D209: Data Mining I Task 2

Justin Huynh

Student ID: 012229514

M.S. Data Analytics

A1. Proposal of Question

My research question for this project is “How can we predict whether a customer will churn based on their attributes using decision trees?” We will be using decision trees to answer our question.

A2. Defined Goal

The primary goal of this analysis is to develop a decision tree model that can accurately predict whether a customer will churn or not based on their demographic, service usage, account information, etc. This model will help an organization identify key contributing factors to customer churn and enable them to proactively take action to improve retention rate.

B1: Explanation of Prediction

The decision trees analyze the data set by recursively splitting the data into subsets based on the most significant features that best separates the data according to the target variable ‘Churn’. At each split, the algorithm selects the feature that results in the most significant information gain or the greatest reduction in impurity (e.g., using Gini impurity or entropy). The process of splitting continues until an established threshold is met, such as reaching a maximum tree depth or having a minimum number of samples per leaf node. Finally, once the tree is built, predictions for new data points are made by traversing the tree from the root to a leaf node based on the feature values of the new data points.

The expected outcome should be that customers are separated into ‘churn’ and ‘not churn’ categories based on their attributes (demographic, usage, billing info, etc.) and the model will provide insights on which features contribute the most significantly to

customer churn. We can visualize the decision tree to understand the decision rules and the hierarchy of feature importance.

B2: Summary of Method Assumption

One assumption of decision trees is the assumption of feature independence. The model assumes the features are independent of one another and that each feature contributes independently to the prediction of ‘Churn’. This means that the algorithm does not account for any interactions or correlations between features when making splits. This assumption is often violated in practice, but decision trees still tend to perform well despite the assumption not holding up perfectly.

B3: Packages or Libraries List

For this analysis, we will be using Python and its libraries and packages. We will be using pandas for data manipulation and analysis due to its ability to provide structures like dataframes that help simplify cleaning, encoding, and splitting data sets. We’ll also be using numpy due to its ability to perform numerical operations and array manipulations. We’ll also use scikit-learn since this machine learning library can implement model training, evaluation, and validation. Lastly, we’ll use matplotlib and seaborn for creating visualizations, which will help us visualize data distributions, relationships, and evaluation metrics to better understand the model performance.

C1: Data Processing

The primary data preprocessing goal is to transform the raw data set into a format suitable for training a decision tree model. This involves handling missing values, encoding categorical variables, and scaling numerical variables where necessary.

These steps ensure that the model can effectively learn from the data and make accurate predictions regarding customer churn.

C2: Data Set Variables

These are the identified variables for our analysis:

- Numeric Variables
    - ‘Age’
    - ‘Bandwidth_GB_Year’
    - ‘Children’
    - ‘Contacts’
    - ‘Income’
    - ‘MonthlyCharge’
    - ‘Outage_sec_perweek’
    - ‘Tenure’
    - ‘Yearly_equip_failure’
- Categorical Variables
    - ‘Contract’
    - ‘DeviceProtection’
    - ‘Gender’
    - ‘InternetService’
    - ‘Job’
    - ‘Marital’
    - ‘Multiple’
    - ‘OnlineBackup’

- ‘OnlineSecurity’
- ‘PaperlessBilling’
- ‘PaymentMethod’
- ‘Phone’
- ‘StreamingMovies’
- ‘StreamingTV’
- ‘TechSupport’
- ‘Techie’
- ‘Churn’ (target variable)

C3: Steps for Analysis

For our first step, we’ve loaded the ‘churn_clean.csv’ file into our notebook so we can display the data using pandas to verify the data loaded correctly. Next, we select only the relevant categorical and numeric variables needed for our analysis after we check if there are any missing values. The categorical variables will be encoded using One Hot Encoding so they can be fitted and transformed for the model. The numerical variables will be initialized with StandardScaler so they can also be fitted and transformed. We then split the data into training and testing sets and ensure all are numeric before we initialize the Decision Trees to train the model. After verifying the training and testing sets, our results will be saved into a csv file.

See code attached in WGU_D209_Task_2.ipynb.

C4: Cleaned Data Set

A copy of ‘cleaned_churn_data_d209_task2.csv' will be submitted with this task.

D1: Splitting the Data

```
# save the processed training and testing data to CSV files
X_train.to_csv('X_train_task_2.csv', index=False)
X_test.to_csv('X_test_task_2.csv', index=False)
y_train.to_csv('y_train_task_2.csv', index=False)
y_test.to_csv('y_test_task_2.csv', index=False)

# display training and testing sets shapes
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(8000, 671) (2000, 671) (8000,) (2000,)
```

Copies of the training and testing sets will be submitted in csv files with this task.

See code attached in WGU_D209_Task_2.ipynb.

D2: Output and Intermediate Calculations

For the data analysis, a Decision Tree Classifier was used to predict customer churn based on various features. We started with data preprocessing, where non-relevant columns were removed, missing values were checked, and the target variable 'Churn' was converted to numeric values to fit the model. Categorical variables were encoded using OneHotEncoder, and numeric variables were scaled using StandardScaler. The dataset was then split into training (80%) and testing (20%) sets. The Decision Tree model was trained on the training set and evaluated on the test set. We're also using a classification report to look at precision, recall, and F1-score for both churn (1) and non-churn (0).

```
# remove non-relevant columns
non_relevant_columns = ['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
                        'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone',
                        'Port_modem', 'Tablet', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5',
                        'Item6', 'Item7', 'Item8']

df = df.drop(columns=non_relevant_columns)

# verify the remaining columns
print(df.columns)
```

```
# check for missing values in the relevant columns
relevant_columns = df.columns
missing_values = df[relevant_columns].isnull().sum()
print(missing_values[missing_values > 0])

# convert 'Churn' to numeric values
df['Churn'] = df['Churn'].map({'Yes': 1, 'No': 0})
```

```
# list of categorical variables
categorical_columns = ['Contract', 'DeviceProtection', 'Gender', 'InternetService', 'Job',
                       'Marital', 'Multiple', 'OnlineBackup', 'OnlineSecurity', 'PaperlessBilling',
                       'PaymentMethod', 'Phone', 'StreamingMovies', 'StreamingTV', 'TechSupport', 'Techie']

# initialize the OneHotEncoder
encoder = OneHotEncoder(sparse_output=False, drop='first')

# fit and transform the categorical columns
encoded_categorical_data = encoder.fit_transform(df[categorical_columns])

# create a DataFrame with the encoded categorical data
encoded_categorical_df = pd.DataFrame(encoded_categorical_data, columns=encoder.get_feature_names_out(categorical_co

# drop the original categorical columns from the data set
df = df.drop(columns=categorical_columns)

# add the encoded categorical columns to the data set
df = pd.concat([df, encoded_categorical_df], axis=1)

# verify the columns in the feature set
print(df.columns)
```

```
# list of numeric columns
numeric_columns = ['Age', 'Bandwidth_GB_Year', 'Children', 'Contacts', 'Income',
                   'MonthlyCharge', 'Outage_sec_perweek', 'Tenure', 'Yearly_equip_failure']

# initialize the StandardScaler
scaler = StandardScaler()

# fit and transform the numeric columns
df[numeric_columns] = scaler.fit_transform(df[numeric_columns])

# verify the scaled feature set
print(df.head())
```

```
# define the feature set (X) and the target variable (y)
X = df.drop(columns=['Churn'])
y = df['Churn']

# split the data into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# save the processed training and testing data to CSV files
X_train.to_csv('X_train_task_2.csv', index=False)
X_test.to_csv('X_test_task_2.csv', index=False)
y_train.to_csv('y_train_task_2.csv', index=False)
y_test.to_csv('y_test_task_2.csv', index=False)

# display training and testing sets shapes
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
# load the processed training and testing data
X_train = pd.read_csv('X_train_task_2.csv')
X_test = pd.read_csv('X_test_task_2.csv')
y_train = pd.read_csv('y_train_task_2.csv')
y_test = pd.read_csv('y_test_task_2.csv')

# ensure all columns are numeric
for column in X_train.columns:
    if X_train[column].dtype == object:
        print(f"Column {column} has string values:")
        print(X_train[column].unique())
        # Convert column to numeric, if possible
        X_train[column] = X_train[column].apply(lambda x: 1 if x.strip().lower() == 'yes' else (0 if x.strip().lower

for column in X_test.columns:
    if X_test[column].dtype == object:
        print(f"Column {column} has string values:")
        print(X_test[column].unique())
        # Convert column to numeric, if possible
        X_test[column] = X_test[column].apply(lambda x: 1 if x.strip().lower() == 'yes' else (0 if x.strip().lower()

# verify the conversion
print(X_train.dtypes)
print(X_test.dtypes)
```

```
# ensure target variable 'y' is in correct format
y_train = y_train.values.ravel()  # convert DataFrame to 1D array
y_test = y_test.values.ravel()    # convert DataFrame to 1D array

# initialize the Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=42)

# train the model
clf.fit(X_train, y_train)

# make predictions on the test set
y_pred = clf.predict(X_test)

# calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# generate classification report
report = classification_report(y_test, y_pred)
print(report)

# generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

See code attached in WGU_D209_Task_2.ipynb.

D3: Code Execution

See code attached in WGU_D209_Task_2.ipynb.

E1: Accuracy and MSE

The Decision Tree Classifier used for predicting customer churn resulted in 86.95% accuracy. Accuracy measures the correct predictions out of the total predictions made. The Mean Squared Error (MSE) of the model was calculated by comparing the predicted probabilities of churn to the actual binary outcomes. The MSE value of 0.1305 quantifies the average squared difference between the predicted probabilities and the actual outcomes. A lower MSE value means that the predicted probabilities are most likely close to the actual outcomes, which confirms the model's reliability in identifying

customers who are at high risk of churn. Both accuracy and MSE demonstrate the model's strong predictive performance.

See code attached in WGU_D209_Task_2.ipynb.

E2: Results and Implications

The model's accuracy of 86.95% indicates that it correctly predicts customer churn in the majority of cases. The MSE value of 0.1305 provides additional insight into the model's performance, indicating that the predicted probabilities are close to the actual binary outcomes. These results imply that the model is effective for creating customer retention strategies, which allows an organization to focus on identifying and retaining at-risk customers.

The precision, recall, and F1-score metrics further illustrate the model's effectiveness, with high precision for identifying customers who will not churn, and a balance between precision and recall for those who will churn. For class 0 (non churn), the precision is 0.91, meaning that 91% of the customers predicted not to churn actually did not churn. The recall for class 0 is 0.92, indicating that 92% of the actual non-churners were correctly identified by the model. The F1-score, which balances precision and recall, is also 0.91 for class 0. For class 1 (churn), the precision is 0.77, meaning that 77% of the customers predicted to churn actually churned. The recall for class 1 is 0.75, indicating that 75% of the actual churners were correctly identified. The F1-score for class 1 is 0.76, showing a reasonable balance between precision and recall. The macro average F1-score is 0.83, and the weighted average F1-score is 0.87, reflecting the model's overall balanced performance across both classes.

The confusion matrix also provided additional insight into the model's performance:

- 1333 (True Negatives): Customers correctly predicted not to churn.
- 123 (False Positives): Customers incorrectly predicted to churn.
- 138 (False Negatives): Customers incorrectly predicted not to churn.
- 406 (True Positives): Customers correctly predicted to churn.

The high number of true negatives and true positives shows that the model effectively distinguishes between customers who will churn and who will not. The relatively lower number of false positives and false negatives shows the model's reliability in its predictions.

See code attached in WGU_D209_Task_2.ipynb.

E3: Limitation

One limitation of the data analysis is the potential for overfitting with the Decision Tree Classifier. Decision trees can create complex models that fit the training data very closely but may not generalize very well to unseen data. This could result in decreased performance on new customer data that the model has not seen before.
Cross-validation and pruning techniques can help mitigate this issue, but they were not applied in this analysis. Also, while MSE provides useful information, it is not a standard measure for classification problems and may not fully capture the nuances of classification performance.

E4: Course of Action

A course of action an organization can take based on the results is implementing targeted retention campaigns by using the model’s performance to identify customers

who are at high risk of churning and create campaigns specifically to retain these customers by offering discounts, personalized promos, enhancing customer support etc. The organization should also continue to refine and validate the prediction model, potentially incorporating other techniques like cross-validation and model pruning to ensure robust performance and improve generalizing to new data.

F: Panopto Video

The URL link will also be submitted in the Performance Assessment task submission.

G: Sources of Third Party Code

1. “Using StandardScaler() Function to Standardize Python Data” Retrieved from https://www.digitalocean.com/community/tutorials/standardscaler-function-in-python
2. “1.10.Decision Trees” Retrieved from https://scikit-learn.org/stable/modules/tree.html

H: Web Sources

1. “Confusion Matrix in Machine Learning.” Retrieved from https://www.geeksforgeeks.org/confusion-matrix-machine-learning/
2. “Python | Mean Squared Error” Retrieved from https://www.geeksforgeeks.org/python-mean-squared-error/
3. “StandardScaler” Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html