**Research Question**

Can we identify which customer characteristics are the most significant predictors of churn using the Decision Tree Classifier?

**Data Analysis Goal**

Our goal is to identify the key customer characteristics (such as bandwidth usage, tenure, services, and customer interaction history) that are most strongly associated with churn, using a decision tree model. The aim is to provide actionable insights that can help reduce churn by targeting at-risk customers with tailored retention strategies.

**Method Justification**

Our decision tree model works by recursively splitting the dataset into branches based on features (such as customer demographics or service usage patterns) that provide the most significant distinction between churned and non-churned customers. At each node, the model selects the feature that best reduces a metric, aiming to create pure subsets where most customers either churn or don’t.

For example, the decision tree might first split the data based on “age” or “bandwidth usage,” identifying thresholds that best separate churners from non-churners. The tree will continue splitting the data into smaller branches until it either reaches the maximum depth or the nodes are pure (all customers in the branch either churn or don’t). The outcome is a set of rules that can be used to predict churn for new customers.

The expected outcome is a visual representation of the decision tree showing how different factors contribute to churn, clear identification of the most important features driving churn, and a set of interpretable rules that can be applied to target at-risk customers.

One key assumption of the decision tree model is the relationships between input features and the target variable (churn) can be captured through recursive binary splits. The model assumes that each feature contributes to the target (churn) in a way that can be expressed through a series of simple decision rules. It also assumes that these splits can adequately separate the classes (churn vs. non-churn) without needing complex transformations of the data.

Our packages we will be using are Scikit-learn, because of its robust machine learning libraries in Python that offers tools for building and evaluating our decision tree model. Pandas for its data manipulation and cleaning functions that allow use to preprocess the data by filtering, sorting, and transforming variables, which is necessary for preparing the data for the decision tree model. NumPy, for its efficient array handling and mathematical functions that work well with Pandas and Scikit-learn. It will be used for numerical computations and handling large matrices of features and target values during model building. Matplotlib/Seaborn will be used for data visualization, helping to create plots that represent feature importance, churn distribution, and decision tree structures. The visualizations can provide deeper insights into customer behavior patterns. These libraries collectively support our analysis by enabling efficient data preparation, building and evaluating the decision tree model, and visually interpreting the results to understand churn-driving factors.

**Data Preparation**

Our goal in data preprocessing for the decision tree model is to handle missing values and ensure that both numeric and categorical features are properly formatted. This is crucial because decision trees can handle both types of data but require that they are clean and properly represented. Handling missing values, outliers, and duplicates is important because they could skew the splits and decrease the accuracy of our model which influences our analysis. Our categorical variables also need to be encoded properly to be used in the splits.

The objective is to enhance the quality of the dataset to enhance the quality of our decision tree model. The initial data set variables that we will use to perform the analysis for our predictions are our numeric variables MonthlyCharge(float64), Bandwidth\_GB\_Year(float64), Age(int64), Tenure(float64), and our categorical variables Item1(string), Item2(string), Item3(string), Item4(string), Item5(string), Item6(string), Item7(string), Item8(string), Contract(string), InternetService(string), OnlineSecurity(string), and PaymentMethod(string). We bring in our data with the pd.read\_csv() to read in our data file and pd.DataFrame(df) turns the data into a data frame for us, df.head() gives us a quick view of the top five rows to make sure our data was pulled in correctly.

The df.info() allows us to get insight into the data, showing the index, column names, non-null count, and data types for each feature. The df.describe() allows us to see the statistical descriptions of our dataset, df.dtypes shows us the data type of each feature in the dataset. We then move on to checking for null values using df.isnull().sum() to get a count of missing values for each feature column. We have no missing values to account for, so we move on to checking for duplicates with df.duplicated().sum().

We will be using customers service data and customer satisfaction data in our model so to improve understanding; satisfaction survey column names were changed using df.rename(). Instead of using Item1- Item8 as column names, each item has been updated to represent what the survey question refers to. We will be using the satisfaction variables as categories based on customer responses, so we convert the columns to strings using df[column].astype(dtype). We then move into addressing outliers in the data. We are going to replace numerical outliers with the median value of each column, and we used the quartile range lower and upper bounds to identify outliers df[column] = np.where((df[column] < lower\_bound) | (df[column] > upper\_bound), median\_value, df[column]). Finally, we encode our categorical variables

Set Categorical and Numerical Variables

categorical\_columns = ['Contract', 'InternetService', 'TimelyResponse', 'TimelyFixes',

'TimelyReplacements', 'Reliability', 'Options', 'RespectfulResponse', 'CourteousExchange', 'EvidenceOfActiveListening','OnlineSecurity','PaymentMethod']

numerical\_columns = ['MonthlyCharge', 'Bandwidth\_GB\_Year', 'Age', 'Tenure']

Separate the features and target

X = df[numerical\_columns + categorical\_columns]

y = df['Churn']

One-Hot Encode the Categorical Columns

encoder = OneHotEncoder(sparse=False, handle\_unknown='ignore')

X\_categorical\_encoded = encoder.fit\_transform(X[categorical\_columns]).

Label Encode the Target Variable

label\_encoder = LabelEncoder()

y\_encoded = label\_encoder.fit\_transform(y).

Now we have a cleaned dataset with no missing values, no duplicates, addressed outliers, and encoded values ready for modeling.

**Analysis**

We want to ensure that the decision tree model generalizes well to new data, so the dataset needs to be split into training and test sets. The training set is used to build the model, while the test set is used to evaluate its performance on unseen data. The decision tree algorithm is used to analyze the relationship between customer characteristics and churn. The model is built using the training data, and the key technique is the recursive splitting of the dataset based on the feature that best separates churned from non-churned customers. The Gini impurity criterion is used at each node to determine the optimal split. The final model provides a tree structure where each leaf node represents a class (churn or no churn), and intermediate nodes represent decision points based on customer characteristics. The Gini impurity algorithm calculates how “pure” each split is, and the decision tree will rank features based on how much they contribute to the splits. Our expected outcomes are identifying the most significant factors that contribute to churn within our dataset, provide insight into thresholds or patterns that are influencing churn and ultimately help the organization predict which customers are most at risk of churning, allowing for targeted interventions to improve retention.

A screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

A number of numbers on a white background

Description automatically generated

A diagram of a company structure

Description automatically generated

**Data Summary and Implications**

**Significant Predictors**

* **MonthlyCharge:** Higher charges increase the likelihood of churn.
* **Contract (Month-to-month):** Customers with month-to-month contracts are more likely to churn.

 **Tenure:** Customers with shorter tenure are at higher risk of churning.

 **InternetService (Fiber Optic):** Customers with fiber optic service are less likely to churn.

 **Bandwidth\_GB\_Year:** Higher bandwidth usage is associated with lower churn.

 **Options:** Having more options or features available to customers might reduce churn.

The accuracy of the model is 86.85%, which means that approximately 87 out of every 100 predictions made by the decision tree model are correct. In the context of customer churn, this indicates that the model correctly classifies customers who churn or don't churn most of the time. To interpret mean squared error(MSE) for our decision tree model, we use the misclassification rate, which would be (1 - accuracy). In this case, the error rate would be approximately 13.15%, meaning the model misclassifies a little over 13% of customers.

The model has high precision for the not churned class (91%) and a slightly lower precision for the churn class (76%). Precision refers to how many of the customers predicted as churners, churned. Recall for the churn class (76%) is lower than for the not churned class (91%). This indicates that while the model performs well in detecting non-churners, it struggles a bit more to identify all churners. Many customers who do churn are missed; they are predicted to not churn but do. The F1-score for the churn class (76%) suggests that there is some imbalance in how well the model identifies churners compared to non-churners. The weighted average F1-score of 87% suggests the model balances precision and recall well for the overall dataset, but there's room for improvement in churn prediction.

The results show that the model is effective at predicting non-churners, which is useful in ensuring that resources are not wasted on retaining customers who are unlikely to churn. However, the model is not as effective at predicting churners, which may result in missed opportunities to intervene with at-risk customers. This suggests that we should further refine the model or implement additional strategies to identify churners more effectively.

**Analysis Limitations**

One major limitation of our data analysis is the class imbalance between churners and non-churners. This imbalance can lead the model to predict the majority class (non-churn) more frequently, which results in a high accuracy score but a poor ability to detect actual churners.

In this case relying solely on accuracy as a metric can be misleading because the model might perform well on the majority class but fail to adequately identify customers who are at risk of churning. Based on the accuracy, the model might be simply classifying almost all customers as non-churners, which isn't useful for predicting the smaller subset of churners.

The ROC curve and the associated AUC (Area Under the Curve) metric address this limitation by evaluating the model's ability to distinguish between the two classes (churners and non-churners) across different threshold values. Instead of focusing on accuracy, the ROC curve helps measure how well the model balances between correctly identifying positive cases (churners) and avoiding false positives (incorrectly predicting churn for non-churners).

The True Positive Rate (Recall) measures how many actual churners were correctly identified by the model and the False Positive Rate shows how often the model incorrectly classifies non-churners as churners. Plotting these two metrics across various decision thresholds, the ROC curve provides a more nuanced view of the model's performance, especially in the presence of class imbalance. A high AUC score, of 0.94 in this case, indicates that the model is very effective at distinguishing between churners and non-churners, even when the class distribution is imbalanced.

A graph of a curve

Description automatically generated with medium confidence

**Recommended Course of Action**

Focus on ‘Likely Churners’, since the model can accurately identify customers who are likely to churn, the organization should proactively engage with these at-risk customers. This could include offering discounts, upgraded services, or loyalty rewards to high-risk customers to incentivize them to stay. Customers identified as potential churners, should be sent feedback forms to understand their pain points and exit surveys should be sent to churned customers to gain insight on why they decided to churn.

**References**

W3schools.com accessed, 20 October 2024, https://www.w3schools.com/python/pandas/default.asp

W3schools.com accessed, 20 October 2024, https://www.w3schools.com/python/numpy/numpy\_array\_iterating.asp.

W3schools.com accessed, 20 October 2024, https://www.w3schools.com/python/matplotlib\_intro.asp

Seaborn.pydata.org accessed, 20 October 2024, https://seaborn.pydata.org/tutorial/function\_overview.html

Scikit-learn.org accessed, 20 October 2024, https://scikit-learn.org/stable/modules/impute.html

Scikit-learn.org accessed, 20 October 2024, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

Scikit-learn.org accessed, 20 October 2024, https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_split.html#sklearn.model\_selection.train\_test\_split

Scikit-learn.org accessed, 20 October 2024, https://scikit-learn.org/stable/api/sklearn.metrics.html#module-sklearn.metrics

Scikit-learn.org accessed 20 October 2024, https://scikit-learn.org/1.5/modules/tree.html

Scikit-learn.org accessed 20 October 2024, https://scikit-learn.org/stable/modules/generated/sklearn.feature\_selection.chi2.html

docs.scipy.org accessed 20 October 2024, https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest\_ind.html#ttest-ind

Scikit-learn.org accessed 20 October 2024, https://scikit-learn.org/dev/modules/generated/sklearn.metrics.auc.html