WGU D209 PA 2

Andrew Shrestha

Part I: Research Question

A1. Question: Are we able to accurately predict which customers are at risk of Churn? Can we identify the significant variables that predict the churn outcome? This will be done through the use of Decision Tree Algorithms.

A2. Objectives and Goals: There are 2 very important reasons why a company would benefit from tracking its churn rate. First, the loss of customers is equivalent to loss of revenue, and due to this being a driving factor in a company's future growth and development, becomes dire to prevent. Secondly, it is generally more expensive to gain new customers than it is to maintain existing ones, therefore it is in the company's and shareholder's best interest to allocate resources to prevent customer churn **(Zendesk 2021).**

With the knowledge of the significant variables identified and our prediction of which customers will churn, companies will have a strong sense of where to allocate their resources to ensure that customer churn is reduced.

Part II: Method Justification

B1. Explanation of our Prediction Model

Decision Trees or Classification and Regression Trees (CART) is a non- parametric supervised learning technique based on multiple decision rules which are derived from data features. This tree structure consists of 3 main parts: Root Nodes, Internal Nodes, and Leaf Nodes.

Root Nodes are the first node in the training set. These then get split into subgroups of internal nodes through the decision tree algorithm Internal nodes are then split into nested nodes based on threshold value of an attribute. Finally, this is then split into their leaf nodes which will ultimately hold the decision.

As we progress down the tree like structure, we will end up getting to the optimal choice.

B2. Summary of One Assumption for Decision Tree

The most significant assumption that holds in the decision tree algorithm is that the whole training set is considered the root at the very beginning. Also because of the nature and structure of the decision tree, values are preferred to be categorical. If this is not the case, then we are to discretize our data before using it in the model.

B3. Package and Library List

In this Project, I will be leveraging the use of Python Programming language as it is a general purpose and production ready language with clear and easily interpretable syntax. On a personal note, I have used Python before in previous Data Science Certifications and have introductory knowledge on the language.

Libraries and Packages that will support the data cleaning process will be as follows:

Pandas: create and store the data in a tabular form which makes it extremely effective for data manipulation. Pandas is also able to detect missing values and replace them with an “NAN” or “NA” and is useful in manipulating both numerical and string values.

Numpy: Grants us the use of multi dimensional arrays and computational functions for mathematical and statistical analysis

Matplotlib: Grants us the use of data visualization and creates various types of plots and graphs with the goal of representing data. Also very handy for visually providing insights into identifying potential outliers or missing data.

Seaborn: Based on Matplotlib, it gives us advanced data visualization with more informative statistical graphics

Scikit – Learn: predictive analysis with useful features such as Classifications, Regression, and Clustering.

SciPy: mathematical analysis for integration, optimization, algebra, and statistics

Part III: Data Preparation

C1. One Preprocessing Goal Relevant to Classification

The one preprocessing goal for this analysis is to ensure that the binary categorical variables of Yes/No or Male/Female are all converted to binary dummy variables with results of 1 or 0. This will be the most important step to prepare our data for the analysis.

C2. Data Set Variables and Classification

Initial Data Set Continuous Variables:

1) Children
2) Income
3) Outage_sec_perweek
4) Email
5) Contacts
6) Yearly_equip_failure
7) Tenure
8) MonthlyCharge
9) Bandwidth_GB_Year

Initial Data Set Binary Categorical Variables:

1) Techie (Yes/No)
2) Contract (Month-to-Month/One Year/Two Year)
3) Port_modem(Yes/No)
4) Tablet (Yes/No)
5) InternetService (DSL/Fiber Optic/None)
6) Phone (Yes/No)
7) Multiple (Yes/No)
8) OnlineSecurity (Yes/No)
9) OnlineBackup (Yes/No)
10) DeviceProtection (Yes/No)
11) TechSupport (Yes/No)
12) StreamingTV (Yes/No)
13) StreamingMovies (Yes/No)

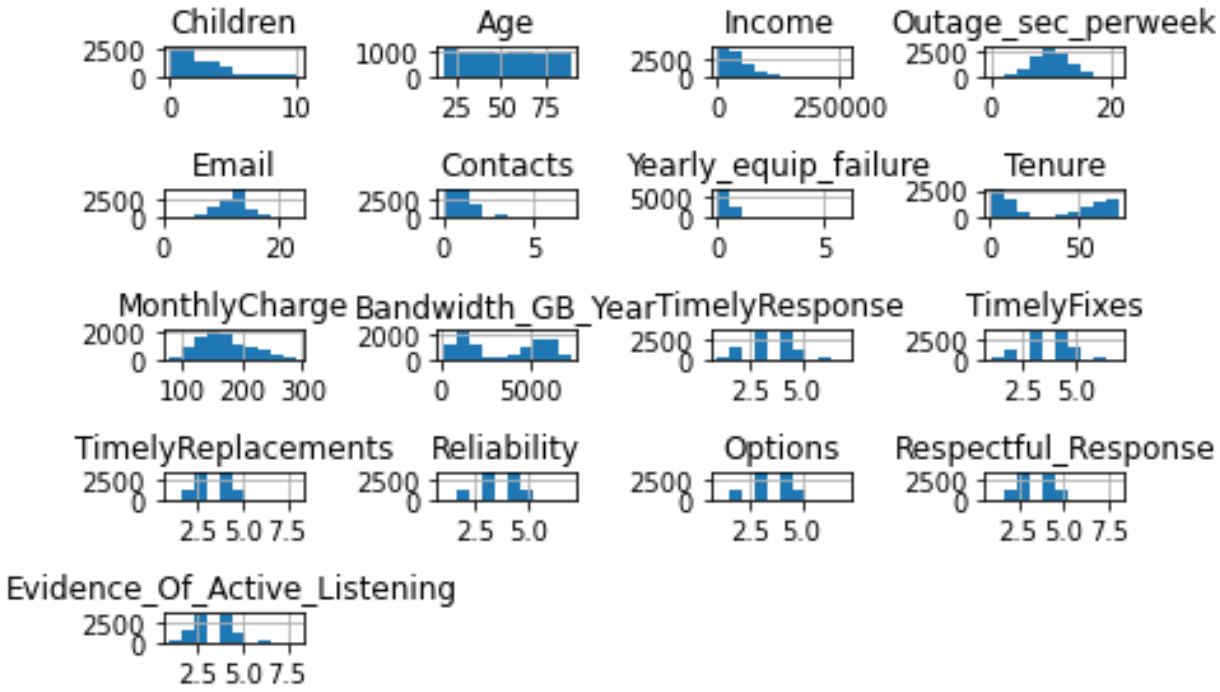
Initial Data Set Discrete Numerical Variables:

1) Item1: Timely Response
2) Item2: Timely Fixes
3) Item3: Timely Replacements
4) Item4: Reliability
5) Item5: Options
6) Item6: Respectful Response
7) Item7: Courteous Exchange
8) Item8: Evidence of Active Listening

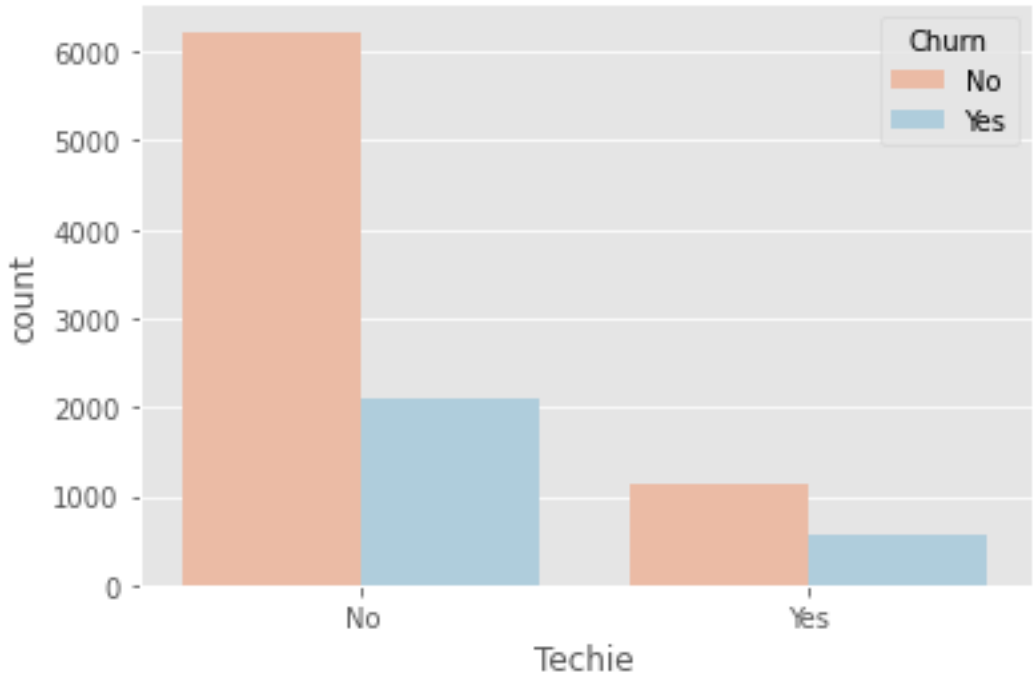
C3. Data preparation code

```
#Importing the standard Python libraries for Analysis and Visualizations
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import sklearn
from sklearn import datasets
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report


#Load the CSV data set file into our Pandas Dataframe
churn_clean_df =
pd.read_csv(r"C:\Users\andre\OneDrive\Desktop\churn_clean.csv")


#now we want to get a surface level understanding of our data that we just
imported via basic statistical analysis
churn_clean_df.shape
churn_clean_df.columns

churn_clean_df.head()
```

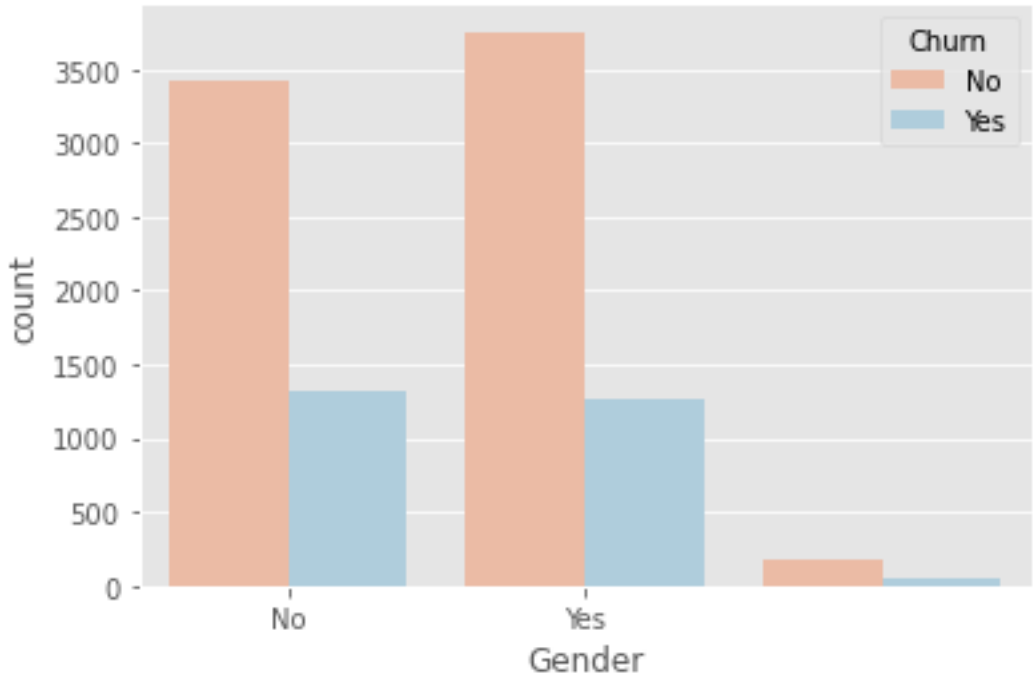
```
churn_clean_df.info

churn_clean_df.describe()

churn_clean_df.dtypes



#we want to now rename the survey questions from basic numerical labels to
more detailed labels in order to avoid confusion and make it clearer in the
analysis process
churn_clean_df.rename(columns = {'Item1':'TimelyResponse',
                     'Item2':'TimelyFixes',
                      'Item3':'TimelyReplacements',
                      'Item4':'Reliability',
                      'Item5':'Options',
                      'Item6':'Respectful_Response',
                      'Item7':'Courteous_Exchange',
                      'Item8':'Evidence_Of_Active_Listening'},
          inplace=True)


#let us now create some visualizations of the significant continuous and
categorical variables present in the data set via hist plots
churn_clean_df[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email',
          'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',
          'Bandwidth_GB_Year',
'TimelyResponse','TimelyFixes','TimelyReplacements','Reliability', 'Options',
'Respectful_Response', 'Evidence_Of_Active_Listening']].hist()
plt.tight_layout()
```



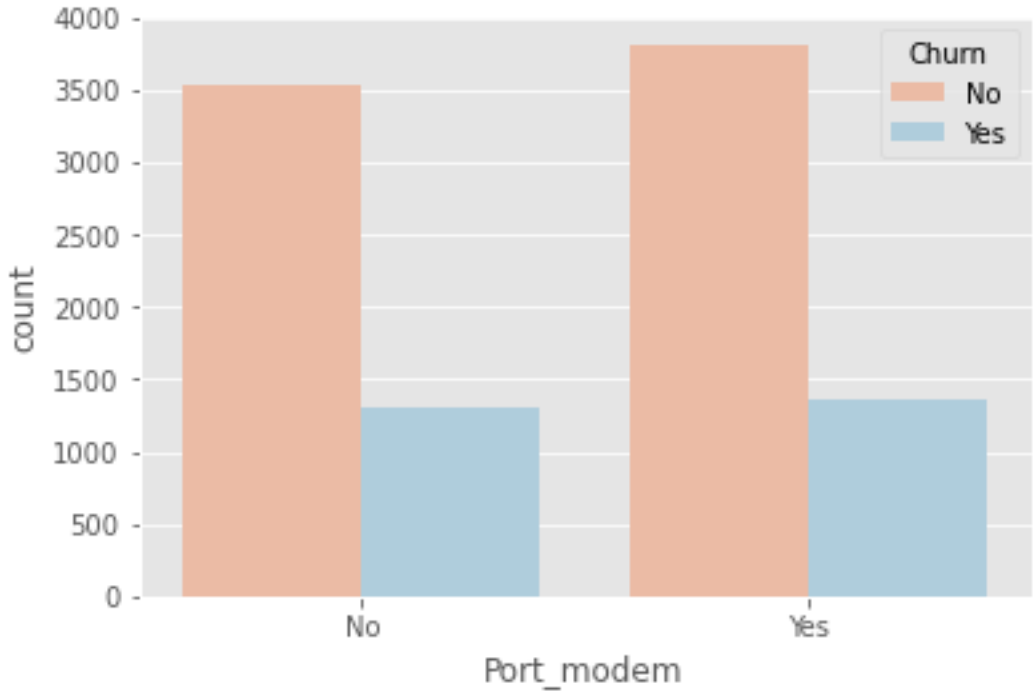

```
#Now using ggplot, we can also see the bivariate visualizations between our
categorical binary variables and that of 'Churn'
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='Techie', hue='Churn', data=churn_clean_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



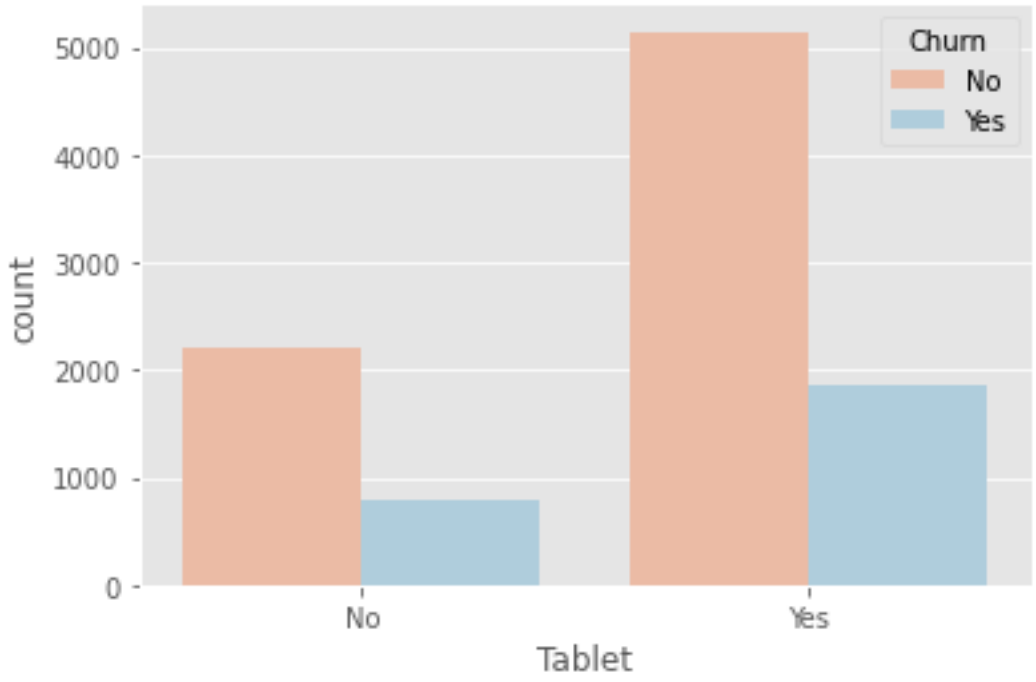

```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='Gender', hue='Churn', data=churn_clean_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



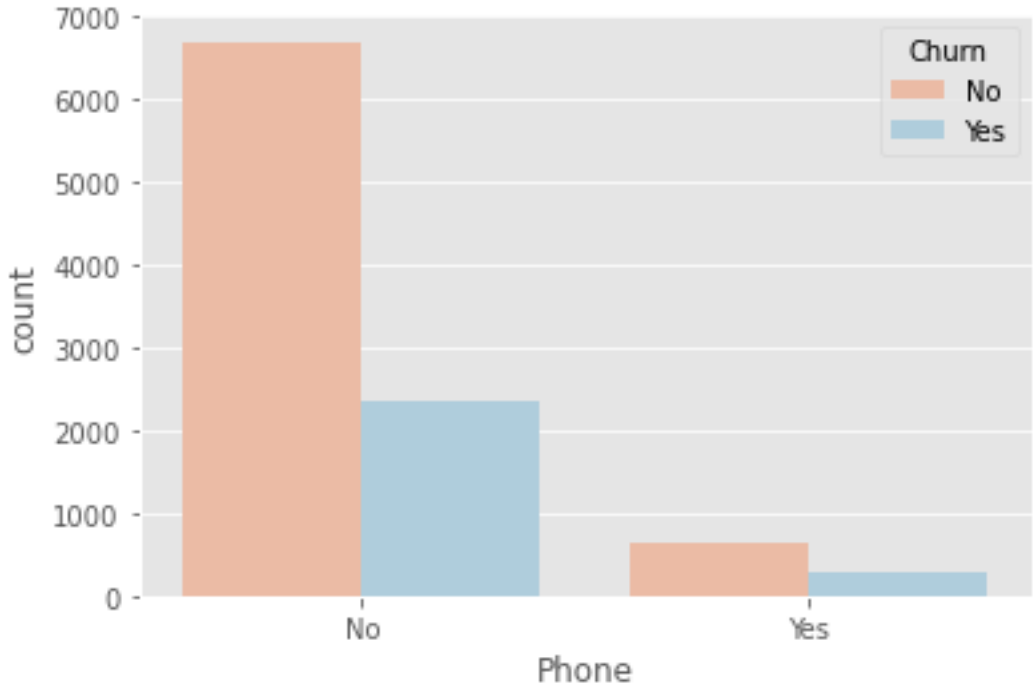
```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='Port_modem', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



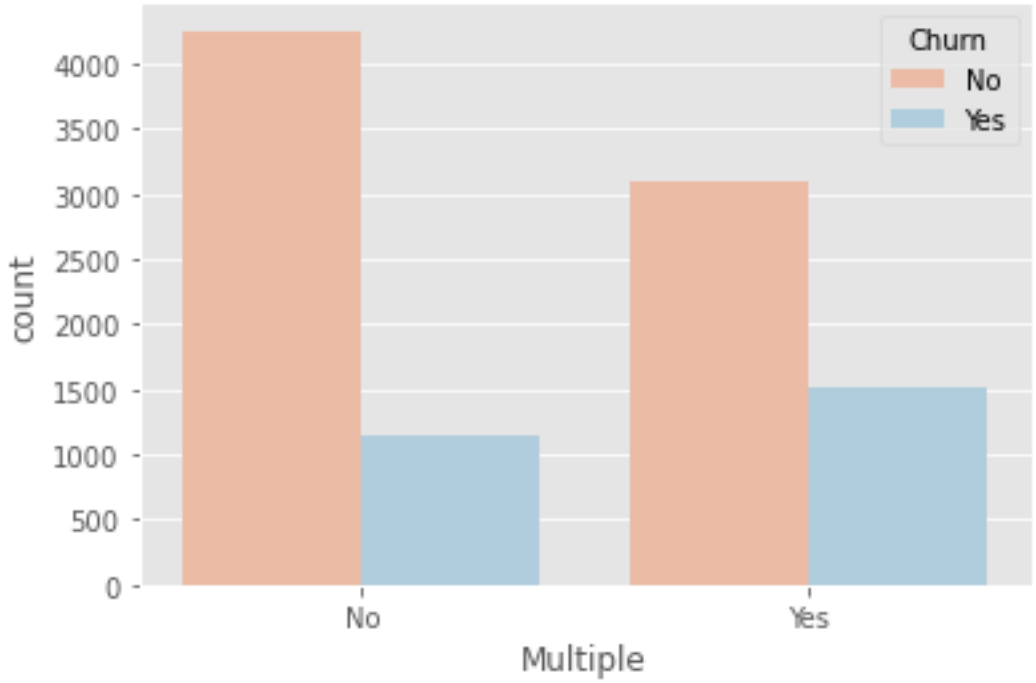

```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='Tablet', hue='Churn', data=churn_clean_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



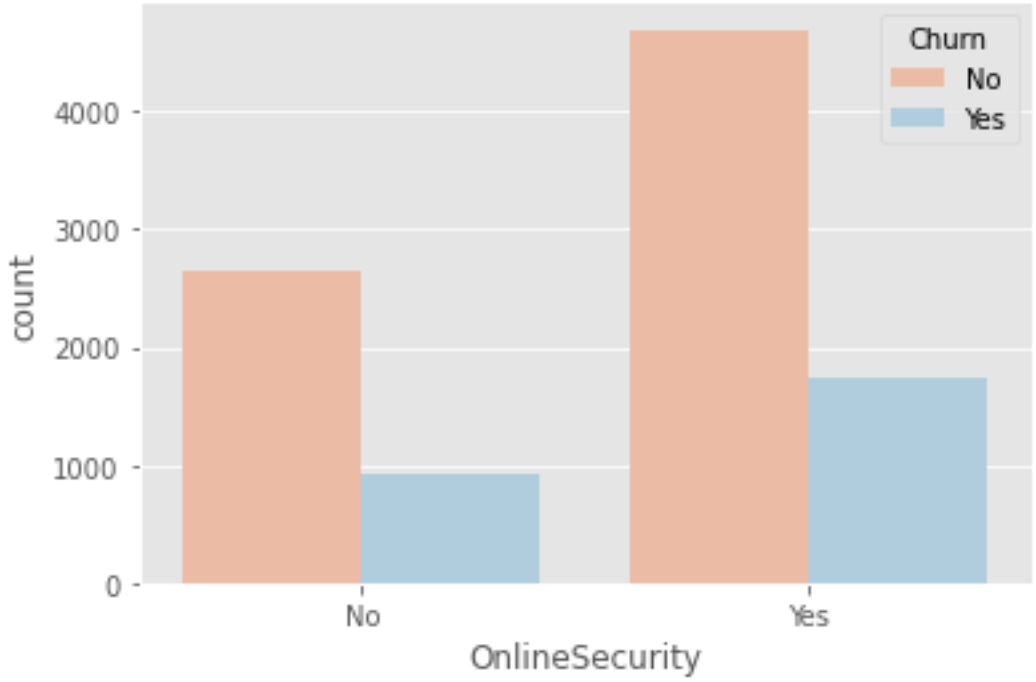

```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='Phone', hue='Churn', data=churn_clean_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



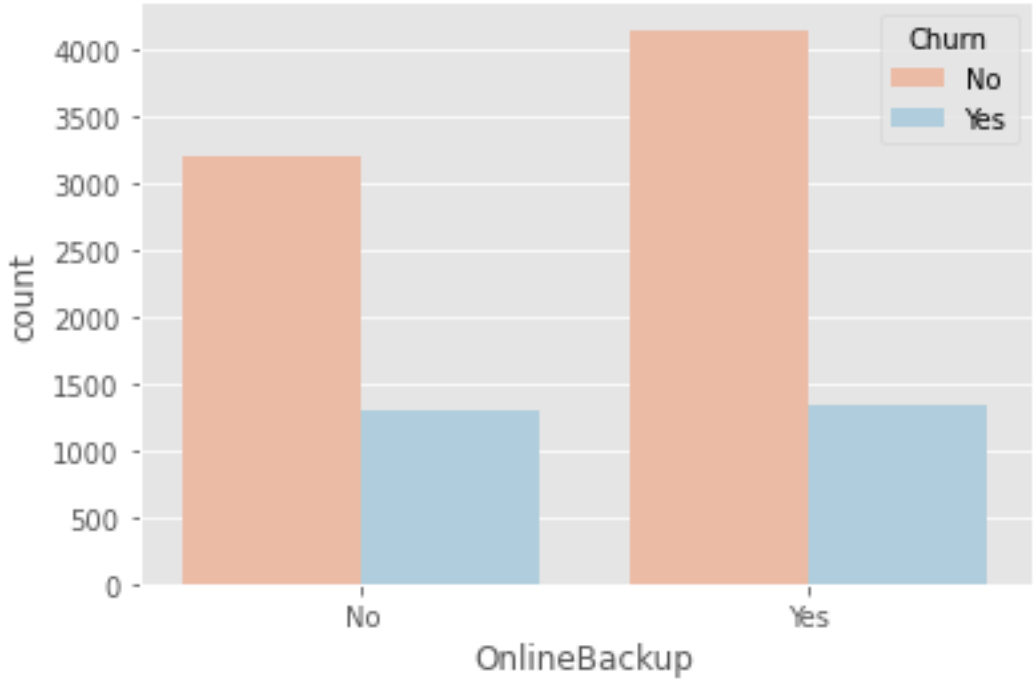
```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='Multiple', hue='Churn', data=churn_clean_df, palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



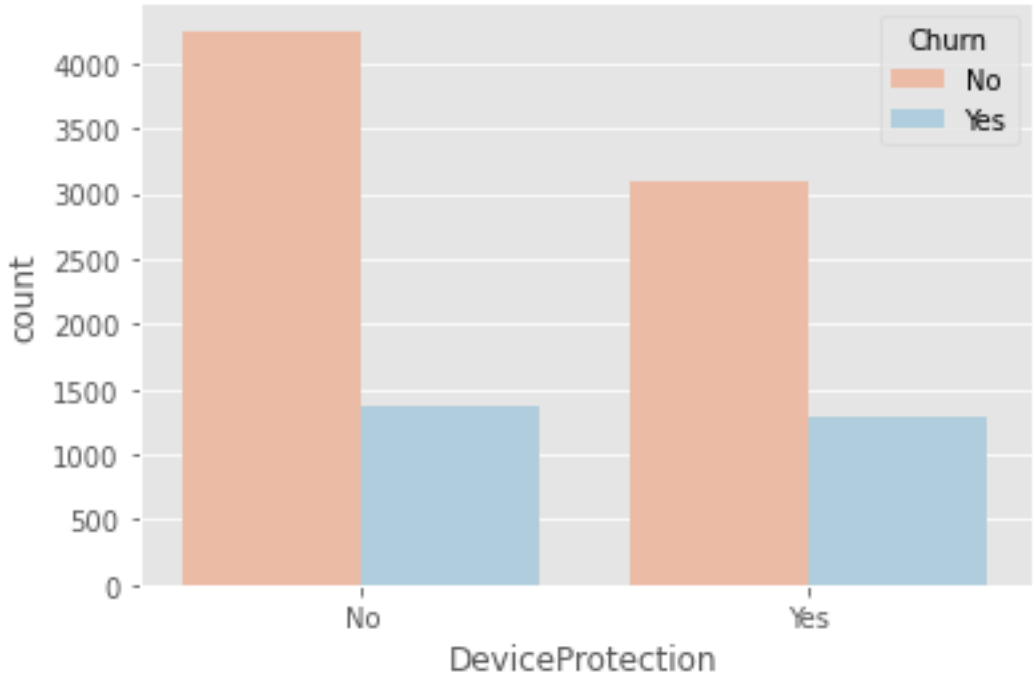

```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='OnlineSecurity', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



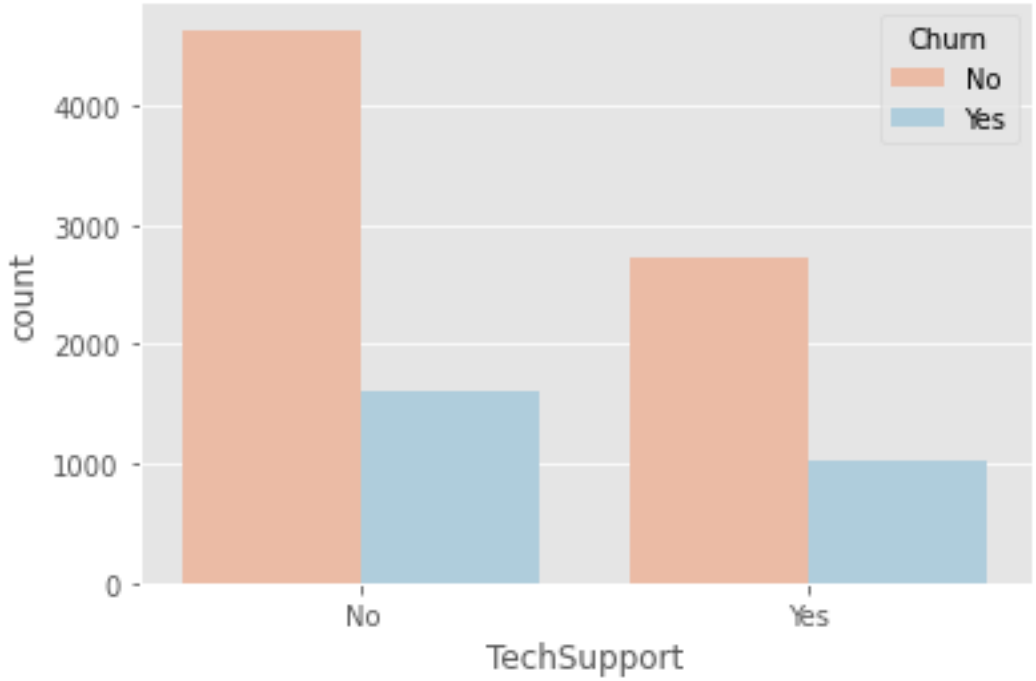

```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='OnlineBackup', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



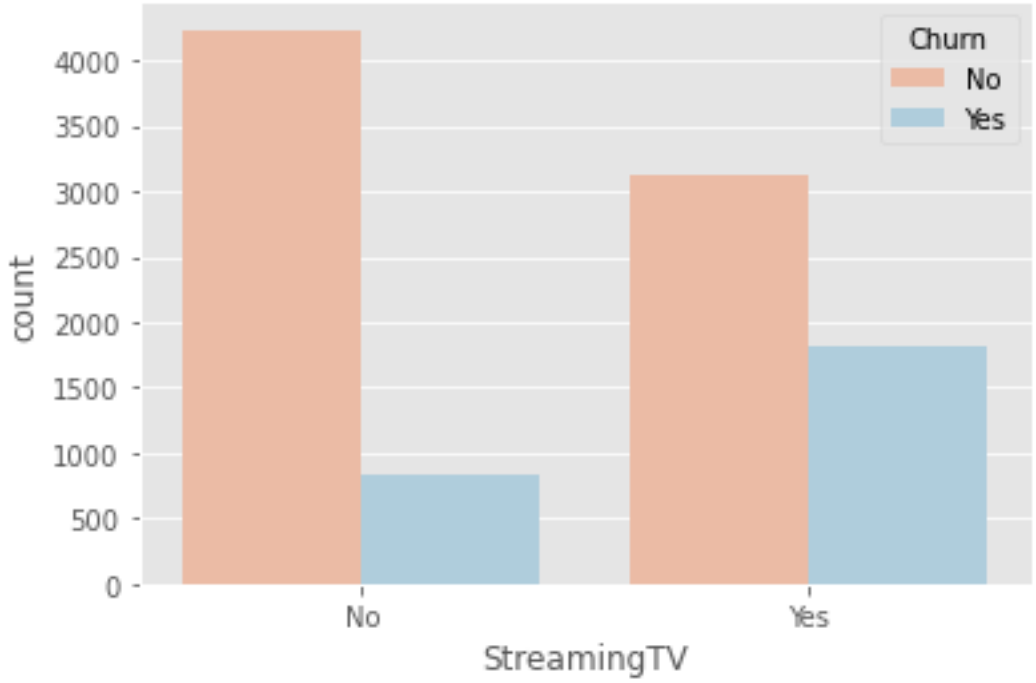
```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='DeviceProtection', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



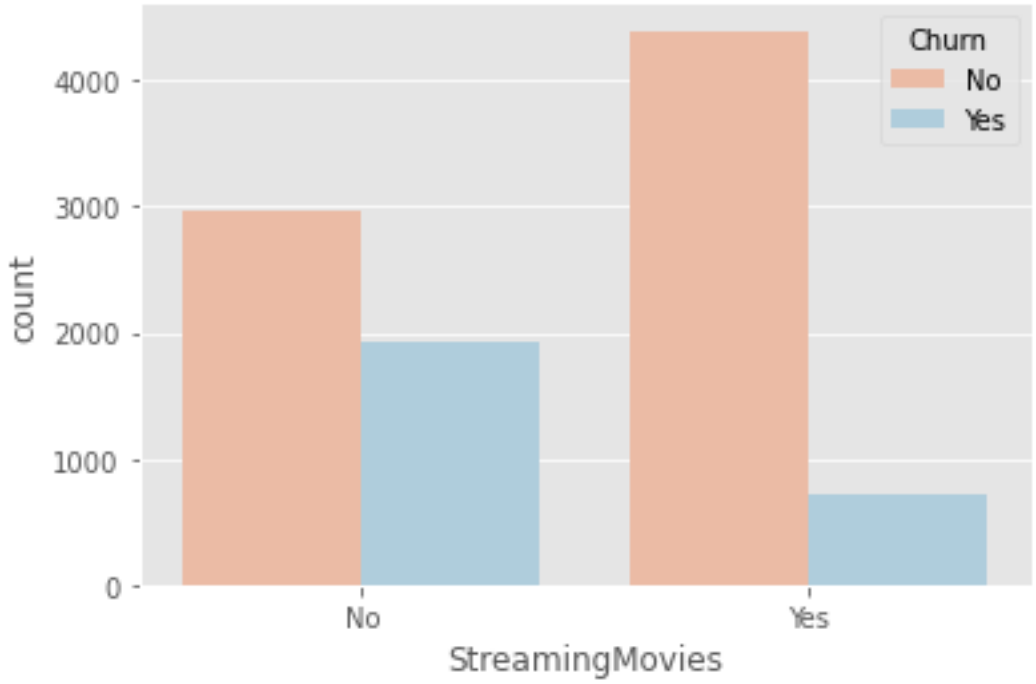

```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='TechSupport', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



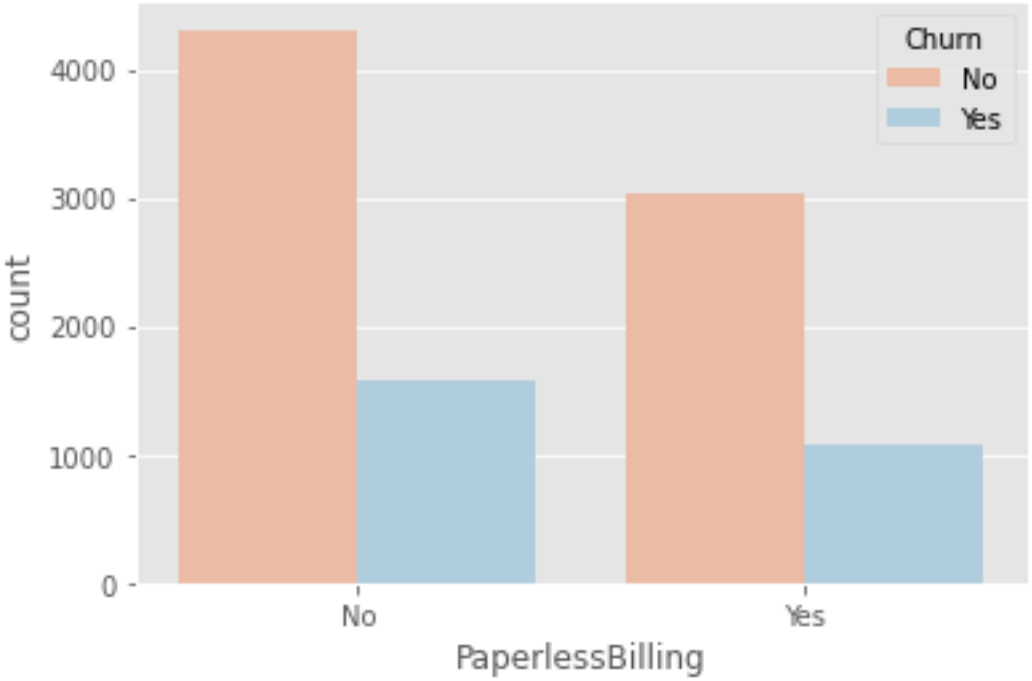

```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='StreamingTV', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```



```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='StreamingMovies', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```




```
plt.style.use('ggplot')

plt.figure()
sns.countplot(x='PaperlessBilling', hue='Churn', data=churn_clean_df,
palette='RdBu')
plt.xticks([0,1], ['No', 'Yes'])
plt.show()
```


Churn
No
Yes
4000
3000
2000
1000
0
count
No
Yes
PaperlessBilling

```
data_nulls = churn_clean_df.isnull().sum()
print(data_nulls)
```

```
CaseOrder                      0
Customer_id                    0
Interaction                    0
UID                            0
City                           0
State                          0
County                         0
Zip                            0
Lat                            0
Lng                            0
Population                     0
Area                           0
TimeZone                       0
Job                            0
Children                       0
Age                            0
Income                         0
Marital                        0
Gender                         0
Churn                          0
Outage_sec_perweek             0
Email                          0
Contacts                       0
Yearly_equip_failure           0
Techie                         0
Contract                       0
Port_modem                     0
Tablet                         0
InternetService                0
Phone                          0
Multiple                       0
OnlineSecurity                 0
OnlineBackup                   0
DeviceProtection               0
TechSupport                    0
StreamingTV                    0
StreamingMovies                0
PaperlessBilling               0
PaymentMethod                  0
Tenure                         0
MonthlyCharge                  0
Bandwidth_GB_Year              0
TimelyResponse                 0
TimelyFixes                    0
TimelyReplacements             0
Reliability                    0
Options                        0
Respectful_Response            0
Courteous_Exchange             0
Evidence_Of_Active_Listening   0
```

```
#Now we have to encode our binary categorical variables with a numerical
value of either 1 or 0. This is so that we are able to properly process this
in our analysis step
churn_clean_df ['DummyGender'] = [1 if v =='Male' else 0 for v in
churn_clean_df['Gender']]
churn_clean_df ['DummyChurn'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['Churn']]
churn_clean_df ['DummyTechie'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['Techie']]
churn_clean_df ['DummyContract'] = [1 if v =='Two Year' else 0 for v in
churn_clean_df['Contract']]
churn_clean_df ['DummyPort_modem'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['Port_modem']]
churn_clean_df ['DummyTablet'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['Tablet']]
churn_clean_df ['DummyInternetService'] = [1 if v =='Fiber Optic' else 0 for
v in churn_clean_df['InternetService']]
churn_clean_df ['DummyPhone'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['Phone']]
churn_clean_df ['DummyMultiple'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['Multiple']]
churn_clean_df ['DummyOnlineSecurity'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['OnlineSecurity']]
churn_clean_df ['DummyOnlineBackup'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['OnlineBackup']]
churn_clean_df ['DummyDeviceProtection'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['DeviceProtection']]
churn_clean_df ['DummyTechSupport'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['TechSupport']]
churn_clean_df ['DummyStreamingTV'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['StreamingTV']]
churn_clean_df ['DummyStreamingMovies'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['StreamingMovies']]
churn_clean_df ['DummyPaperlessBilling'] = [1 if v =='Yes' else 0 for v in
churn_clean_df['PaperlessBilling']]


#Now we will drop the original (Yes/No) categorical variables as we
essentially already created a duplicate of them with binary 0 or 1 values
churn_clean_df = churn_clean_df.drop (columns = ['Gender',
'Churn','Techie','Contract','Port_modem','Tablet','InternetService','Phone','
Multiple','OnlineSecurity','OnlineBackup','DeviceProtection','TechSupport','S
treamingTV','StreamingMovies','PaperlessBilling'])


churn_clean_df.head()


#Remove less significant columns in order to reduce dataset and make it
easier for analysis
```

C4. Copy of Cleaned Data Set

Copy of the Cleaned Data Set saved as 'data_churn_clean_prepared.csv' is attached

Part IV: Analysis

D1. Splitting Data into Training and Testing

```
#Finally we Extract the cleaned data set
churn_clean_df.to_csv('data_churn_clean_prepared.csv')


# Reload the prepared data set for our analysis
churn_df = pd.read_csv('data_churn_clean_prepared.csv')

# Set the predictor feature & target variable
X = churn_df.drop('DummyChurn', axis=1).values
y = churn_df['DummyChurn'].values


# Imports splitting method from sklearn
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV, KFold, cross_val_predict,
train_test_split
from sklearn.metrics import mean_absolute_error as MAE
from sklearn.metrics import mean_squared_error as MSE


#We will now work on splitting the data. First we need to determin a Seed
value as to ensure reproducability.
SEED = 1

# Now we can split the data into our train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,
random_state = SEED)


# Creating the Decision Tree Model
dt = DecisionTreeRegressor(max_depth = 8,
                           min_samples_leaf = 0.1,
                           random_state = SEED)
# Fit dataframe to Decision Tree model
dt.fit(X_train, y_train)

# Compute y_pred
y_pred = dt.predict(X_test)

# Predict outcomes from test set
y_pred = dt.predict(X_test)
```

D2. Analysis Technique and Intermediate Calculations

The analysis Technique used was a decision tree method. This entailed splitting the data into training and testing components, then running a Mean Square Error calculation to judge the accuracy of both our initial model and model calculated via GridSearch Cross Validation. The lower the Mean Square Error, the better our model would be.

```
# Computing the test MSE
mse_dt = MSE(y_test, y_pred)

# Computing the test RMSE
rmse_dt = mse_dt**(1/2)

# Print initial RMSE
print('Initial RMSE score Decison Tree Regressor model:
{:.3f}'.format(rmse_dt))
```

```
Initial RMSE score Decison Tree Regressor model: 0.359
```

D3.Code Execution

```
# Creating Random Random Forest model
rfr = RandomForestRegressor(n_estimators=500, random_state=1)

# Fit model to dataframe
rfr.fit(X_train, y_train)


# Create predictions
train_predictions = rfr.predict(X_train)
test_predictions = rfr.predict(X_test)


# Train/Test Errors
train_error = MAE(y_true=y_train, y_pred=train_predictions)
test_error = MAE(y_true=y_test, y_pred=test_predictions)


# Print the accuracy for seen and unseen data
print("Model error on seen data: {0:.2f}.".format(train_error))
print("Model error on unseen data: {0:.2f}.".format(test_error))


# Print how significant each column is to the model
for i, item in enumerate(rfr.feature_importances_):
    print('{0:s}: {1:.2f}'.format(churn_df.columns[i], item))


Unnamed: 0: 0.05
Children: 0.02
Age: 0.03
Income: 0.04
Outage_sec_perweek: 0.04
Email: 0.03
Contacts: 0.01
Yearly_equip_failure: 0.01
Tenure: 0.28
MonthlyCharge: 0.23
Bandwidth_GB_Year: 0.04
TimelyResponse: 0.01
TimelyFixes: 0.01
TimelyReplacements: 0.01
Reliability: 0.01
Options: 0.01
Respectful_Response: 0.01
Courteous_Exchange: 0.01
Evidence_Of_Active_Listening: 0.01
DummyGender: 0.00
DummyChurn: 0.01
DummyTechie: 0.04
DummyContract: 0.00
DummyPort_modem: 0.00
DummyTablet: 0.03
DummyInternetService: 0.00
```

Part V: Data Summary and Implications

E1. Accuracy and MSE of the Predictive Model

We see below that our MSE of the Decision Tree Model is 0.129. This is a low Mean Square Error number and shows us that there is actually very little variation between our predicted values and actual values. This then implies also that we have a high degree of accuracy with our current predictive model

```
# Import our cross validation metrics
from sklearn.model_selection import cross_val_score

# Computing the coefficient of (R-squared)
scores = cross_val_score(rfr, X, y, scoring='r2')


# Print R-squared value
print('Cross validation R-squared values: ', scores)


from sklearn.metrics import mean_squared_error as MSE
print('Using scikit-lean, the Mean Squared Error: {:.3f}'.format(MSE(y_test,
y_pred)))
```

```
Using scikit-lean, the Mean Squared Error: 0.129
```

E2. Results and Implications

We found that the accuracy of our model using our decision tree method produced a MSE score of 0.129. This is a very good result because with the low MSE, our model's predicted values are very close to the actual values. Therefore, our model has great reliability and can be utilized for decision making by shareholders with confidence

E3. Limitations

One limitation to the analysis is the issue of overfitting. If we are to take previous data into consideration, then this would make the model underperform when seeing new samples or data for the first time. The solution to this is to remove our insignificant variables/branches that have features of low importance as calculated above. Doing this would only leave the most important features and give us a better understanding of the model and more accurate prediction, without being affected by overfitting to insignificant features.

E4. Recommended Course of Action

First it is important to note that our model having a MSE of 0.129 is a great result in terms of prediction capabilities. This should give confidence in the shareholders to trust the insights of the model. It is shown that as more customers opt in for services such as OnlineBackup, Online Security, etc… they are less likely to leave. Therefore, the company and decision makers should focus on providing customers with more services to build more of a brand loyalty and not just rely on the basics of a telecommunication service. This does make sense since other competitors will be offering similar options, and so to differentiate from the rest, companies must come up with more services that are beneficial to their current user database.

Part VI: Demonstration

F. Panapto Video

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=b9e61e67-eb84-4c3c-8ff7-ada00036dc74

G. Third Party Code

Cory Maklin (2019). Decision Tree In Python.
https://towardsdatascience.com/decision-tree-in-python-b433ae57fb93

Julien Beaulieu (2021). Decision Tree.
https://julienbeaulieu.gitbook.io/wiki/sciences/machine-learning/decision-trees

PennState (2021). Cross-Validation Tutorial.
https://quantdev.ssri.psu.edu/tutorials/cross-validation-tutorial

H. Acknowledged Sources

Deepankar (2021). Decision Tree With CART Algorithm.
https://medium.com/geekculture/decision-trees-with-cart-algorithm-7e179acee8ff

Nagesh Chauhan (2020). Decision Tree Algorithm Explained.
https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html