D209- Data Mining I Performance Assessment

Task 2: Predictive Analysis

Western Governs University

Table of Contents

Part I: Research Question

A1. Proposal of Question

What are the major predictor variables in determining or predicting the churn of customers from the churn dataset? This question will be answered using a decision tree.

A2. Goals

Customers can choose from multiple telecommunication services as their providers. Customer churn is the percentage of customers who discontinued their services from a provider within a specific time frame. The churn rate in the data set is documented by customers who discontinued their services within the last month. By examining the reasons behind customer churn stakeholders can identify patterns in customer behaviors and preferences for increased customer retention. This can lead to decreased churn rates and increased business profits. Thus, the goal of this analysis is to identify major predictors of churn and to determine if it's based on factors like 'Population', 'Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'Churn'.

Part II: Method Justification

B1. Explanation of Classification Method

A decision tree is a type of machine- earning algorithm used for predictive modeling and decision making. A decision tree can handle both categorical and numerical data. It is a tree like model where each internal node represents an attribute, each branch represents a decision and each leaf node represents a predicted outcome. Once constructed the decision tree can be used to make predictions on new data by traversing the tree from the root to a leaf node, based on the values of the input features. The predicted outcome is then determined by the majority vote of the same in the corresponding leaf node (Sehra, 2020).

B2. Summary of Method of Assumption

One assumption of a decision tree is it does not rely on any specific distribution assumptions about the data. Thus, decision trees are able to effectively handle a variety of data types, including non-linear, skewed, multi-modal, categorical, ordinal, and non-ordinal data (Navlani, 2018). This feature enhances the model's interpretability and ease of use.

B3. Packages or Libraries List

Analysis for the churn data set was completed utilizing Python. Python is an open-sourced programing language used for analysis and development. Python has a consistent syntax that makes coding and debugging user-friendly for beginners. Python is flexible and has the ability to import packages and to tailor data. The following packages were imported and used for their advantages. Below are the packages/libraries used for the analysis.

Packages/Libraries	Purpose
import pandas **as** pd	Main package for data uploading and manipulation
import numpy **as** np	Main package for working with arrays
from pandas.api.types **import** CategoricalDtype	represents a categorical data type with specified categories and ordering
import matplotlib.pyplot **as** plt	Visualization
import seaborn **as** sns	Advanced visualization
from scipy **import** stats	Normalization and statistics
from statsmodels.stats.outliers_influence **import** variance_inflation_factor	VIF function calculate the VIF to determine multicollinearity
from statsmodels.graphics.mosaicplot **import** mosaic	generates mosaic graphs for bivariant visualization of categorical datatypes
from sklearn.model_selection **import** train_test_split	allows to break dataset into training and testing portions
from sklearn **import** preprocessing	allows for functions for preprocessing of the data
from sklearn.feature_selection **import** SelectKBest, f_classif	selects the top K features, evaluates the relationship between each feature and the target variable
from sklearn.tree **import** DecisionTreeClassifier, `plot_tree`	used to create a decision tree to classify data; plott_tree used to visualize decision tree
from sklearn.ensemble **import** AdaBoostClassifier	an ensemble method to improve classifier performance
from sklearn.model_selection **import** GridSearchCV	implements a grid search algorithm for hyperparametric tuning

from sklearn.metrics **import** confusion_matrix	computes confusion matrix for a classification model
from sklearn.metrics **import** roc_auc_score	computes the receiver operating characteristic area under the curve (ROC AUC)
from sklearn.metrics **import** roc_curve	computes the receiver operating characteristic (ROC) curve for binary classification problem
from sklearn.metrics **import** classification_report	generates a text report that summarizes the performance of a classifier on a classification task
from sklearn.metrics **import** mean_squared_error	used to evaluate model by calculating MSE

Part III: Data Preparation

C1. Data Processing

One important data preprocessing goal that is needed for a decision tree is one-hot encoding. One hot encoding is relevant in the preprocessing step for a decision tree when dealing with categorical variables. Categorical variables have non-number values that cannot be used directly in the model. Thus, one hot encoding is a technique that creates binary indicator variables for each category in a categorical variable.

C2. Data Set Variables

Listed below is a description of the dependent variable (Churn) and the independent variables used in this analysis.

Variable Name	Data Class	Data Type	Description	Example
Population	Quantitative	Continuous	Population of customer residence	8165
Children	Quantitative	Continuous	Number of children of customer	5
Age	Quantitative	Continuous	Age of customer	30
Income	Quantitative	Continuous	Customer annual income reported	64256.81
Gender	Qualitative	Categorical	Customer gender	Male
Outage_sec_perweek	Quantitative	Continuous	Avg number of seconds per week of system outages in customer's neighborhood	12.63069124
Email	Quantitative	Continuous	Number of emails sent to customer over past year	10
Contacts	Quantitative	Continuous	Number of times customer contacted technical support	3
Yearly_equip_failure	Quantitative	Continuous	Number of times customer's equipment failed and replaced	0

Techie	Qualitative	Categorical	If customer considers themselves technically inclined	No
TechSupport	Qualitative	Categorical	If customer has technical support add-on	Yes
Tenure	Quantitative	Continuous	# of months customer has stayed with provider	10.06019902
MonthlyCharge	Quantitative	Continuous	Amount charged to customer monthly	160.8055418
Churn	Qualitative	Categorical	If the customer discontinued services in the last month	Yes

C3. Steps for Analysis

To determine which factors impact customer tenure in relation to the dependent variable 'churn' a decision tree was competed. Prior to completing the decision tree, there are data preprocessing steps that needed to be completed as shown below:

1. *df.info:* provided details for columns in the data set

2. *df.info(file_path):* provided details of data types for each column

3. *df.isnull().sum():* checked for missing/null values

4. *df.duplicated():* checked for duplicates in the data

5. df.shape: *Display the dimension of dataframe*

6. *df.hist(figsize = (15,15)):* visualize each column in dataset on histogram

7. *checked for outliers and removal of outliers:*
print(df.shape)
df = df[(np.abs(stats.zscore(df.select_dtypes(include=np.number))) < 3).all(axis=1)]
print(df.shape)

8. *display data set with all the columns*
df.head()

9. *df.describe():* shows statically information of continuous variables

10. *df.nunique():* calculated the number of unique values

11. *df.value_counts():* counted number of unique values

12. *Drop 'Bandwidth_gb_year' or 'Tenure'; from previous analysis, these two features were highly correlated*
df = df.drop('Bandwidth_GB_Year', axis = 1)

13. *Create dummy variables in order to encode categorical, yes/no data points into 1/0 numerical values* (GeeksforGeeks, 2023) *:*

```
df['DummyChurn'] = [1 if v == 'Yes' else 0 for v in df['Churn']]
df['DummyGender'] = [1 if v == 'Male' else 0 for v in df['Gender']]
df['DummyTechie'] = [1 if v == 'Yes' else 0 for v in df['Techie']]
df['DummyTechSupport'] = [1 if v == 'Yes' else 0 for v in df['TechSupport']]
```

14. *Drop original categorical features from dataframe:*

```
df = df.drop(columns=['Gender', 'Churn', 'Techie','TechSupport'])
```

15. *df.columns: v*isualize updated columns in the data frame

16. *df.head():* Display the first five rows of the data frame
17. *Feature Selection with SelectKBest:*

```
# Assign values to X for all predictor features
# Assign values to y for the dependent variable
X = df[['Population', 'Children', 'Age', 'Income', 'Outage_sec_perweek',
    'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'DummyGender',
'DummyTechie', 'DummyTechSupport']]
y = df['DummyChurn']

# Initialize the class and call fit_transform
skbest = SelectKBest(score_func=f_classif, k='all') # k=10
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 and their p-values
print("Selected Features:")
print(features_to_keep)
print("\nP-values:")
print(p_values)
```

output:

```
Selected Features:
8               Tenure
9        MonthlyCharge
11         DummyTechie
10         DummyGender
12    DummyTechSupport
Name: Feature, dtype: object

P-values:
                 Feature        p_value
8                 Tenure   0.000000e+00
9          MonthlyCharge  3.617355e-293
```

```
11           DummyTechie   2.565685e-10
10           DummyGender   5.504573e-03
12       DummyTechSupport  3.930658e-02
5                  Email   6.638305e-02
2                    Age   3.168504e-01
6               Contacts   4.548932e-01
7   Yearly_equip_failure   5.617654e-01
1               Children   8.625467e-01
3                 Income   9.548859e-01
4      Outage_sec_perweek  9.783497e-01
0             Population   9.981893e-01
```

18. *Check VIF for multicollinearity issues amongst these features (verify no multicollinearity concerns exist demonstrated by VIF >10)*

```
# Create a new DataFrame with the selected features
X_new = X[features_to_keep]

# Calculate the VIF for each feature
vif = pd.DataFrame()
vif["Feature"] = X_new.columns
vif["VIF"] = [variance_inflation_factor(X_new.values, i) for i in range(X_new.shape[1])]

# Print the VIFs
print(vif)
```

output:

```
            Feature       VIF
0            Tenure  2.457351
1     MonthlyCharge  3.887466
2       DummyTechie  1.190937
3       DummyGender  1.823764
4  DummyTechSupport  1.615528
```

C4. Cleaned Data Set

The new data frame was saved to a new file and attached as a csv file.

```
# Prepared dataset saved to new file
df.to_csv('D209_prepared_churn_task2.csv', index=False)
```

Part IV: Analysis

D1. Splitting the Data

The data was split into training and test data set and attached using the following code. This is to ensure the model has similar accuracy when predicting unseen data.

Split the data set with an 80/20 split
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_train.2csv')
pd.DataFrame(y_test).to_csv('y_test2.csv')

D2. Output and Intermediate Calculations

To properly analyze the data this analysis began by performing a feature selection on the data using the 'SelectKBest' function. This began by assigning values to 'X' for all predictor features and assigning values to 'Y' for the dependent variable. Then p-values were calculated to select the statistically significant features given the following output:

```
Selected Features:
8               Tenure
9        MonthlyCharge
11         DummyTechie
10         DummyGender
12    DummyTechSupport
Name: Feature, dtype: object

P-values:
                 Feature        p_value
8                 Tenure   0.000000e+00
9          MonthlyCharge  3.617355e-293
11           DummyTechie   2.565685e-10
10           DummyGender   5.504573e-03
12      DummyTechSupport   3.930658e-02
5                  Email   6.638305e-02
2                    Age   3.168504e-01
6               Contacts   4.548932e-01
7    Yearly_equip_failure   5.617654e-01
1               Children   8.625467e-01
3                 Income   9.548859e-01
4      Outage_sec_perweek   9.783497e-01
0             Population   9.981893e-01
```

From the feature analysis, the variables to keep were 'tenure', 'monthlycharge', 'dummytechie', 'dummygender, 'dummygender'. 'Tenure' and 'MonthCharge' have very low p-values indicating they are highly correlated with the dependent variable, 'dummyChrun'. 'DummyTechie' and 'DummyGender and 'DummyTechSupport' also have p-values of less than

0.05 indicating they are statistically significant to churn but at a less rate. All the other features had p-values greater than 0.05, meaning they are not statically significant.

The next step was to use the variance inflation factors to check for VIF for multicollinearity issues among the features. To get the following results:

```
             Feature       VIF
0             Tenure  2.457351
1      MonthlyCharge  3.887466
2        DummyTechie  1.190937
3        DummyGender  1.823764
4  DummyTechSupport  1.615528
```

The VIF of the features are all less than 5 indicating there is not a significant amount of multicollinearity amongst these features and not a concern for this analysis. Cross-validation was used to validate the performance of the model. Cross-validation helps to reduce the risk of overfitting by providing a more reliable estimate of a model's performance on the data. A higher score correlates to a better fit model. The scores for the 5 folds ranged from 0.71899441 to 0.88547486, with a mean score of 0.815883798. This suggests that the decision tree classifier model is performing moderately well but may need improvement.

```
Cross-validation scores: [0.71899441 0.7849162 0.88547486 0.87206704 0.813
96648]
```

The next steps were splitting the data into a training and testing set for the accuracy of the model and saving them to new files. Next was defining the hyperparameters of the grid to search and creating a decision tree classifier. This provided the best hyperparameters. The accuracy and AUC of the model were also calculated. The accuracy score indicates the model correctly predicted the outcome for 82.9% of the cases. The AUC of 0.5 indicates random guessing while a score of 1 indicates a perfect classifier. The AUC of the model is 0.7361084809845899. The best hyperparameters for the decision tree are also shown below.

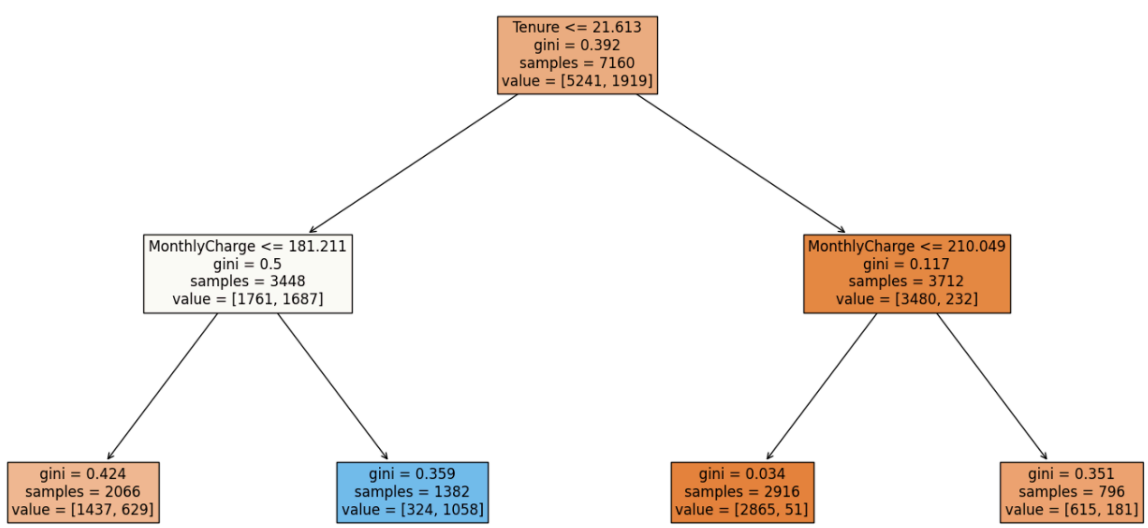
```
Accuracy: 0.829050279329609
AUC: 0.7361084809845899
Best hyperparameters: {'max_depth': 2, 'min_samples_leaf': 1}
```

The decision tree model was then fit with the found parameters and the plot for the decision tree was completed. A classification report was then used to assess the precision, recall, F1-score, and support of the model to provide:

	precision	recall	f1-score	support
0	0.85	0.93	0.89	1323
1	0.73	0.54	0.62	467
accuracy			0.83	1790
macro avg	0.79	0.74	0.76	1790
weighted avg	0.82	0.83	0.82	1790

The model's performance is evaluated on two classes of 0 and 1. The precision of class 0 indicates 85% of predicted instances were true positives. The recall for class 0 indicates 93% of instances were correctly classified. The F-1 score of class 0 shows good overall accuracy for this class. The precision of class 1 indicates 73% of predicted class 1 instances were true positives and the recall indicates 54% of true class 1 instances were correctly classified. The F1-score of class 1 shows poor accuracy. The last row in the model is the weighted average and provides the overall summary of the model's performance. Then accuracy is 0.83 indicating the model correctly predicts the class label for 83% of instances. The macro-avg f1-score is 0.76 indicating reasonable overall performance but also indicating some imbalance in the model.

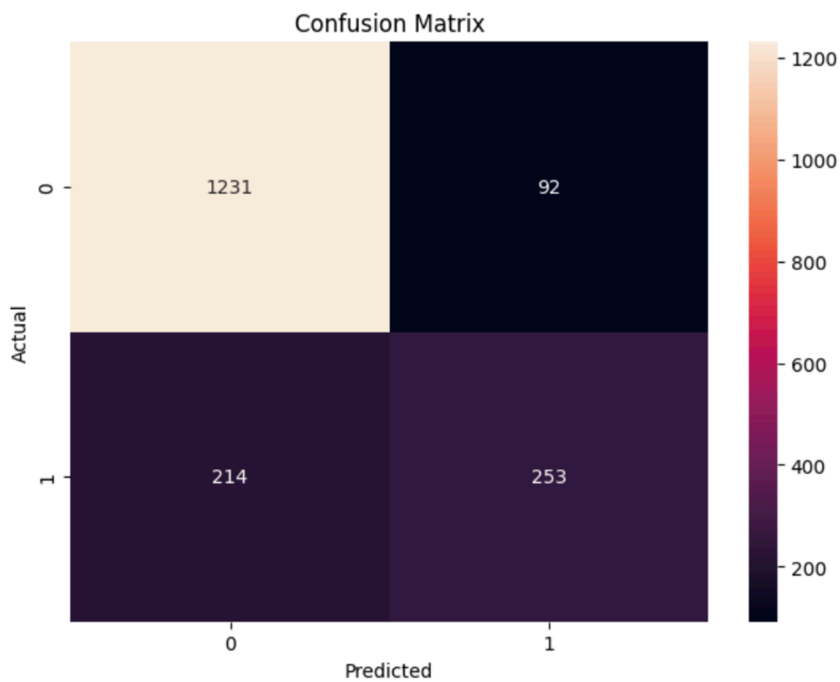
Decision Tree:



A confusion matrix was then created. A confusion matrix is useful in evaluating a binary classification model's performance and its ability to identify the types of errors the model is making. The confusion matrix revealed there were 1231 instances the model correctly classified as negative and 92 instances that incorrectly classed as positive. There were 214 instances the model incorrectly classifies a negative by the model. And there were 253 instances the model correctly classified as positive.

Confusion matrix:

```
[[1231   92]
 [ 214  253]]
```



The accuracy of the training and test data sets were calculated next. The accuracy of both sets were very similar indicating they are performing similarly.

Accuracy:

```
Training set accuracy: 0.8344972067039106
Test set accuracy: 0.829050279329609
```

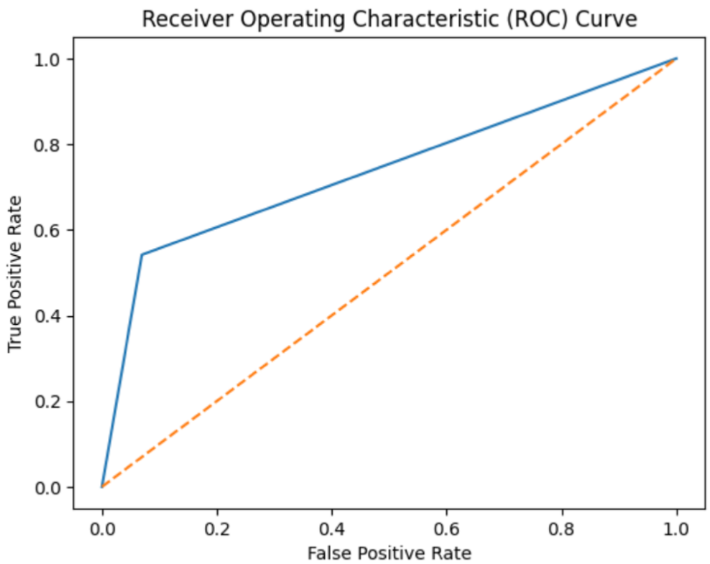
The AUC-ROC score was 0.836 suggesting the model is performing moderately well in distinguishing between the positive and negative classes. An AUC-ROC score of 0.5 shows random guessing while 1 shows a perfect performance.

AUC-ROC:

```
AUC-ROC score: 0.7361084809845899
```

The receiver operating characteristic (ROC) curve is used for evaluating the performance of a binary classification model. An AUC-ROC score of less than 0.6 is considered to be poor performing. The ROC plot's central diagonal line represents a 50% correct classification rate, which would be expected of a completely random classification (Brownlee, 2018). A classifier's performance can be plotted on this graph, with a curve below the diagonal reflecting a poor prediction rate and a curve above the line reflecting a good prediction rate. Thus, this model has a good prediction rate.

ROC Curve:



The mean squared error and root mean squared error were then calculated.

output:
Mean squared error: 0.17094972067039105
Root mean squared error: 0.4134606639940383

D3. Code

```
# Create dummy variables in order to encode categorical, yes/no data points into 1/0 numerical values.
df['DummyChurn'] = [1 if v == 'Yes' else 0 for v in df['Churn']]
df['DummyGender'] = [1 if v == 'Male' else 0 for v in df['Gender']]
df['DummyTechie'] = [1 if v == 'Yes' else 0 for v in df['Techie']]
df['DummyTechSupport'] = [1 if v == 'Yes' else 0 for v in df['TechSupport']]
```

```
# Drop original categorical features from dataframe
df = df.drop(columns=['Gender', 'Churn', 'Techie','TechSupport'])

# Assign values to X for all predictor features
# Assign values to y for the dependent variable
X = df[['Population', 'Children', 'Age', 'Income', 'Outage_sec_perweek',
      'Email', 'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge', 'DummyGender',
'DummyTechie', 'DummyTechSupport']]
y = df['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 and their p-values
print("Selected Features:")
print(features_to_keep)
print("\nP-values:")
print(p_values)


# Check VIF for multicollinearity issues amongst these features

# Create a new DataFrame with the selected features
X_new = X[features_to_keep]

# Calculate the VIF for each feature
vif = pd.DataFrame()
vif["Feature"] = X_new.columns
vif["VIF"] = [variance_inflation_factor(X_new.values, i) for i in range(X_new.shape[1])]

# Print the VIFs
print(vif)

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

# Print the cross-validation scores
print("Cross-validation scores:", scores)
```

```
#Split the data set with an 80/20 split
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')


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

# Create a decision tree classifier
clf = DecisionTreeClassifier()

# Perform grid search using 5-fold cross-validation
grid_search = GridSearchCV(clf, param_grid, cv=5)

# Train the model on the training data using grid search
grid_search.fit(X_train, y_train)

# Predict the class labels of the testing data using the best model
y_pred = grid_search.predict(X_test)

# Calculate the accuracy and AUC of the best model
accuracy = accuracy_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("AUC:", auc)

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


#Fit the decision tree model with the found parameters
dt = DecisionTreeClassifier(max_depth = 2, min_samples_leaf = 1)
dt.fit(X_train, y_train)

y_pred = dt.predict(X_test)

# Plot the decision tree
plt.figure(figsize=(20,10))
plot_tree(grid_search.best_estimator_, filled=True, fontsize=12, feature_names=X.columns)
plt.show()
```

```
#Classification Report
print(classification_report(y_test, y_pred))

#Print confusion matrix
cnf_matrix = confusion_matrix(y_test, y_pred)
print(cnf_matrix)

#Use seaborn heatmap to visualize the confusion matrix
sns.heatmap(pd.DataFrame(cnf_matrix), annot = True, fmt = 'g')
plt.tight_layout()
plt.title('Confusion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')


# Predict the class labels of the training and 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 on the training and testing data
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 on the training and testing data
print("Training set accuracy:", accuracy_train)
print("Test set accuracy:", accuracy_test)


# Assuming y_pred and y_true are the predicted and true labels respectively
auc_roc = roc_auc_score(y_test, y_pred)
print("AUC-ROC score:", auc_roc)


#ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.plot(fpr, tpr)
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.show()


#Calculate MSE & RMSE
mse_dt = mean_squared_error(y_test, y_pred)
rmse_dt = mse_dt**(1/2)

print("Accuracy: ", dt.score(X_test, y_test))
print("Mean squared error: ", mse_dt)
print("Root mean squared error: ", rmse_dt)
```

Part V: Data Summary and Implications

E1. Accuracy and MSE

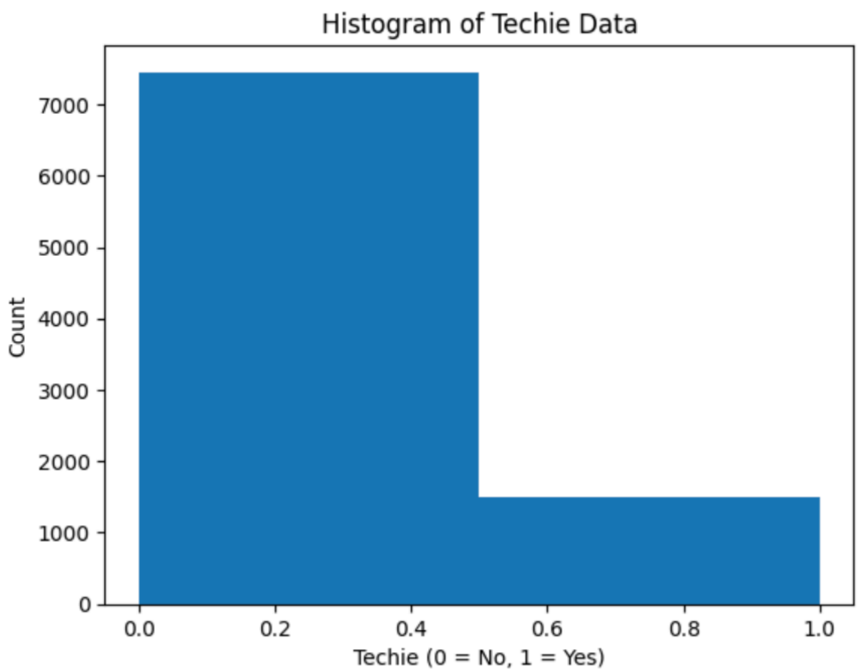
The accuracy of the model is 0.829 showing that the model correctly predicts the class labels for about 83% for the instances in the test set. The mean squared error (MSE) is 0.171. This MSE suggests the model's predictions were off by about 0.17 units from the true values. A lower MSE generally indicates better permanence meaning the model's predictions are closer to the truce values on average. The root mean squared error (RMSE) is 0.413. This suggests the model's predictions were off by about 0.413 units from the true values. A lower RMSE also generally indicates a better performance. However, this metric is often more appropriate for regression models and may not be accurate for a classification model.

E2. Results and Implications

Overall, the model demonstrates good performance. The SelectKBest approach was first used to select which features were the most significant in the model. This is indicated by the model having an accuracy of 0.829 and an AUC score of 0.736. The precious for class 0 (not churn) is 0.85 which means the model predicts a customer will. not churn and it's correct 85% of the time. The recall for class 0 is 0.93. This means that out of all customers who did not churn the model correctly identified 93% of them. The precision for class 1 (churn) is 0.73 indicating when the model predicts a customer will churn, it is correct 73% of the time. The recall for class 1 is 0.54 meaning that out of all the customers who churned the model correctly identified 54% of them. The confusion matrix shows the model correctly predicted 1231 customers who did not churn and 253 customers who did churn. The hyperparameters were relatively simple and may be prone to underfitting. In conclusion, the model correctly identifies a high percentage of customers who did not churn but struggle to identify customers who actually did churn. The hyperparameters suggest that a more complex model may be needed for improved predictions. The model accuracy could be further improved by performing hyperparameter tuning on additional parameters to increase the accuracy of the model.

E3. Limitations

One limitation of this analysis is decision trees can be biased. Bias can come from the tree being too shallow or if the data is unbalanced. For example, the target variable, 'Techie', contains about double the number of *No* values as the number of *Yes* values.



E4. Course of Action

Based on the model there are several recommendations that could be made to reduce customer churn. The model suggests customers who have been with the company for a longer time and who have higher monthly charges are less likely to churn. Thus, the company could focus on retaining these customers by offering them targeted promotions like loyalty rewards. The model also suggests the company should invest in improving tech support services as seen from the model indicating customers who have tech support are less likely to churn. The analysis also suggests gender may play a role in customer churn. The company could target potential reasons for disparities in gender and question why female customers are more likely to churn than male customers.

Part V: Demonstration

F. Panopto Recording

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=4cab612a-49fa-48c5-9c73-afd40115b700

G. Sources of Third-Party Code

Brownlee, J. (2018). Metrics to evaluate machine learning algorithms in Python. Retrieved 10/20/2021 from https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python

GeeksforGeeks. (2023, January 11). *ML: Label encoding of datasets in Python*. GeeksforGeeks. Retrieved March 15, 2023, from https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/

H. Sources

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

Navlani, A. (2018, December 28). *Attention Required! | Cloudflare*. Datacamp. https://www.datacamp.com/community/tutorials/decision-tree-classification-python

Sehra, C. (2020, November 30). *Decision Trees Explained Easily - Chirag Sehra*. Medium. https://chiragsehra42.medium.com/decision-trees-explained-easily-28f23241248