D209 – Data Mining (Task 2)

Morrell J. Parrish

Western Governors University

Table of Contents

D209 – Data Mining (Task 2)

A. Research Question

During this course of research, we will determine which customers are at a higher risk of churn and which features (variables) can be an indicator for churn.

A2. Analysis Goal

The objective of this analysis is to use the decision tree methodology to determine which variables determine if a customer will churn or not; this analysis will reduce the number of our predictor variables down to the most significant one(s). "The churn rate, also known as the rate of attrition or customer churn; is the frequency in which consumers discontinue doing business with a company. It is commonly represented as the percentage of service subscribers who cancel their memberships within a specified time frame" (Frankenfield, 2022).

B. Chosen Technique

This analysis will employ the ***decision tree methodology***. The ***decision tree methodology*** is a widely used data mining method for developing prediction algorithms for a target variable or establishing classification systems based on multiple covariates. This model was chosen due to its simplicity; a decision tree is a "flowchart-like tree structure in which each internal node represents a test on an attribute, each branch represents a test outcome, and each leaf node (terminal node) holds a class label" (Geeks for Geeks, 2002).

B2. Assumption

One assumption of decision trees is that feature values should be categorical. If the values are continuous, they are discretized before the model is built. Recursively, records are distributed based on attribute values.

B3. Packages/Libraries Use Justification

The following packages/libraries will be used for this analysis:

- Pandas
    - used to read and manipulate data via series (one-dimensional structure) or dataframes (multi-dimensional data structure)
- NumPy
    - used to perform mathematical computations
- Matplotlib
    - used to create visualization (plotting and graphing)
- Seaborn
    - used to create visualization (plotting and graphing)
- Scikit-learn
    - used to perform scientific computations
    - used to split our data into training and test sets
    - used for predicting and classification analysis
- SciPy
    - used for scientific and technical computation
- Graphviz
    - used to create graph objects, which can be completed using different nodes and edges
- DMBA
    - used in data mining for business analytics
- PIL
    - Python imaging library, that adds image processing capabilities

C. Data Preparation Description

One goal of data preprocessing (**data cleaning/mining**) is to make the training/testing process easier by appropriately transforming and scaling the entire dataset. Before training machine learning models, preprocessing is required. Outliers are removed during preprocessing, and the features are scaled to an equivalent range (Misra et al., 2020).

To use the churn dataset in our analysis we will first need to prepare the data. The following steps were taken to prepare the dataset for analysis:

- download the churn dataset
- determine which variables will be used in the analysis
- import the dataset into *PyCharm*
- remove independent variables, demographics, and personal identification variables not being used in the analysis
  - caseorder, customer_id, interaction, UID, city, state, county, zip, lat, lng, population, timezone, job, email, contacts
- determine if any outliners exist and remove them

C2. Variable Identification and Classification

The **continuous variables** (16) that will be used in this analysis will include age, children, income**,** outage_sec_perweek, yearly_equip_failure, tenure, monthlycharge, **b**andwidth_GB_Year, item1 (timelyresponse), item2 (fixes), item3 (replacements), item4 (reliability), item5 (options), item6 (respectfulness), item7 (courteous), and item8 (listening).

The **categorical variables** (19) that will be used in this analysis will include area, marital, gender, churn, techie, contract, portmodem, tablet, internetservice, phone, multiple, onlinesecurity, onlinebackup, deviceprotection, techsupport, streamingtv, streamingmovies, paperlessbilling, paymentmethod.

C3. Data Preparation Steps

To use the churn dataset in our analysis we will first need to prepare the data:

- import the dataset into *Python (PyCharm)*
- view the dataframe's description, structure, and data types
- view summary statistics
- evaluate the dataset, remove null or missing values
- remove any outliners
- remove demographics, and personal identification
    - caseorder, customer_id, interaction, UID, city, state, county, zip, lat, lng, population, area, timezone, job, email, contacts

The below code was used to prepare our data:

```python
# Standard data science imports
import numpy as np
import pandas as pd
from pandas import Series, DataFrame
import scipy
import scipy.stats as stats
import csv

# Visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.patches as patches
# import matplotlib.pylab as plt
import graphviz

# PIL image
from PIL import Image as pNg

# Scikit-learn
import sklearn
from sklearn.metrics import confusion_matrix
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import classification_report
from sklearn import metrics, tree
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.neighbors import NearestNeighbors
from sklearn.model_selection import KFold, cross_val_score, train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, _tree
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.pipeline import FeatureUnion, Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler, LabelEncoder
from sklearn.tree import export_graphviz as dt  # decisiontree
from sklearn.base import BaseEstimator, TransformerMixin

# plot DecisionTree
from dmba import plotDecisionTree, classificationSummary, regressionSummary

# Import helper files
from helper import *

# Ignore Warning Code
import warnings

warnings.filterwarnings('ignore')

# Load data set into Pandas dataframe
df = pd.read_csv('churn_clean.csv')

# Remove less meaningful demographic variables
df = df.drop(columns=['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng',
                      'Population', 'TimeZone', 'Email', 'Contacts', 'Job'])
```

```python
# Display Churn dataframe
print(df)
print('\n')
# Rename last 8 columns
df.rename(columns={'Item1': 'TimelyResponse', 'Item2': 'Fixes', 'Item3': 'Replacements', 'Item4': 'Reliability',
                   'Item5': 'Options', 'Item6': 'Respectfulness', 'Item7': 'Courteous', 'Item8': 'Listening'},
          inplace=True)

# Get column info
print(df.info())

print('\n')
# Describe Churn dataset
print(df.describe())

# Save stats summary to excel
df.describe().to_excel('summary_stat.xlsx', index=False)
```

C4. Cleaned Data Set

The prepared dataset used for this analysis has been uploaded with the assessment file.

D. Analysis

The training and test datasets used for this analysis have been uploaded with the assessment file.

D2. Analytical Technique Description

Our analytical technique includes the following steps: (1) read in or load the data using Pandas' ***read( )*** function – in this case it will be our cleaned churn data set, (2) check and verify datatypes using Panda's ***info ( )*** function, (3) verify summary statistics using the ***describe ( )*** function, (4) set predictor and target variables; split up the dataset into inputs (X) and our target variable (y) using Pandas' ***drop( )*** function – this allows you to drop the target variable from the dataframe and store it in the variable 'X', (5) define both the categorical and numerical features, (6) split the dataset into training and test sets using Scikit-learn's function '***train_test_split',*** (7) extract both training and test datasets, (8) create model - this can be done using the ***DecisionTreeClassifier ( )*** and ***fit ( )*** functions , (9) plot the decision tree variable and boundaries, (10) verify model's accuracy via the ***classification summary ( )*** function,

```
# Load data set into Pandas dataframe
df = pd.read_csv('churn_clean.csv')
```

```
# Get column info
print(df.info())
```

```
Data columns (total 35 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Area                  10000 non-null  object
 1   Children              10000 non-null  int64
 2   Age                   10000 non-null  int64
 3   Income                10000 non-null  float64
 4   Marital               10000 non-null  object
 5   Gender                10000 non-null  object
 6   Churn                 10000 non-null  object
 7   Outage_sec_perweek    10000 non-null  float64
 8   Yearly_equip_failure  10000 non-null  int64
 9   Techie                10000 non-null  object
 10  Contract              10000 non-null  object
 11  Port_modem            10000 non-null  object
 12  Tablet                10000 non-null  object
 13  InternetService       10000 non-null  object
 14  Phone                 10000 non-null  object
 15  Multiple              10000 non-null  object
 16  OnlineSecurity        10000 non-null  object
 17  OnlineBackup          10000 non-null  object
 18  DeviceProtection      10000 non-null  object
 19  TechSupport           10000 non-null  object
 20  StreamingTV           10000 non-null  object
 21  StreamingMovies       10000 non-null  object
 22  PaperlessBilling      10000 non-null  object
 23  PaymentMethod         10000 non-null  object
 24  Tenure                10000 non-null  float64
 25  MonthlyCharge         10000 non-null  float64
 26  Bandwidth_GB_Year     10000 non-null  float64
 27  TimelyResponse        10000 non-null  int64
 28  Fixes                 10000 non-null  int64
 29  Replacements          10000 non-null  int64
 30  Reliability           10000 non-null  int64
 31  Options               10000 non-null  int64
 32  Respectfulness        10000 non-null  int64
 33  Courteous             10000 non-null  int64
 34  Listening             10000 non-null  int64
dtypes: float64(5), int64(11), object(19)
```

```
# Describe Churn dataset
print(df.describe())
```

	Count	Mean	STD	Min	25%	50%	75%	Max
Children	10000	2.0877	2.147200446	0	0	1	3	10
Age	10000	53.0784	20.69888156	18	35	53	71	89
Income	10000	39806.92677	28199.9167	348.67	19224.7175	33170.605	53246.17	258900.7
Outage_sec_perweek	10000	10.00184816	2.976019188	0.09974694	8.018214	10.01856	11.969485	21.20723
Yearly_equip_failure	10000	0.398	0.635953177	0	0	0	1	6
Tenure	10000	34.52618809	26.44306263	1.00025934	7.917693592	35.430507	61.479795	71.99928
MonthlyCharge	10000	172.6248162	42.94309411	79.97886	139.979239	167.4847	200.734725	290.160419
Bandwidth_GB_Year	10000	3392.34155	2185.294852	155.5067148	1236.470827	3279.536903	5586.14137	7158.98153
TimelyResponse	10000	3.4908	1.037797216	1	3	3	4	7
Fixes	10000	3.5051	1.034640536	1	3	4	4	7
Replacements	10000	3.487	1.027976981	1	3	3	4	8
Reliability	10000	3.4975	1.025816251	1	3	3	4	7
Options	10000	3.4929	1.024819309	1	3	3	4	7
Respectfulness	10000	3.4973	1.033585768	1	3	3	4	8
Courteous	10000	3.5095	1.028501595	1	3	4	4	7
Listening	10000	3.4956	1.028633292	1	3	3	4	8

```
# Set predictor variables and target variable

target = 'Churn'

X = df.drop(columns=[target])
y = df[target]
```

```
# Split dataset into training and test set
# Define primary feature and target data
target = 'Churn'

X = df.loc[:, df.columns != target]
y = df.loc[:, df.columns == target]

# Train the test set

tts = train_test_split(X, y, test_size=0.3, random_state=13)
(X_train, X_test, y_train, y_test) = tts

print('\n')
print('X_train: {}'.format(X_train.shape))
print('y_train: {}'.format(y_train.shape))
print('X_test: {}'.format(X_test.shape))
print('y_test: {}'.format(y_test.shape))
```

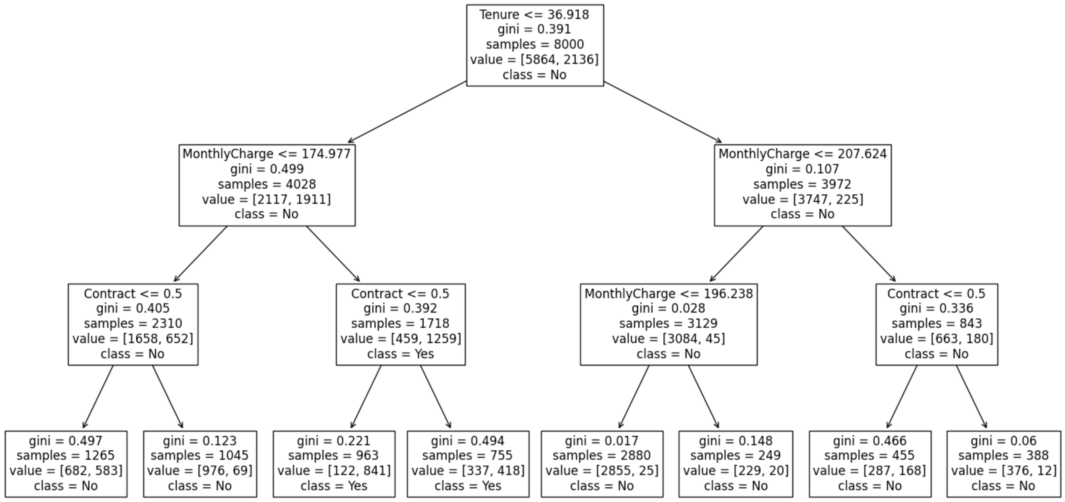
```
X_train: (7000, 34)
y_train: (7000, 1)
X_test: (3000, 34)
y_test: (3000, 1)
```

```
# Create model

dt = DecisionTreeClassifier(max_depth=2, random_state=13)
dt.fit(X_train, y_train)

print('Target: [{}: {}]'.format(target, ', '.join(dt.classes_)))
plt.figure(figsize=(10, 10))
_ = tree.plot_tree(dt, feature_names=X_train.columns.to_list(), class_names=dt.classes_,
                   filled=False, fontsize=12, rounded=False)
plt.show()
# plot decision boundaries

fig, ax = plt.subplots()
fig.set_size_inches(8, 5)
```

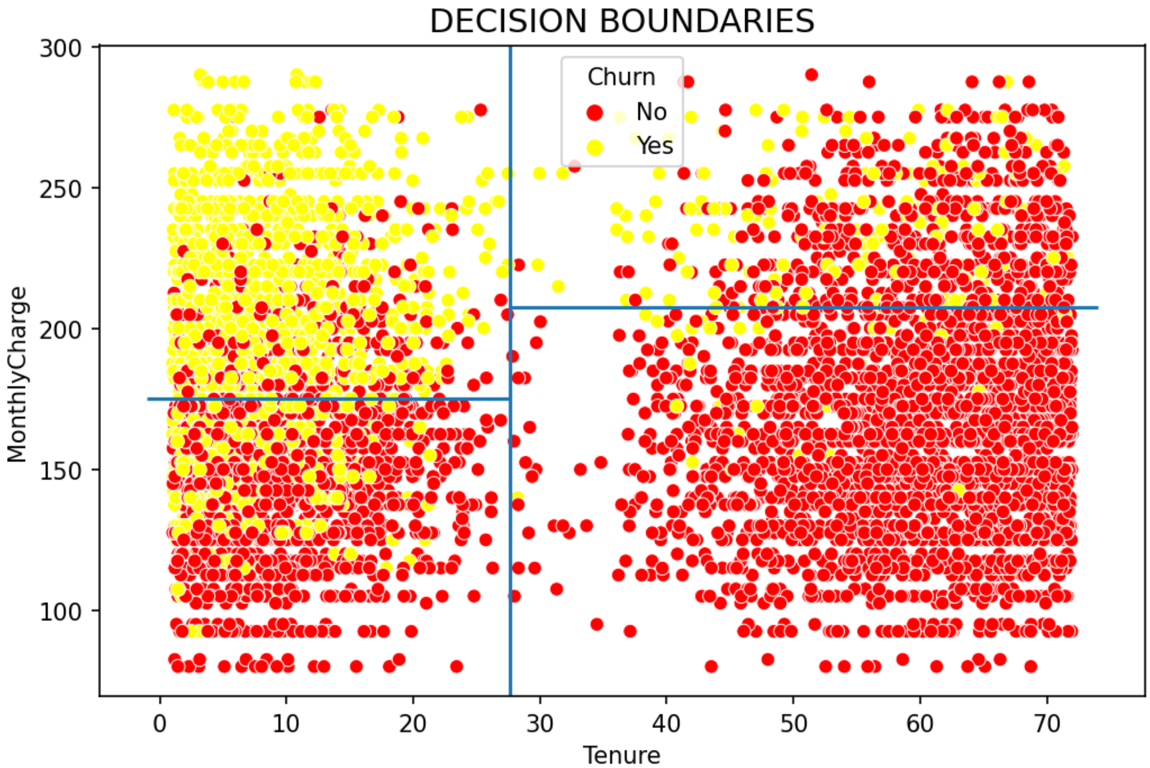



```
# define plot variables

x = 'Tenure'
y = 'MonthlyCharge'
title = 'Decision Boundaries'

sns.scatterplot(x=x, y=y,
                palette=['red', 'yellow'], hue=target,
                data=y_train.merge(X_train, left_index=True, right_index=True))
ax.axvline(x=27.659)
ax.hlines(y=174.977, xmin=-1, xmax=27.659)
ax.hlines(y=207.624, xmin=27.659, xmax=74)

ax.set_title(title.upper(), fontsize=14)
```



The above scatterplot represents the four terminal nodes of our decision tree

D3. Classification Analysis Code

```
# Set predictor variables and target variable

target = 'Churn'

X = df.drop(columns=[target])
y = df[target]
# Define primary feature and target data
target = 'Churn'

X = df.loc[:, df.columns != target]
y = df.loc[:, df.columns == target]

# Split dataset into training and test set
tts = train_test_split(X, y, test_size=0.3, random_state=13)
(X_train, X_test, y_train, y_test) = tts
# Create model

dt = DecisionTreeClassifier(max_depth=2, random_state=13)
dt.fit(X_train, y_train)

print('Target: [{}: {}]'.format(target, ', '.join(dt.classes_)))
plt.figure(figsize=(10, 10))
_ = tree.plot_tree(dt, feature_names=X_train.columns.to_list(), class_names=dt.classes_,
                   filled=False, fontsize=12, rounded=False)
plt.show()
# plot decision boundaries
```

```
# Print training dataset classification summary

classificationSummary(y_train, dt.predict(X_train))
```

```
Confusion Matrix (Accuracy 0.8330)

       Prediction
Actual    0    1
     0 5405  459
     1  877 1259
```

```
# Print test dataset classification summary

classificationSummary(y_test, dt.predict(X_test))
```

```
Confusion Matrix (Accuracy 0.8405)

       Prediction
Actual    0    1
     0 1366  120
     1  199  315
```

E. Summary

The training model's classification tree achieved an accuracy rate of 0.8330, or 83**%**; the test model achieved an accuracy rate of 0.8405, or 84**%**. Both models' accuracy was calculated by adding the TP and TN and then dividing the total number of records (TP + TN + FP + FN).

E2. Classification Analysis Results and Implications

Using tenure and monthly charge as predictor variables, the test model predicted our target class with an 84% accuracy rate; it should also be noted that the predicted class has a 16% chance of being incorrect.

E3. Analysis Limitation

One limitation of this analysis is that a small change within the dataset can cause variance and make the tree structure unstable; for instance, increasing or decreasing a customer’s monthly charge can affect the outcome.

E4. Recommended course of Action

I would recommend that the company focus on those individuals where their monthly charge <= $175; those customers tend to churn more. They should look at contracts as well; those customers with a monthly charge <= $175 and with a contract <= 5 months will churn. Focusing on those customers meeting these parameters or criteria will allow the company to provide or create incentives for them to stay; customers with longer tenure tend not to churn – the ultimate goal is to retain as many customers as possible and to increase their tenure.

G. Panopto video recording

VideoLink

References

Decision Tree. Geeks for Geeks. (2022, July 20). Retrieved August 7, 2022, from https://www.geeksforgeeks.org/decision-tree/

Misra, S., Li, H., & He, J. (2020). *Data preprocessing*. Data Preprocessing - an overview | ScienceDirect Topics. Retrieved August 7, 2022, from https://www.sciencedirect.com/topics/engineering/data-preprocessing

WGU. (n.d.). NVM2 TASK 2: *Predictive Analysis*. WGU Performance Assessment. Retrieved July 23, 2022, from https://tasks.wgu.edu/student/000194226/course/20900018/task/2807/overview