorical-Variables)

In [8]:

df.select\_dtypes(exclude='number').describe()

Out[8]:

|  | **Port\_modem** | **Tablet** | **Phone** | **Multiple** | **OnlineSecurity** | **OnlineBackup** | **DeviceProtection** | **TechSupport** | **StreamingTV** | **StreamingMovies** | **PaperlessBilling** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 10000 | 10000 | 10000 | 10000 | 10000 | 10000 | 10000 | 10000 | 10000 | 10000 | 10000 |
| **unique** | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **top** | No | No | Yes | No | No | No | No | No | No | No | Yes |
| **freq** | 5166 | 7009 | 9067 | 5392 | 6424 | 5494 | 5614 | 6250 | 5071 | 5110 | 5882 |

##### Data Description: Continuous Variables[¶](#Data-Description:-Continuous-Variables)

In [9]:

df.select\_dtypes(include='number').describe()

Out[9]:

|  | **MonthlyCharge** | **Bandwidth\_GB\_Year** |
| --- | --- | --- |
| **count** | 10000.000000 | 10000.000000 |
| **mean** | 172.624816 | 3392.341550 |
| **std** | 42.943094 | 2185.294852 |
| **min** | 79.978860 | 155.506715 |
| **25%** | 139.979239 | 1236.470827 |
| **50%** | 167.484700 | 3279.536903 |
| **75%** | 200.734725 | 5586.141370 |
| **max** | 290.160419 | 7158.981530 |

#### Data Transformation[¶](#Data-Transformation)

In [11]:

# perform ordinal encoding

df.replace('No', 0, inplace=True)

df.replace('Yes', 1, inplace=True)

df.head(5)

Out[11]:

|  | **MonthlyCharge** | **Bandwidth\_GB\_Year** | **Port\_modem** | **Tablet** | **Phone** | **Multiple** | **OnlineSecurity** | **OnlineBackup** | **DeviceProtection** | **TechSupport** | **StreamingTV** | **StreamingMovies** | **PaperlessBilling** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 172.455519 | 904.536110 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| **1** | 242.632554 | 800.982766 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| **2** | 159.947583 | 2054.706961 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| **3** | 119.956840 | 2164.579412 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| **4** | 149.948316 | 271.493436 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

#### Output[¶](#Output)

In [12]:

df.to\_csv("source\_output/churn\_prepped2.csv")

In [ ]:

## Part 2: Decision Tree Regressor Model Code

### Decision Tree Regressor[¶](" \l "Decision-Tree-Regressor)

In [1]:

# read prepped data using pandas

# Title: panda-dev/pandas

# Author: The pandas development team

# Date: 2023

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.7741580

import pandas as pd

df = pd.read\_csv("source\_output/churn\_prepped2.csv", header='infer', index\_col=0)

df.head(5)

Out[1]:

|  | **MonthlyCharge** | **Children** | **Age** | **Tenure** | **Bandwidth\_GB\_Year** | **Port\_modem** | **Tablet** | **Phone** | **Multiple** | **OnlineSecurity** | **OnlineBackup** | **DeviceProtection** | **TechSupport** | **StreamingTV** | **StreamingMovies** | **PaperlessBilling** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 172.455519 | 0 | 68 | 6.795513 | 904.536110 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| **1** | 242.632554 | 1 | 27 | 1.156681 | 800.982766 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| **2** | 159.947583 | 4 | 50 | 15.754144 | 2054.706961 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| **3** | 119.956840 | 1 | 48 | 17.087227 | 2164.579412 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| **4** | 149.948316 | 0 | 83 | 1.670972 | 271.493436 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

In [2]:

# partition data

# Title: scikit-learn/scikit-learn

# Author: Grisel, et al.

# Date: 2022

# Code Version: latest

# Availability: https://doi.org/10.5281/zenodo.6543413

from sklearn.model\_selection import train\_test\_split

X = df.drop(columns='MonthlyCharge')

y = df['MonthlyCharge']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

#### Output[¶](#Output)

In [3]:

X\_train.to\_csv("source\_output/train\_data2.csv")

X\_test.to\_csv("source\_output/test\_data2.csv")

#### Model[¶](#Model)

In [4]:

# assign variables and fit model

from sklearn.tree import (DecisionTreeRegressor, plot\_tree)

regressor = DecisionTreeRegressor(criterion='friedman\_mse', max\_depth=4, min\_samples\_leaf=5, min\_impurity\_decrease=0.00)

regressor.fit(X\_train, y\_train)

Out[4]:

DecisionTreeRegressor(criterion='friedman\_mse', max\_depth=4, min\_samples\_leaf=5)

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.   
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

DecisionTreeRegressor

DecisionTreeRegressor(criterion='friedman\_mse', max\_depth=4, min\_samples\_leaf=5)

#### Decision Tree[¶](#Decision-Tree)

In [5]:

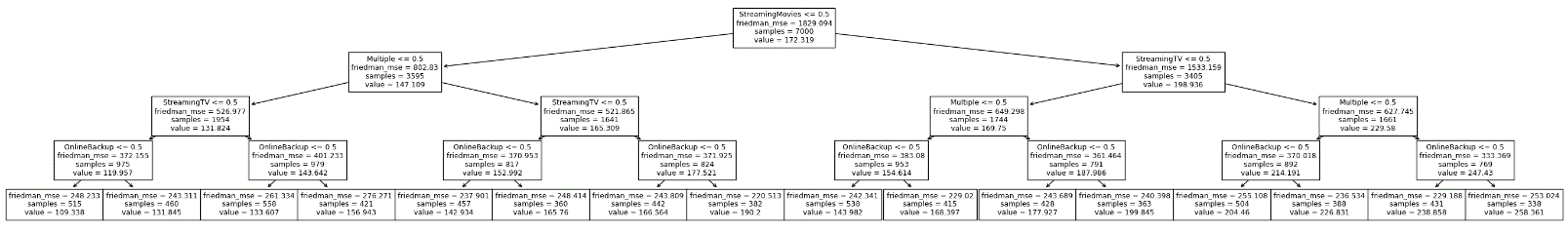
# import matplotlib and plot the decision tree

import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(35, 5))

plot\_tree(regressor, feature\_names=X.columns, ax=ax, fontsize=9)

plt.show()



#### Evaluation: R-squared[¶](#Evaluation:-R-squared)

In [6]:

# calculate coefficient of determination, R-squared

regressor.score(X\_test, y\_test)

Out[6]:

0.8679509136474834

#### Evaluation: Mean Squared Error[¶](#Evaluation:-Mean-Squared-Error)

In [7]:

# calculate MSE

from sklearn.metrics import mean\_squared\_error as mse

y\_pred = regressor.predict(X\_test)

MSE = mse(y\_test, y\_pred)

MSE

Out[7]:

247.9618648945777

#### Evaluation: Root Mean Squared Error[¶](#Evaluation:-Mean-Squared-Error)

In [8]:

# calculate RMSE

import math

math.sqrt(MSE)

Out[8]:

15.746804910666091

In [ ]:

# Section G

## Part 1: Demonstration

To view a walkthrough demonstration of the code and programming environment, view the following Panopto link: …

# Section H

## Part 1: Web Sources

Gommers, et al. (2023). doi:10.5281/zenodo.7655153

Grisel, et al. (2022). doi:10.5281/zenodo.6543413

The pandas development team. (2023). doi:10.5281/zenodo.7741580

## Part 2: References

Bruce, P., Bruce, A., & Gedeck, P. (2019). *Practical Statistics for Data Scientists : 50+ Essential Concepts Using R and Python* (2 ed.). O'Reilly Media, Incorporated. Retrieved April 2023

Pedregosa, e. (2011). *scikit-learn*. Retrieved April 2023, from sklearn.feature\_selection.SequentialFeatureSelector: https://scikit-learn.org/stable/modules/generated/sklearn.feature\_selection.SequentialFeatureSelector.html