D209 Task 2: Predictive Analysis

Part I: Research Question

A1. <u>Research Question</u>

Is it possible to predict what predictor variables are at the highest risk of churn using the decision tree method? If so, what are those variables?

A2. <u>Data Analysis Goal</u>

The goal of this data analysis is to develop a decision tree machine learning model to gain more insight to help the company predict which variables from the churn dataset are at the highest risk of customer churn.

Part II: Method Justification

B1. <u>Prediction Method Justification</u>

The decision tree is a machine learning algorithm is used for predictive modeling as well as decision-making. It consists of a tree-like flowchart to show the break down of a data set into smaller groups with similarities. Each group begins at the root, has internal nodes to represent the attributes, branches from the nodes to represent the test outcomes, and the last node is a leaf to represent the label of a class.

The expected outcome is that the decision tree can help in deciding what actions to take by visually mapping out the pros, cons, and costs of every possible option (Cohen, 2021).

B2. <u>Method Assumption Summary</u>

An assumption of the decision tree method is that it is non-parametric. This means there is not a specific distribution, making the model much simpler and more user-friendly (Navlani, 2018).

B3. <u>Packages & Libraries Used</u>

I will be using Python to perform this data analysis because of the consistent syntax that makes it easy to learn and follow along, the flexibility to create and learn new things, and all of the libraries and packages that it has to offer. For example, I will be using the following libraries and packages for my analysis (R or Python 2023):

- pandas- to load datasets
- NumPy- to work with arrays
- Sci-kit Learn- for machine learning and to transform our data
- Matplotlib- for basic plotting generally consisting of bars, lines, pies, scatter plots, and graphs
- Seaborn- for a variety of visualization patterns

Part III: Data Preparation

C1. Data Preprocessing Goal

A data preprocessing goal is to remove any outliers, impute any missing data, and one-hot encoding by re-expressing the binary categorical variables (yes/no) as dummy variables where 1 represents yes and 0 represents no.

C2. Dataset Variables

Variable Name	Data Class	Data Type
Children	Quantitative	Numeric
Income	Quantitative	Numeric
Churn	Qualitative	Categorical
Outage_sec_perweek	Quantitative	Numeric
Email	Quantitative	Numeric
Contacts	Quantitative	Numeric
Yearly_equip_failure	Quantitative	Numeric
Techie	Qualitative	Categorical
Contract	Qualitative	Categorical
Port_modem	Qualitative	Categorical
Tablet	Qualitative	Categorical
InternetService	Qualitative	Categorical
Phone	Qualitative	Categorical
Multiple	Qualitative	Categorical
OnlineSecurity	Qualitative	Categorical
OnlineBackup	Qualitative	Categorical
DeviceProtection	Qualitative	Categorical
TechSupport	Qualitative	Categorical
StreamingTV	Qualitative	Categorical
StreamingMovies	Qualitative	Categorical
Tenure	Quantitative	Numeric
MonthlyCharge	Quantitative	Numeric
Bandwidth_GB_Year	Quantitative	Numeric
Item1	Quantitative	Numeric
Item2	Quantitative	Numeric
Item3	Quantitative	Numeric
Item4	Quantitative	Numeric
Item5	Quantitative	Numeric
Item6	Quantitative	Numeric
Item7	Quantitative	Numeric
Item8	Quantitative	Numeric

C3. Analysis Steps

1. Import any necessary libraries and packages.
2. Load dataset into pandas data frame using read_csv command. The data frame is named “df”.
3. Explore the dataset in order to determine how to evaluate the input data by using the head() command to print the first 5 rows of the dataset.
4. Rename the survey columns to describe the variables better.
5. Print column names to check corrections made.
6. Calculate the total null values and total duplicate values in the dataset. If there are not any, the values will be shown as 0.
7. Drop columns that are unnecessary for the analysis.
8. Use the head() command to look at what data is left.
9. Find the summary statistics for the variables used in the analysis.
10. Create the univariate visualizations for all the predicting variables and the target variable.
11. Drop either Tenure or Bandwidth_GB_Year because of multicollinearity from a previous analysis.
12. Reformat the columns to have 3 decimal places.
13. Create dummy variables where "Yes" is represented by 1 & "No" is represented by 0.
14. Drop the original categorical columns so only dummy categorical columns are left.
15. Check the new data frame to make sure all the data transferred correctly.
16. Extract the cleaned dataset.

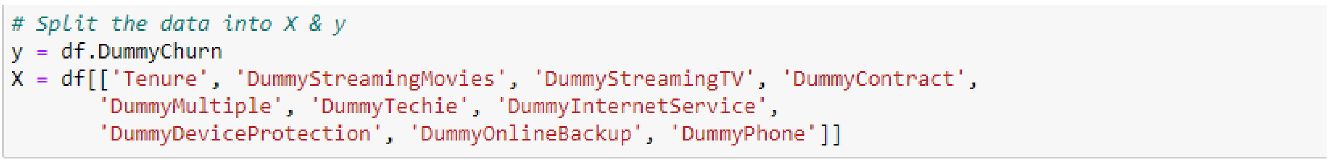
C4. Cleaned Dataset

The cleaned dataset will be attached to submission as “churn2_clean.csv”.

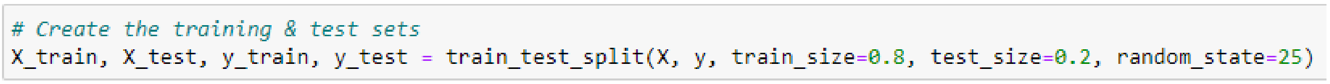
Part IV: Analysis

D1. Splitting the Data

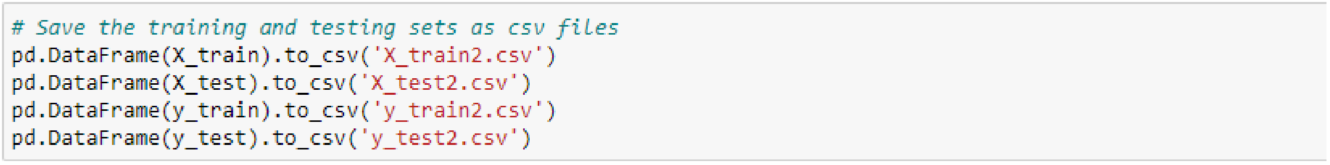
The csv files of the split data into training and testing sets will be attached to the submission as “X_train2.csv”, “X_test2.csv”, “y_train2.csv”, and “y_test2.csv”.



```
# Split the data into X & y
y = df.DummyChurn
X = df[['Tenure', 'DummyStreamingMovies', 'DummyStreamingTV', 'DummyContract',
       'DummyMultiple', 'DummyTechie', 'DummyInternetService',
       'DummyDeviceProtection', 'DummyOnlineBackup', 'DummyPhone']]
```



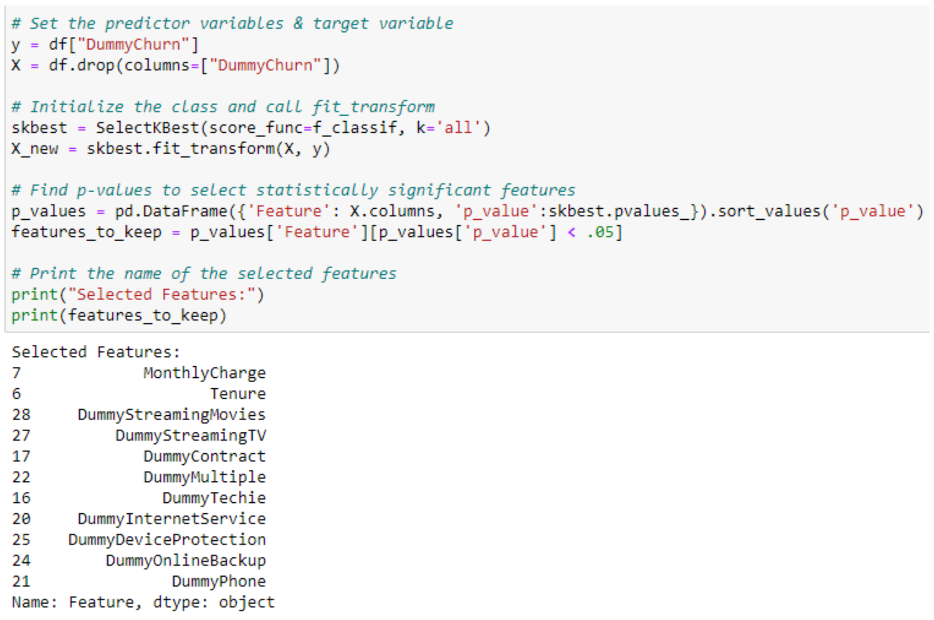
```
# Create the training & test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=25)
```



```
# Save the training and testing sets as csv files
pd.DataFrame(X_train).to_csv('X_train2.csv')
pd.DataFrame(X_test).to_csv('X_test2.csv')
pd.DataFrame(y_train).to_csv('y_train2.csv')
pd.DataFrame(y_test).to_csv('y_test2.csv')
```

D2. Output & Intermediate Calculations

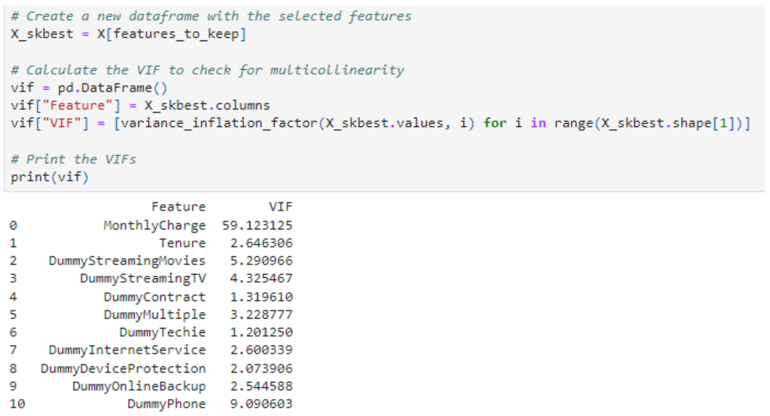
First, I had to find the variables that are most useful in predicting the target variable by using SelectKBest from scikit-learn to select the features with a significant p-value and build an initial model.



```python
# Set the predictor variables & target variable
y = df["DummyChurn"]
X = df.drop(columns=["DummyChurn"])

# Initialize the class and call fit_transform
skbest = SelectKBest(score_func=f_classif, k='all')
X_new = skbest.fit_transform(X, y)

# Find p-values to select statistically significant features
p_values = pd.DataFrame({'Feature': X.columns, 'p_value':skbest.pvalues_}).sort_values('p_value')
features_to_keep = p_values['Feature'][p_values['p_value'] < .05]

# Print the name of the selected features
print("Selected Features:")
print(features_to_keep)
```

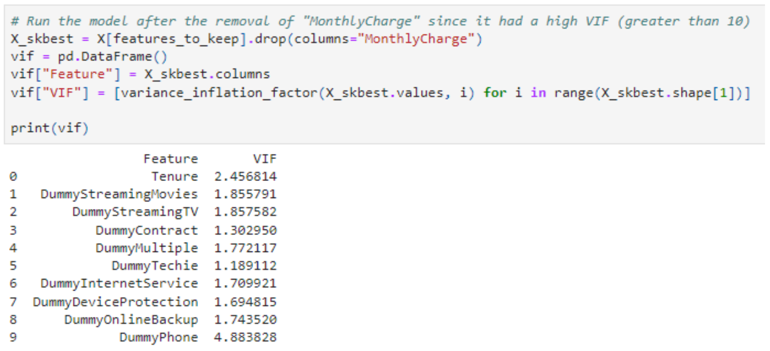
```
Selected Features:
7              MonthlyCharge
6                     Tenure
28      DummyStreamingMovies
27          DummyStreamingTV
17             DummyContract
22             DummyMultiple
16               DummyTechie
20      DummyInternetService
25     DummyDeviceProtection
24         DummyOnlineBackup
21                DummyPhone
Name: Feature, dtype: object
```

The next step is to calculate the variance inflation factors of the selected features to check for multicollinearity. If any of the VIFs are greater than 10, it will be removed from the analysis. In this case, MonthlyCharge had a much greater VIF, so it was removed. The model was run again to make sure the VIFs were less than 10.



```python
# Create a new dataframe with the selected features
X_skbest = X[features_to_keep]

# Calculate the VIF to check for multicollinearity
vif = pd.DataFrame()
vif["Feature"] = X_skbest.columns
vif["VIF"] = [variance_inflation_factor(X_skbest.values, i) for i in range(X_skbest.shape[1])]

# Print the VIFs
print(vif)
```

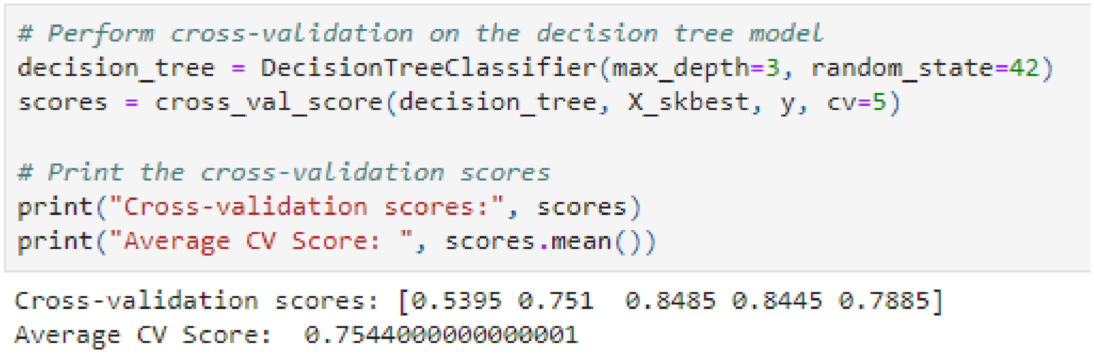
```
                  Feature        VIF
0           MonthlyCharge  59.123125
1                  Tenure   2.646306
2    DummyStreamingMovies   5.290966
3        DummyStreamingTV   4.325467
4           DummyContract   1.319610
5           DummyMultiple   3.228777
6             DummyTechie   1.201250
7    DummyInternetService   2.600339
8   DummyDeviceProtection   2.073906
9       DummyOnlineBackup   2.544588
10             DummyPhone   9.090603
```



```python
# Run the model after the removal of "MonthlyCharge" since it had a high VIF (greater than 10)
X_skbest = X[features_to_keep].drop(columns="MonthlyCharge")
vif = pd.DataFrame()
vif["Feature"] = X_skbest.columns
vif["VIF"] = [variance_inflation_factor(X_skbest.values, i) for i in range(X_skbest.shape[1])]

print(vif)
```

```
                 Feature       VIF
0                 Tenure  2.456814
1   DummyStreamingMovies  1.855791
2       DummyStreamingTV  1.857582
3          DummyContract  1.302950
4          DummyMultiple  1.772117
5            DummyTechie  1.189112
6   DummyInternetService  1.709921
7  DummyDeviceProtection  1.694815
8      DummyOnlineBackup  1.743520
9             DummyPhone  4.883828
```

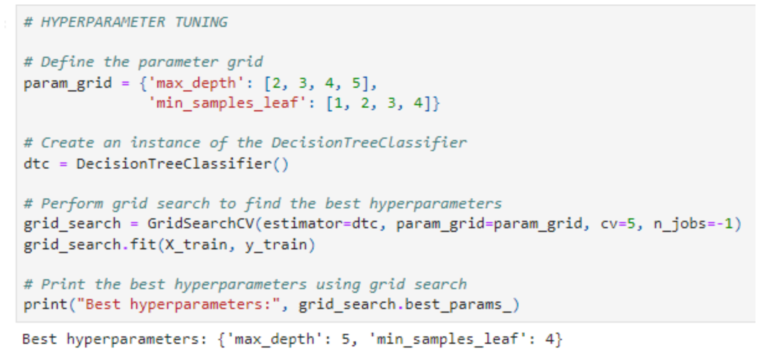
After the features to keep were determined, I split the data, created training and test sets, then performed cross-validation on the decision tree model and printed the scores after. Cross-validation is a method to give an idea if your model is underfitting or overfitting with the provided data. If the score were 0, this would mean there is misclassification and a score of 1 would represent a good fit model. Based on the scores from my model, there are 5 observations used to give an average cross-validation score of 0.7544, or 75.44%. This tells us that it is a decent fit model, but there is still room for improvement (Pramod, 2024).



```
# Perform cross-validation on the decision tree model
decision_tree = DecisionTreeClassifier(max_depth=3, random_state=42)
scores = cross_val_score(decision_tree, X_skbest, y, cv=5)

# Print the cross-validation scores
print("Cross-validation scores:", scores)
print("Average CV Score: ", scores.mean())
```

```
Cross-validation scores: [0.5395 0.751  0.8485 0.8445 0.7885]
Average CV Score:  0.7544000000000001
```

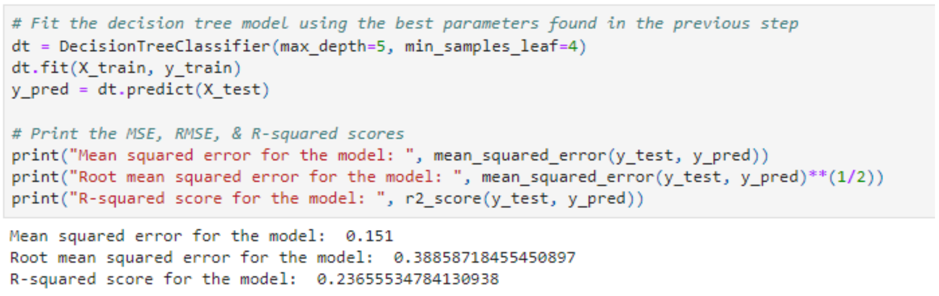
After that, I defined the parameter grid to search and decision tree classifier in preparation to perform grid search. To do so, I used the GridSearchCV() method to define the parameter range. This method is referred to as hyperparameter tuning where a model is built for every possible combination of the provided hyperparameters and the model with the best results is the one chosen (Jordan, 2018)



```
# HYPERPARAMETER TUNING

# Define the parameter grid
param_grid = {'max_depth': [2, 3, 4, 5],
              'min_samples_leaf': [1, 2, 3, 4]}

# Create an instance of the DecisionTreeClassifier
dtc = DecisionTreeClassifier()

# Perform grid search to find the best hyperparameters
grid_search = GridSearchCV(estimator=dtc, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)

# Print the best hyperparameters using grid search
print("Best hyperparameters:", grid_search.best_params_)
```

```
Best hyperparameters: {'max_depth': 5, 'min_samples_leaf': 4}
```

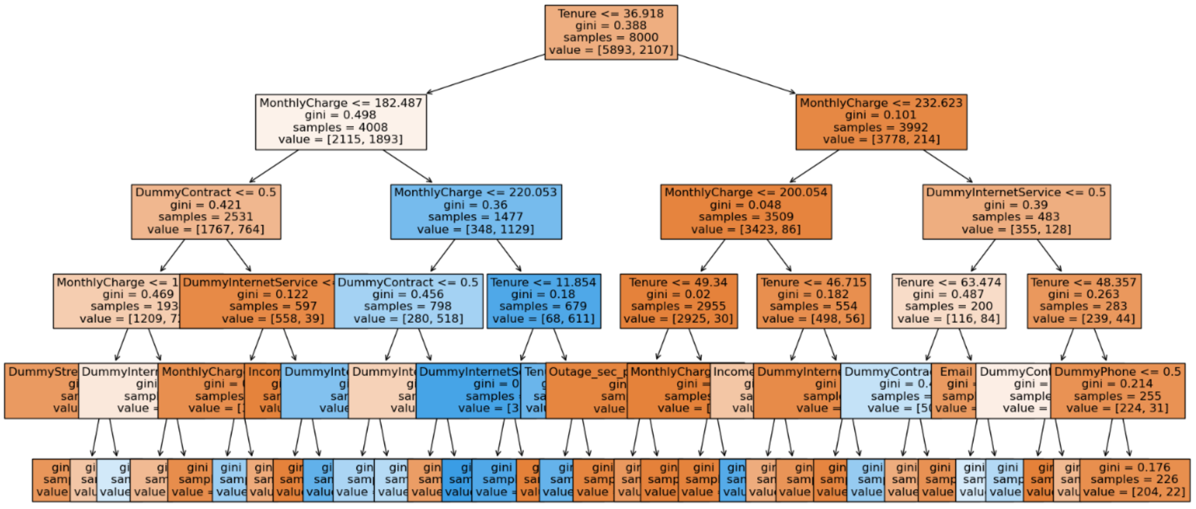
Then, the next step of the intermediate calculations is to fit the decision tree model using the best parameters found in the previous step. I also printed the mean squared error, root mean squared error, and the R-squared score for the model, but I will go into depth on what each of those values mean in part E1.



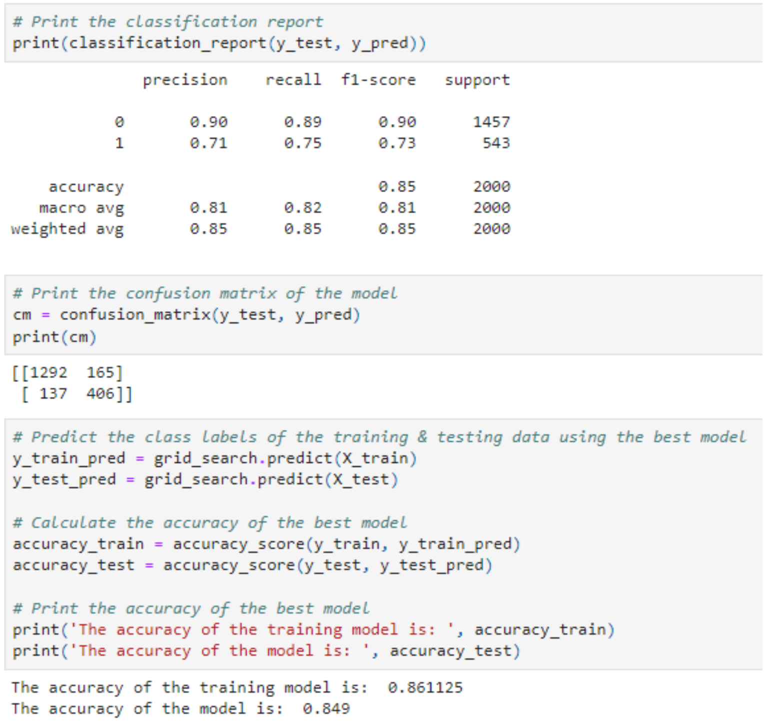
```
# Fit the decision tree model using the best parameters found in the previous step
dt = DecisionTreeClassifier(max_depth=5, min_samples_leaf=4)
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)

# Print the MSE, RMSE, & R-squared scores
print("Mean squared error for the model: ", mean_squared_error(y_test, y_pred))
print("Root mean squared error for the model: ", mean_squared_error(y_test, y_pred)**(1/2))
print("R-squared score for the model: ", r2_score(y_test, y_pred))
```

```
Mean squared error for the model:  0.151
Root mean squared error for the model:  0.38858718455450897
R-squared score for the model:  0.23655534784130938
```

Now that we have all of the calculations and data needed to create the decision tree model, the decision tree model can be created so there is a visualization that can be referred to. The decision tree model starts at the top with the root node, Tenure in this case, and is broken down into subsets. However, the main take away is the root node represents the best predictor.



After the decision tree model was fit and plotted, the classification report of the model is generated as well as a confusion matrix to summarize the performance of the classification model that has been created. As shown below, the confusion shows out of 2000 observations, 1698 were classified correctly while 302 were classified incorrectly. Hence why the accuracy of the model is 84.9%.



```
# Print the classification report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.89      0.90      1457
           1       0.71      0.75      0.73       543

    accuracy                           0.85      2000
   macro avg       0.81      0.82      0.81      2000
weighted avg       0.85      0.85      0.85      2000
```

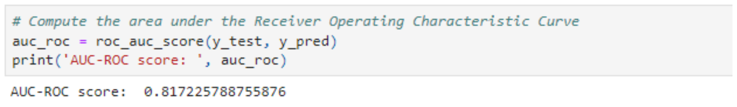
```
# Print the confusion matrix of the model
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[1292  165]
 [ 137  406]]
```

```
# Predict the class labels of the training & testing data using the best model
y_train_pred = grid_search.predict(X_train)
y_test_pred = grid_search.predict(X_test)

# Calculate the accuracy of the best model
accuracy_train = accuracy_score(y_train, y_train_pred)
accuracy_test = accuracy_score(y_test, y_test_pred)

# Print the accuracy of the best model
print('The accuracy of the training model is: ', accuracy_train)
print('The accuracy of the model is: ', accuracy_test)
```

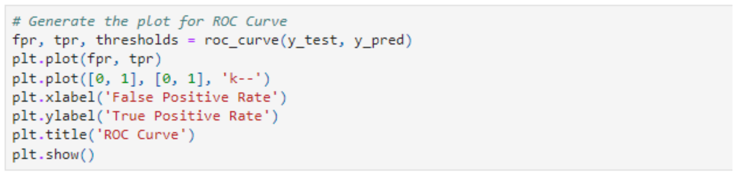
```
The accuracy of the training model is:  0.861125
The accuracy of the model is:  0.849
```

The last step is to compute the area under the Receiver Operating Characteristic (ROC) Curve and generate the plot for the ROC Curve. As shown below, the AUC-ROC score is about 0.82 which is considered a very good score since 1.0 is a perfect score. The black dashed line represents an AUC of 0.5 which is pretty much random (Brownlee, 2018).

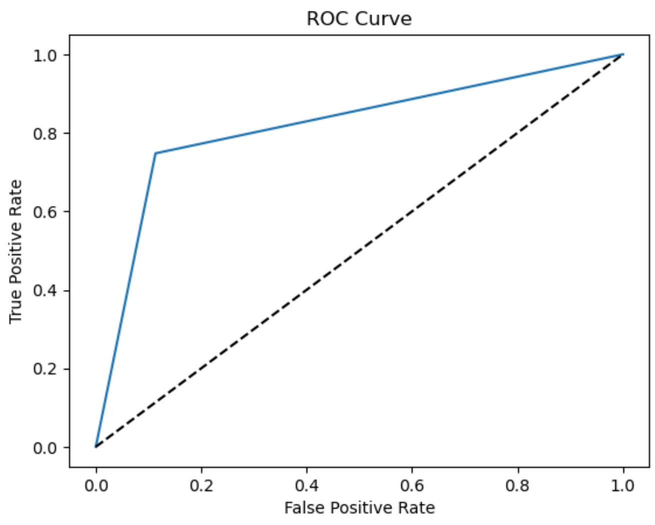


```
# Compute the area under the Receiver Operating Characteristic Curve
auc_roc = roc_auc_score(y_test, y_pred)
print('AUC-ROC score: ', auc_roc)
```

```
AUC-ROC score:  0.817225788755876
```



```
# Generate the plot for ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.plot(fpr, tpr)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
```



D3. <u>Code Execution</u>

All of the code from part D2 can be found in provided file "D209 Task 2.ipynb".

Part V: Data Summary & Implications

E1. <u>Accuracy & MSE</u>

The accuracy of the model is 0.849 which is 84.9%. Classification accuracy represents the ratio of the number of correct predictions out of all the predictions, meaning that the model created predicted the test data 84.9% correctly.

The mean squared error (MSE) for the model is 0.151 and the root mean squared error (RMSE) for the model is 0.389. The lower the MSE and RMSE values, the higher accuracy of the model. But these two metrics are typically used for regression models. Therefore, the accuracy for a classification model may be questionable (Chugh, 2024).

E2. **Results & Implications**

I used the decision tree model to predict which variables from the churn dataset are at the highest risk of customer churn. The features used in the model were 'Tenure', 'DummyStreamingMovies', 'DummyStreamingTV', 'DummyContract', 'DummyMultiple', 'DummyTechie', 'DummyInternetService', 'DummyDeviceProtection', 'DummyOnlineBackup', and 'DummyPhone'. I used the SelectKBest approach to choose which features are most important to use in the model. After that, I found the variance inflation factors of the selected features to check for multicollinearity. If the VIF was greater than 10, it was removed. After that, I created the classification report to provide the precision, recall, F-score, support, and accuracy of the model. The precision for predicting both the negative (0) and positive (1) class was fairly high with the precision of predicting negative class being 0.9 and the positive being 0.71. The recall is the ability of a classifier to identify the true positives in a class. The recall for class 0 was 0.89 and class 1 was 0.75. Therefore, class 0 was nearly perfect, and class 1 was quite a bit lower, but still greater than 0.5. If it were 0.5 or lower, this would mean there were many false negatives, resulting in an imbalanced class (Kohli, 2019). Overall, it is implied that all of the values in the classification report show that this model is a good model which is further proven by the accuracy of 0.849. However, it is difficult to evaluate the accuracy of the predicted output versus the calculated output in terms of the initial research question because the R-squared score of 0.237 is so low. Since it is so close to 0, that means the model does not have all of the variables that are needed for the outcome. Also, based on the visualization of the decision tree model, it may too complex and lead to overfitting. To improve an overfitting model, it may be helpful to perform cross-validation on multiple data subsets next time or pruning the decision tree by reducing the number of parameters.

E3. **Limitation**

One limitation of the decision tree model would be how they can be prone to overfitting. Sometimes, decision trees become too complex, making it hard to categorize new data.

E4. **Course of Action**

Based on this analysis results and the model's accuracy score, the company could consider this decision tree model to predict customer churn. However, based on the visualization of the decision tree model, I don't think this model is very reliable because it may too complex and lead to overfitting. To improve this overfitting model, it may be helpful to perform cross-validation on multiple data subsets next time or pruning the decision tree by reducing the number of parameters. Focusing on the topmost portion of the model, it seems customers who have stayed with the company longer who have a higher monthly charge are less likely to churn. The model also suggests that customers with a shorter contract are paying more monthly and most of the monthly charges are from their internet service. Based on results from previous courses, a course of action could be to persuade customers to get the fiber optic internet service because it seems that customers with that internet service rather than DSL or no internet service are more likely to stay with the company. This could help to retain customers because it would show them that you appreciate their loyalty and could save them money.

Part VI: Demonstration

F. Panopto Video

G. Third-Party Code Sources

N/A

H. Sources

Brownlee, J. (2018). How to Use ROC Curves and Precision-Recall Curves for Classification in Python. https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

Chugh, A. (2024, January 18). Mae, MSE, RMSE, coefficient of determination, adjusted R squared-which metric is better? Medium. https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e

Cohen, S. (2021). Decision tree algorithm. Decision Tree Algorithm . https://www.sciencedirect.com/topics/computer-science/decision-tree-algorithm#:~:text=A%20Decision%20Tree%20Algorithm%20is,classification%20tasks%20in%20computer%20science.

Jordan, J. (2018, December 5). Hyperparameter tuning for machine learning models. https://www.jeremyjordan.me/hyperparameter-tuning/

Kohli, S. (2019, November 18). Understanding a classification report for your machine learning model. Medium. https://medium.com/@kohlishivam5522/understanding-a-classification-report-for-your-machine-learning-model-88815e2ce397

Navlani, A. (2018). "Decision Tree Classification in Python Tutorial." Datacamp, www.datacamp.com/community/tutorials/decision-tree-classification-python.

Pramod, O. (2024, February 29). Cross validation. Medium. https://medium.com/@ompramod9921/cross-validation-623620ff84c2#:~:text=The%20array%20of%20scores%20represents,model%20performed%20in%20each%20iteration.

I. Professional Communication

Demonstrate professional communication in the content and presentation of your submission.