D209 Task 2

This is the code for my d209 performance assessment task 2. Student id: 012047746

A1:PROPOSAL OF QUESTION

The research question I want to ask for this task 2, is "whether or not a decision tree algorithm can predict if a customer will churn using binary qualitative data?". This analysis is similar to my task 1 for D209 but uses a different algorithm. This is a useful analysis to explore because it helps us understand what factors may possibly affect what causes someone to be a customer that churns and a customer that doesn't. It can also help with planning for the future, to see if there will be more or less customers that are churning.

A2:DEFINED GOAL

The goal is to determine if its possible to create a decision tree model that can accurately determine whether or not a customer is going to churn given the input of binary qualitative factors. In order to do this, I am going to use scikitlearn and the decision tree method. From the accuracy of the model, we can determine whether or not it will be useful to use this forecast to try and implement into business practices.

B1:EXPLANATION OF PREDICTION

For the analysis I am going to be using decision trees to classify contributing factors as to whether or not a customer is going to churn. Decision trees work by starting with a root node, and defining different possible factors that could affect an outcome. For instance, it could start by being if you are outside, and then one of the nodes if you are outside could be whether the weather is sunny or not. This would continue until you reach an outcome or leaf node that categorizes the steps into an outcome.

B2:SUMMARY OF METHOD ASSUMPTION

One assumption of the decision tree methods is that all variable can be represented as a binary choice. This is because the decision tree will create splits, and develop two subsets of the data based upon a particular feature. It also is assumed that the data can be split into smalled subsets of data. Another assumption of decision trees is that they assume that there is feature independence between the inputted features for the model.

These are all things that we have to take into account when it comes to implementing decision trees. For instance, we can't use data that is not representable as a binary choice. When it comes to numerical data we would have to determine a way for the model to split the data into two subsets, like above the value of ten and below the value of ten. We also should avoid using features that are not independent of one another, like height and weight.

B3:PACKAGES OR LIBRARIES LIST

I have used the following packages for my analysis:

- Pandas: This library is essential to import the CSV and apply analysis to the data. This is also used to map the qualitative variables to a numeric value depending on whether its 'yes' or 'no'.
- numpy: We use numpy to use arrays and set up the dataframe to be used for statistical analysis

- scikitlearn: This is the library that is used to bring in the decision tree model. I also use mse, classification report, and auc scoring

In [2]:
```
# import the libraries
import pandas as pd
from pandas import DataFrame
import numpy as np
import sklearn
from sklearn.model_selection import train_test_
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_repo
```

C1:DATA PREPROCESSING

We need to remove the numerical variables and non-binary categorical variables from the dataset. We are doing this as it does not fit our initial goal of testing whether or not we can build this model based off of binary qualitative factors. To accurately test this, we need to remove all of these variables. This process is done later on in the code by selecting only the variables we need and keeping only those variables in the dataframe.

C2:DATA SET VARIABLES

For the qualitative variables we need:

- Churn: This is a binary variable with either yes or no values. This variable is categorical.
- Techie: This is a binary variable with either yes or no values. This variable is categorical.
- Port_modem: This is a binary variable with either yes or no values. This variable is categorical.
- Tablet: This is a binary variable with either yes or no values. This variable is categorical.
- Phone: This is a binary variable with either yes or no values. This variable is categorical.
- Multiple:This is a binary variable with either yes or no values. This variable is categorical.
- OnlineSecurity: This is a binary variable with either yes or no values. This variable is categorical.
- OnlineBackup: This is a binary variable with either yes or no values. This variable is categorical.
- DeviceProtection: This is a binary variable with either yes or no values. This variable is categorical.
- TechSupport: This is a binary variable with either yes or no values. This variable is categorical.
- StreamingTV: This is a binary variable with either yes or no values. This variable is categorical.
- StreamingMovies: This is a binary variable with either yes or no values. This variable is

categorical.
- PaperlessBilling: This is a binary variable with either yes or no values. This variable is categorical.

All these variables are categorical

```
In [3]: dfq_cq = ['Churn','Techie','Port_modem','Tablet
```

C3:STEPS FOR ANALYSIS

I will start by selecting only the variables that we need. This means removed all numerical and non-binary qualitative variables.

```
In [4]: df = pd.read_csv('churn_clean.csv')
```

```
In [5]: dfq = df[dfq_cq]
        dfq
```

Out[5]:

	Churn	Techie	Port_modem	Tablet	Phone	Multi
0	No	No	Yes	Yes	Yes	
1	Yes	Yes	No	Yes	Yes	Y
2	No	Yes	Yes	No	Yes	Y
3	No	Yes	No	No	Yes	
4	Yes	No	Yes	No	No	
...	...	...	...	...	...	
9995	No	No	Yes	Yes	Yes	Y
9996	No	No	No	No	Yes	Y
9997	No	No	No	No	Yes	Y
9998	No	No	No	Yes	No	Y
9999	No	No	Yes	No	Yes	Y

10000 rows × 13 columns

Check for nulls in data

We check for nulls in the data to make sure that there are no empty values. In the table below you can see that there are none.

In [6]:
```
dfq.isnull().sum()
```

```
Out[6]: Churn               0
        Techie              0
        Port_modem          0
        Tablet              0
        Phone               0
        Multiple            0
        OnlineSecurity      0
        OnlineBackup        0
        DeviceProtection    0
        TechSupport         0
        StreamingTV         0
        StreamingMovies     0
        PaperlessBilling    0
        dtype: int64
```

One hot encoding

We use the .map function to map yes to 1 and 0 to no

```
In [7]: dfq_ql = ['Churn','Techie','Port_modem','Tablet
        dfq_e = pd.DataFrame()
        for column in dfq_ql:
            dfq_e[column] = df[column].map({'Yes': 1,'N
```

```
In [8]: dfq_e
```

Out[8]:

	Churn	Techie	Port_modem	Tablet	Phone	Multi
0	0	0	1	1	1	
1	1	1	0	1	1	
2	0	1	1	0	1	
3	0	1	0	0	1	
4	1	0	1	0	0	
...	...	...	...	...	...	
9995	0	0	1	1	1	
9996	0	0	0	0	1	
9997	0	0	0	0	1	
9998	0	0	0	1	0	
9999	0	0	1	0	1	

10000 rows × 13 columns

There is no need to chcek for outliers since there are non for binary qualitative variables.

C4:CLEANED DATA SET

Here we export the cleaned dataset. The main difference between this cleaned dataset and the original one is that we have one hot encoded the categorical binary data into 0 and 1. This allows us to read into the model easier the data that we want it to predict on.

In [23]:
```
dfq_e.to_csv('prepared_data_task2.csv')
```

D1:SPLITTING THE DATA

We use the train test split function to split our data into a training dataset and a testing dataset. The model will be trained on the training dataset, and then will compare results against the testing dataset to determine accuracy.

In [10]:
```
# Split into X and y
X = dfq_e.drop("Churn", axis=1)
y = dfq_e["Churn"]
```

In [11]:
```
# split into test and training data sets for bo
X_train, X_test, y_train, y_test = train_test_s
```

In [12]:
```
pd.DataFrame(X_test).to_csv('task2_test_data.cs
```

In [13]:
```
pd.DataFrame(X_train).to_csv('task2_train_data.
```

D2 & D3:OUTPUT AND INTERMEDIATE CALCULATIONS

Here we create the model, we choose a number for random state so the model results are reproducable and consistent for when we run this again. Then we fit the training data to the model. After that we run the predictions which produces the results for us in churn_results_df.

```
In [14]: dt_model = DecisionTreeClassifier(random_state=
```

```
In [15]: dt_model.fit(X_train, y_train)
```

```
Out[15]: DecisionTreeClassifier(random_state=42)
```

```
In [16]: # predicts the actual class for churn
         y_pred = dt_model.predict(X_test)
```

```
In [17]: # used for class probabilties
         y_prob = dt_model.predict_proba(X_test)[:, 1]
```

```
In [18]: #prints the results
         churn_results_df = pd.DataFrame({
             'Churn Label': y_test,
             'Predicted Label': y_pred,
             'Probability Score': y_prob
         })
```

```
In [19]: churn_results_df
```

Out[19]:

	Churn Label	Predicted Label	Probability Score
8158	0	0	0.500000
3484	0	0	0.428571
5443	0	0	0.000000
7278	0	0	0.000000
8278	0	1	0.666667
...	...	...	...
3759	0	0	0.000000
7659	0	0	0.000000
8081	0	0	0.400000
3072	0	0	0.000000
3530	0	0	0.000000

2000 rows × 3 columns

E1:ACCURACY AND MSE

Here we calculate the MSE which accounts for variation in our model. We print the accuracy and the MSE, and then use the classification report for more details.

In [20]:
```
mse = mean_squared_error(y_test, y_prob)
```

In [21]:
```
# print the results
print(f"Accuracy: {dt_model.score(X_test, y_tes
print(f"MSE: {mse}")
```

```
Accuracy: 0.7265
MSE: 0.2223947000989015
```

```
In [22]: print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   s
upport

           0       0.78      0.87      0.82
1470
           1       0.48      0.32      0.38
530

    accuracy                           0.73
2000
   macro avg       0.63      0.60      0.60
2000
weighted avg       0.70      0.73      0.71
2000
```

E2:RESULTS AND IMPLICATIONS

The accuracy of the model that was developed for this analysis is 0.7265. Accuracy in relation to this model represents the amount of points that our model correctly categorized based on the test data.

The MSE of the model that was develop for this analysis is 0.2223. MSE means is the average of the squared difference between the predicted and actual values. A lower MSE means lower variance and a closer prediction to the actual result.

E3:LIMITATION

A limitation is that the decision trees may overly give weight to nodes earlier on in the tree. This may reduce the amount of nuance that is considered in the analysis in regards to how different variances in variables may interect with one another.

E4:COURSE OF ACTION

The next course of action would to assess the accuracy of this model with the model in task 1 because they both attempt to predict churn in customers. If one is more accurate than another, than it may make sense to implement that model into assisting with decision making when it comes to whether or not a customer will churn. This can help us decide what kind of customer that we need to focus on, and if there are certain commonalities among these customers that the model is predicting is going to churn.

G:SOURCES FOR THIRD-PARY CODE

No third party code used

H:SOURCES

https://www.mastersindatascience.org/learning/machine-learning-algorithms/decision-tree/

https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html#:~:text=Below%20are%20some%20of%2

https://medium.com/featurepreneur/decision-tree-regression-9170dc2f6a95#:~:text=We%20can%20predict%20the%2

https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/#h-decision-tree-assumptions