WGU D209 PA 1

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 K-Nearest Neighbours Technique.

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 Classification Method using K-Nearest Neighbors

The K-Nearest Neighbors technique uses our already labeled data to gain insights into classifying any new data obtained. It does this by analyzing the distances between the new query data and all other data points in our dataset to then vote on the most frequent classification type based on the surrounding classifications and the number of K selected by the model **(Onel Harrison 2018)**

B2. Summary of One Assumption for K-Nearest Neighbors

The assumption of importance is that data that are close together in proximity are typically similar and therefore we can take our knowledge of one point in or dataset that is known, and draw accurate conclusions for classification on a new data that is unknown **(Peter Grant 2019).**

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)

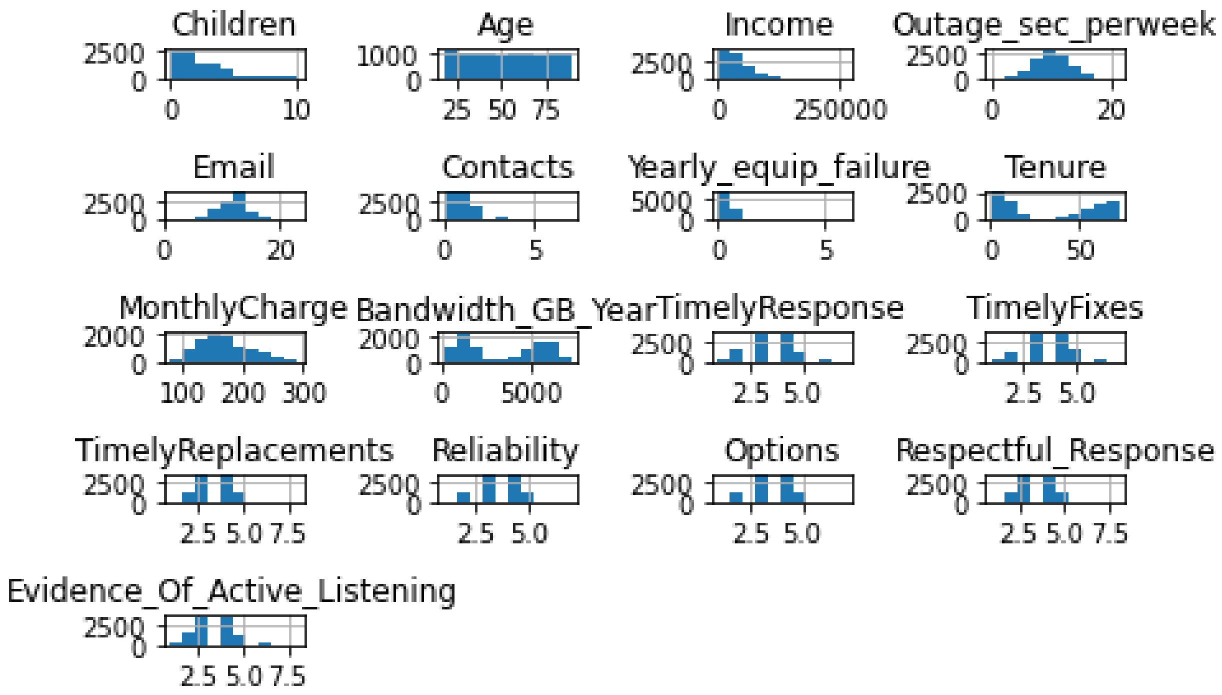
10) DeviceProtection (Yes/No)
11) TechSupport (Yes/No)
12) StreamingTV (Yes/No)
13) StreamingMovies (Yes/No)

Initial Data Set Discrete Numerical Variables:

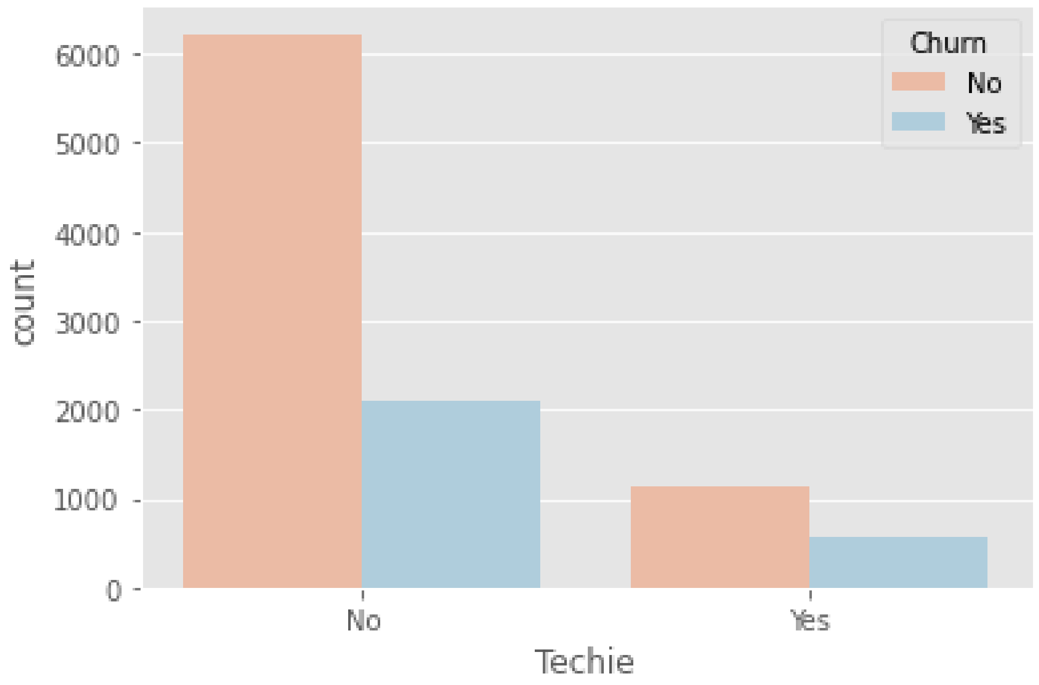
1) Item1: Timely Response
2) Item2: Timely Fixes
3) Item3: Timely Replacements
4) Item4: Reliability
5) Item5: Options
6) Item6: Respectful Response
7) Item7: Courteous Exchange
8) Item8: Evidence of Active Listening

C3. Data preparation code

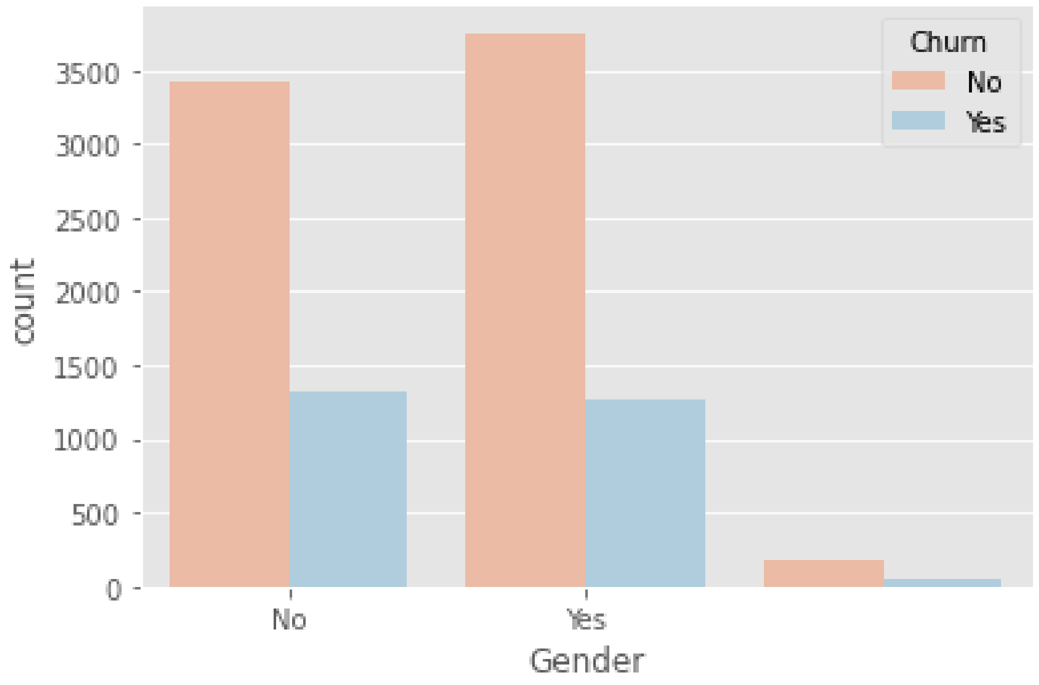
```
#Importing the standard Python libraries for Analysis and Visualizations
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

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()
```



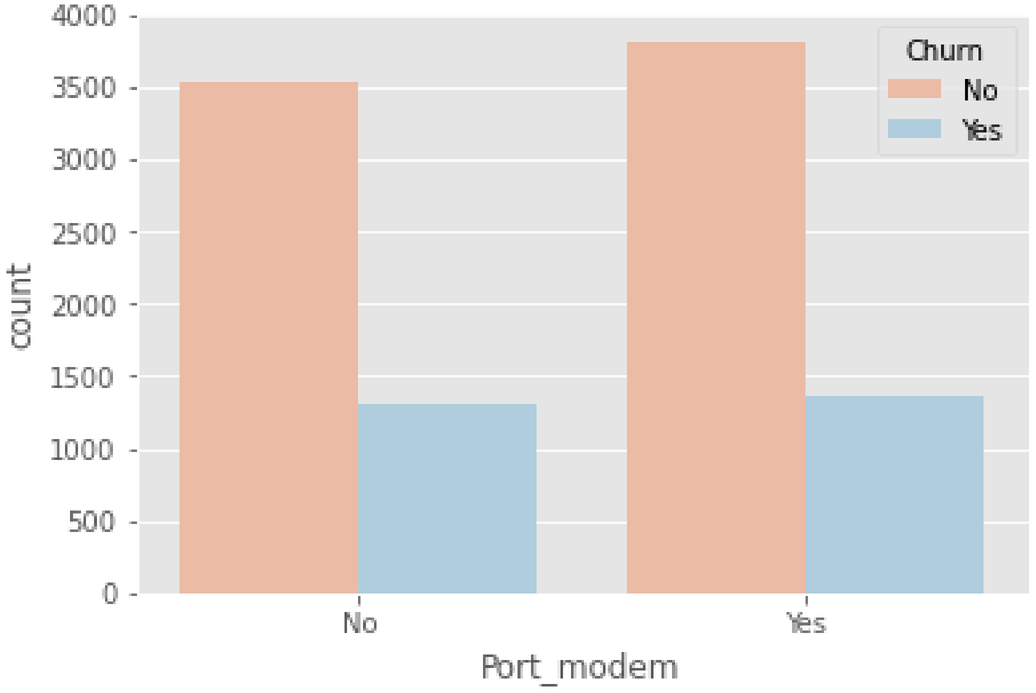
```
#Now using ggplot, we can also see the bivariate visualizations between our
categorical binary variables and that of 'Churn'
plt.style.use('ggplot')

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



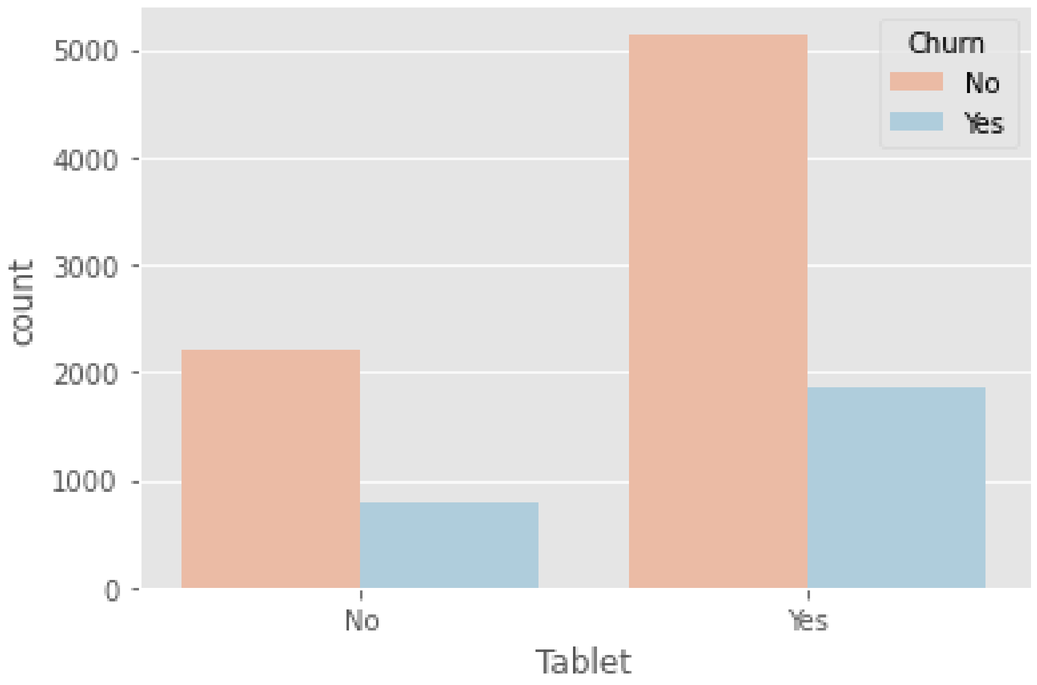
```
plt.style.use('ggplot')

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



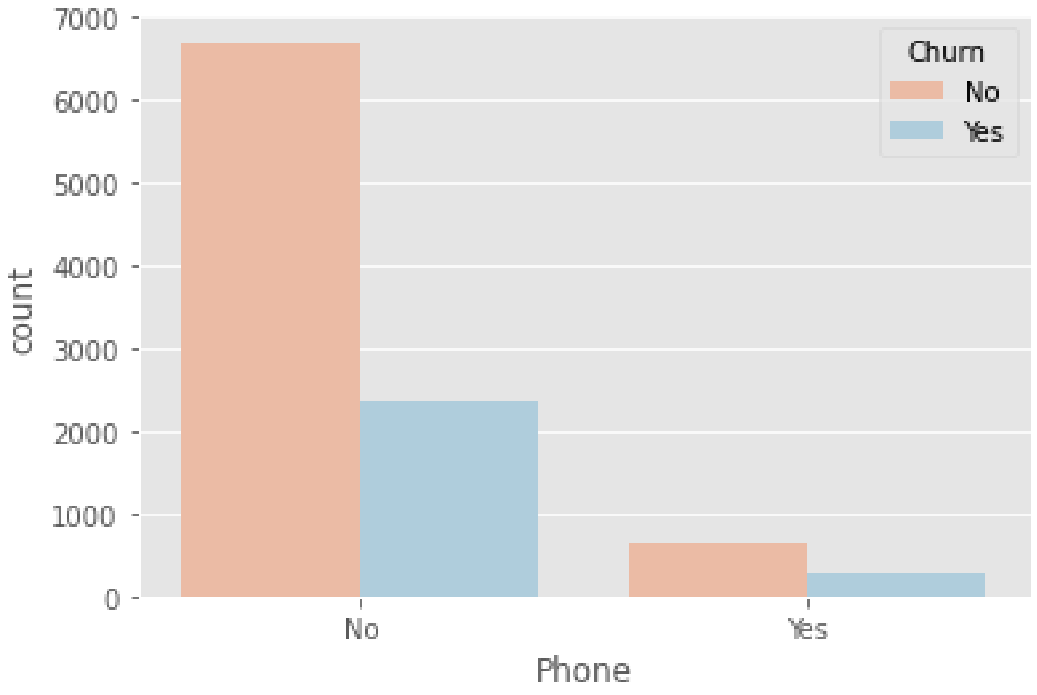
```
plt.style.use('ggplot')

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



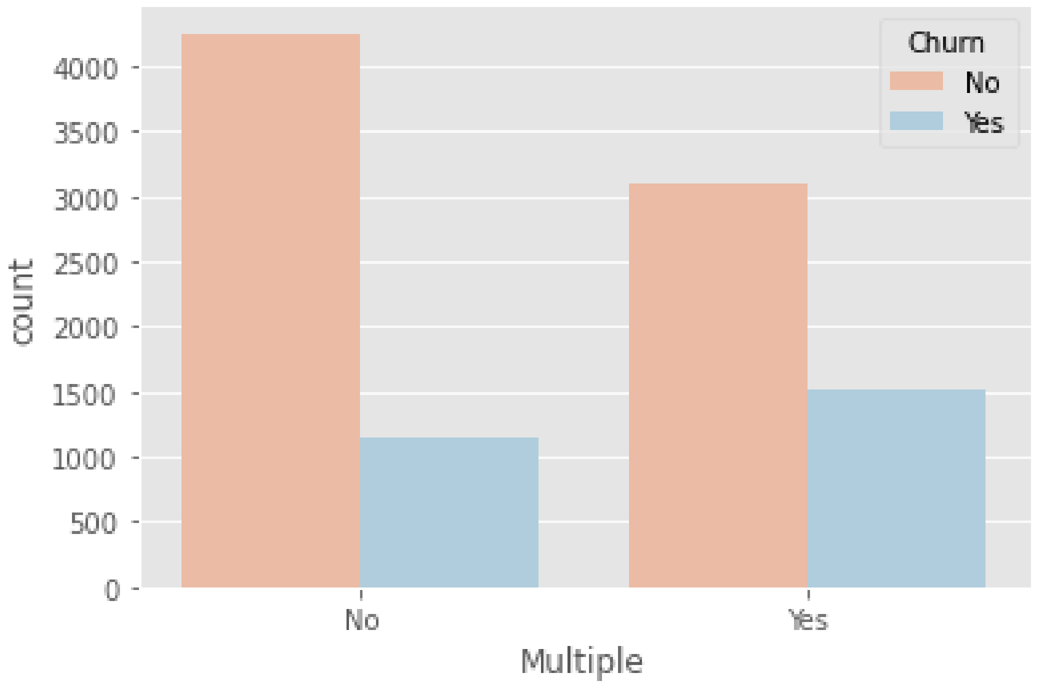
```
plt.style.use('ggplot')

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



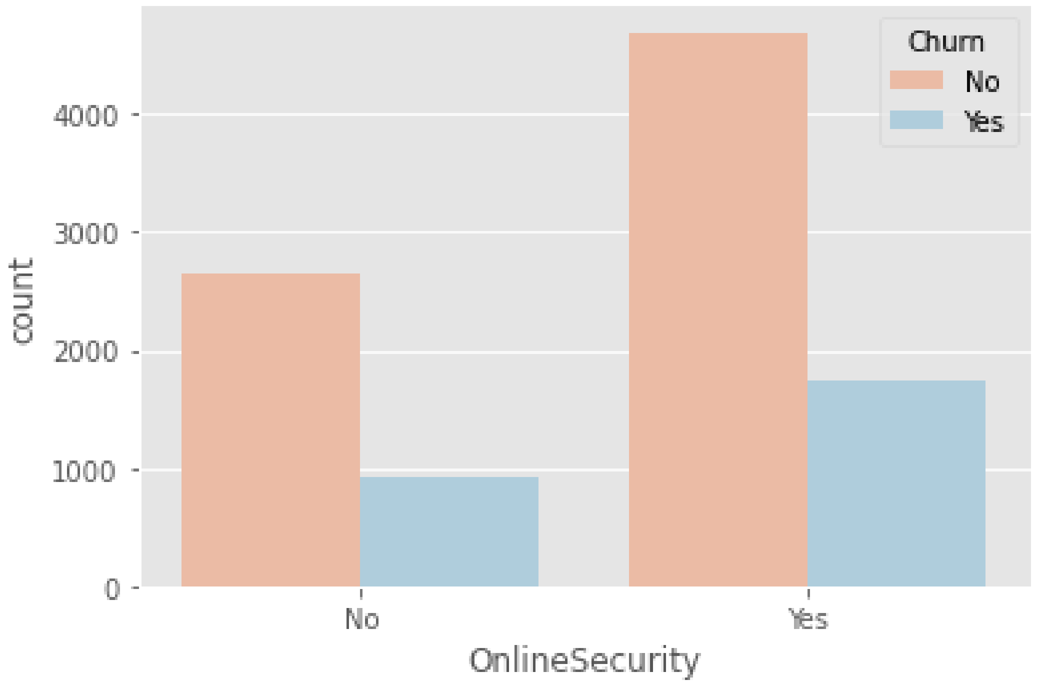
```
plt.style.use('ggplot')

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



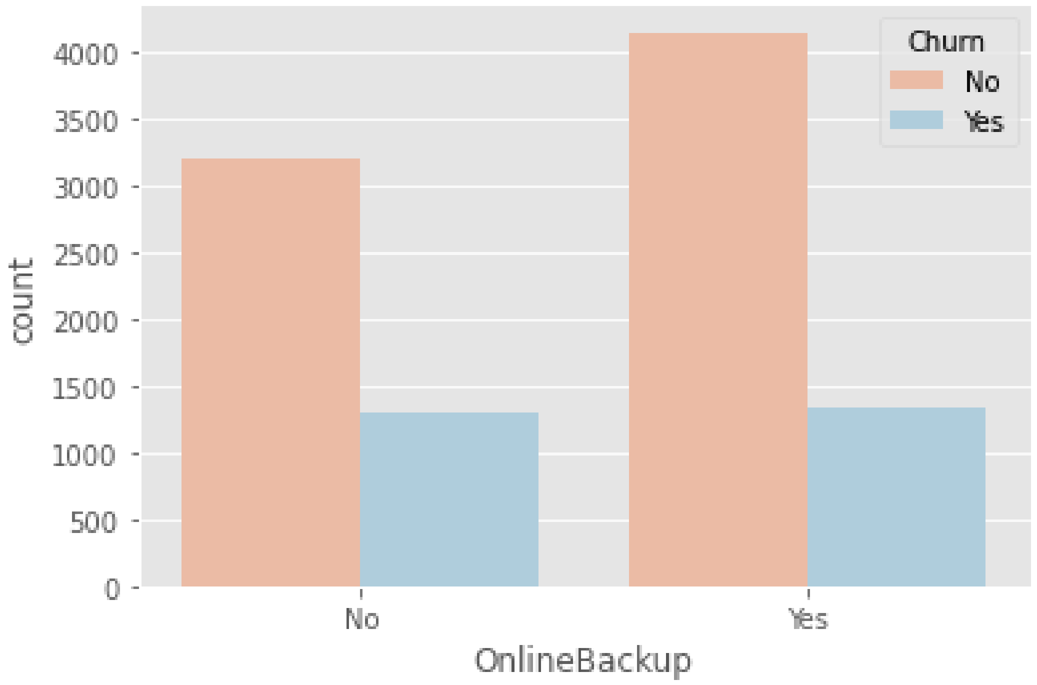
```
plt.style.use('ggplot')

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



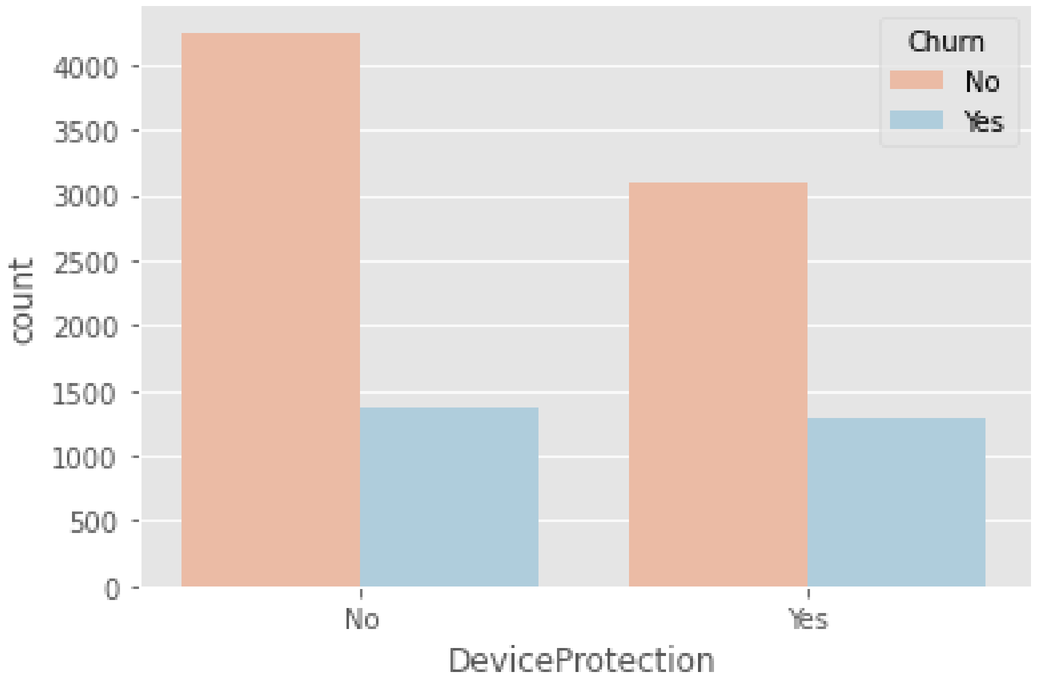
```
plt.style.use('ggplot')

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



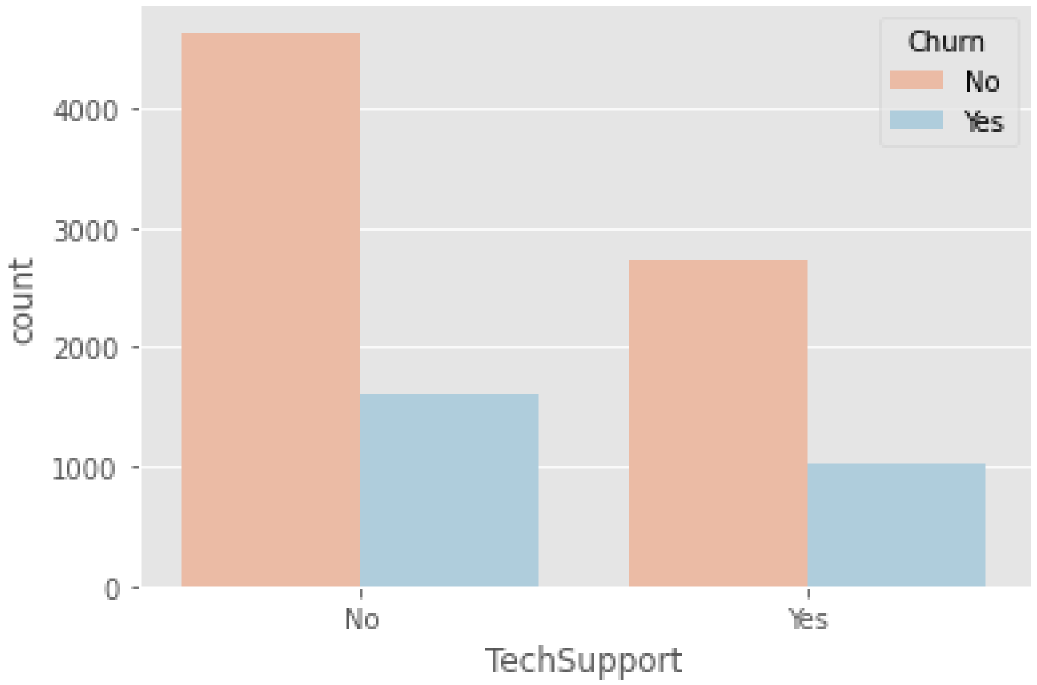
```
plt.style.use('ggplot')

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



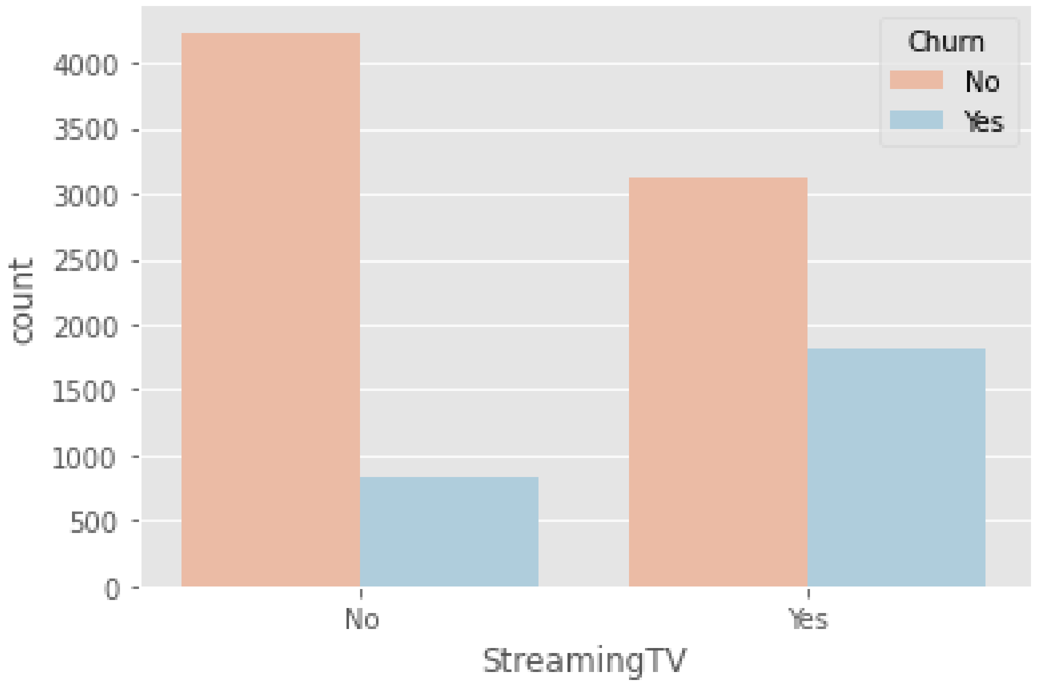
```
plt.style.use('ggplot')

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



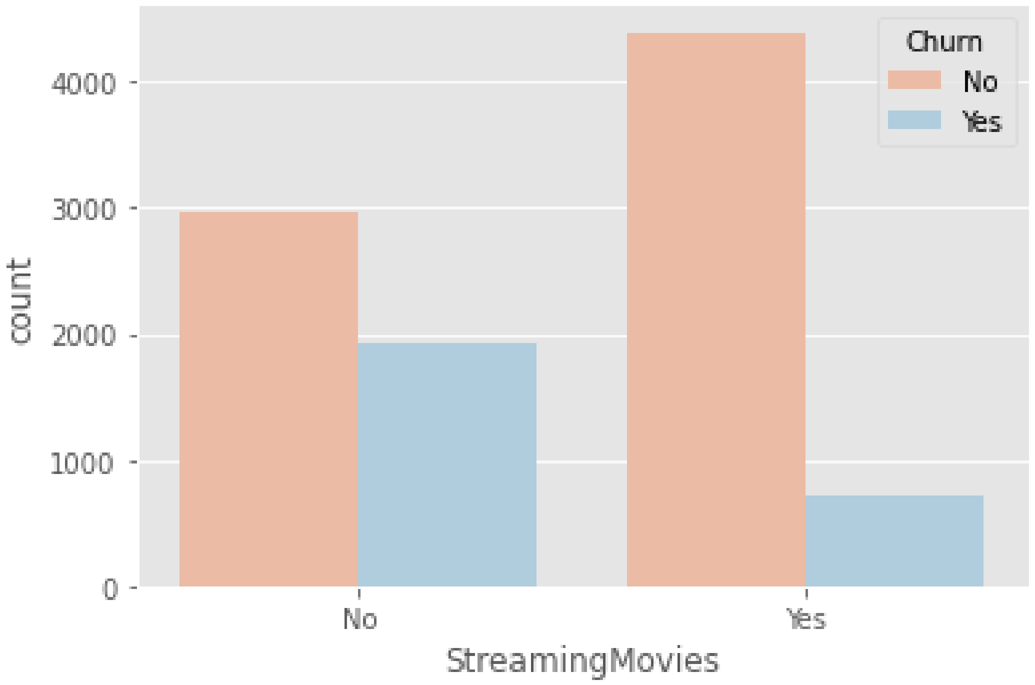
```
plt.style.use('ggplot')

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



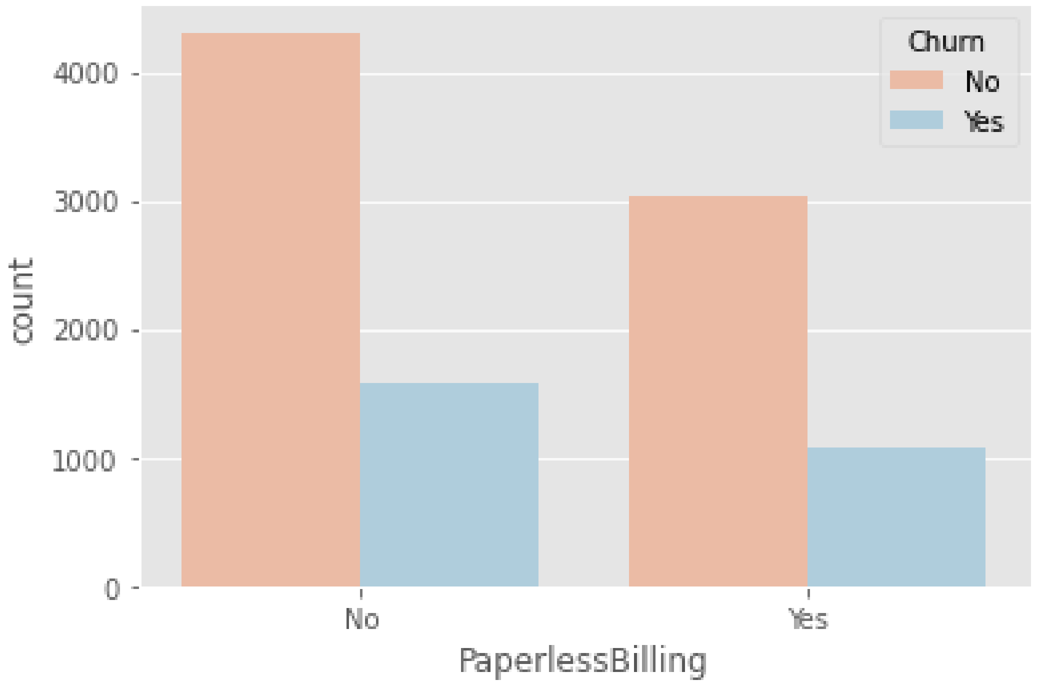
```
plt.style.use('ggplot')

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()
```



```
#Now we will see if there are any anomolies in the form of missing data in the
given data set before applying it to the analysis process
```

```
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
```

C3. 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

```
# 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


#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)


# Instantiate our KNN model by using n nearest neighbors value of 7
knn = KNeighborsClassifier(n_neighbors = 7)

# Fit our data to KNN model
knn.fit(X_train, y_train)

# Now we predict from our train test split
y_pred = knn.predict(X_test)
```

D2. Analysis Technique and Intermediate Calculations

Analysis technique used was the K-Nearest Neighbors method. First, through our sklearn library, we scaled the data in order to get rid of much the variance. Then we pipelined the split data into train/test split. Finally, we created an accuracy score of our KNN model along with a classification report and a confusion matrix of the outputs.

```
# Print our initial accuracy score of the KNN Model
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score, train_test_split

print('Initial accuracy score KNN model: ', accuracy_score(y_test, y_pred))
```

```
Initial accuracy score KNN model:  0.7145
```

```
# lets now run our Classification Report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.83      0.81      1442
           1       0.49      0.40      0.44       558

    accuracy                           0.71      2000
   macro avg       0.63      0.62      0.62      2000
weighted avg       0.70      0.71      0.71      2000
```

D3. Code for the Classification Analysis

```
# It is important to scale our data and create a pipeline object
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score

# pipeline object steps
steps = [('scaler', StandardScaler()),
         ('knn', KNeighborsClassifier())]

# Instantiate pipeline
pipeline = Pipeline(steps)

# Split our dataframe
X_train_scaled, X_test_scaled, y_train_scaled, y_test_scaled =
train_test_split(X, y, test_size = 0.2, random_state = SEED)

# Scale our dateframe with the pipeline object
knn_scaled = pipeline.fit(X_train_scaled, y_train_scaled)

# Predict from the scaled dataframe
y_pred_scaled = pipeline.predict(X_test_scaled)


# Lets now take a look at the new accuracy score of our scaled KNN Model
print('New accuracy score of scaled KNN model:
{:0.3f}'.format(accuracy_score(y_test_scaled, y_pred_scaled)))
```

```
New accuracy score of scaled KNN model: 0.815
```

```
# Compute classification metrics after scaling
print(classification_report(y_test_scaled, y_pred_scaled))
```

```
              precision    recall  f1-score   support

           0       0.85      0.90      0.87      1442
           1       0.70      0.60      0.64       558

    accuracy                           0.81      2000
   macro avg       0.77      0.75      0.76      2000
weighted avg       0.81      0.81      0.81      2000
```

```
#Create a confusion matrix
from sklearn.metrics import confusion_matrix
cf_matrix = confusion_matrix(y_test, y_pred)
print(cf_matrix)
```

```
[[1204  238]
 [ 333  225]]
```

Part V: Data Summary and Implications

E1. Accuracy and Area Under the Curve of the Classification Model

We see above that scaling our data improves both our model's accuracy and precision. Accuracy increases from 0.71 to 0.81 while Precision increases from 0.78 to 0.85. The area under the curve or (AUC) is found to be 0.79.

```
#Create our Cross Validation Model via GridSearchCV
from sklearn.model_selection import GridSearchCV

# grid parameters
param_grid = {'n_neighbors': np.arange(1, 50)}

#  KNN for cross validation
knn = KNeighborsClassifier()

# Instantiate GridSearch cross validation
knn_cv = GridSearchCV(knn , param_grid, cv=5)

# Fit model to
knn_cv.fit(X_train, y_train)

# Print best parameters
print('Best parameters for this KNN model: {}'.format(knn_cv.best_params_))


Best parameters for this KNN model: {'n_neighbors': 6}


# Create the best model for the score
print('Best score for this KNN model: {:.3f}'.format(knn_cv.best_score_))


Best score for this KNN model: 0.735


# Import ROC AUC metrics for the area under the curve
from sklearn.metrics import roc_auc_score

# Fit it to the data
knn_cv.fit(X, y)

# Compute predicted probabilities: y_pred_prob
y_pred_prob = knn_cv.predict_proba(X_test)[:,1]

# Compute and print AUC score
print("The Area under curve (AUC) on validation dataset is:
{:.4f}".format(roc_auc_score(y_test, y_pred_prob)))


The Area under curve (AUC) on validation dataset is: 0.7959
```

E2. Results and Implications

We found that accuracy of our model using our chosen value of k = 7 has a score of 0.84 after scaling. This is not too great (especially after scaling) as we would have liked the accuracy to be atleast 90-100 as a form of reliability to classify correctly. With our technique of cross validation grid search, we see that our optimal value of k should have been 6. This would have lead to an AUC value of 0.79, meaning that it has a decent ability to distinguish the classes.

E3. Limitations

One limitation shown in this analysis is that it is important which value of k you decide to choose for your analysis. We have the ability to 7choose any k value we want, however there are optimal k values to use and others that are not, which will produce vastly different results. We decided to choose the k value of 7 for our model which had decent results, however when using the cross-validation gird search technique, we see that a k value of 6 might have been a better choice.

E4. Recommended Course of Action

First it is important to note that while our scaled data gives us a accuracy of 0.84, this is still not considered super reliable since out of 100 cases, there is a inaccurate prediction of classification 16 times. That being said, if we analyze the variables that are present in most cases that result to churn and reduce them, then this will provide us a better solution to retain customers. 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 have to 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=b00ca9a0-9593-4d68-a6b2-ad9900596aaf

G. Third Party Code

Jason Brownlee (2020). Develop K-Nearest Neighbors in Python from Scratch.
https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/

SK-Learn.Org (2020). Parameter Estimation Using Grid Search With Cross-Validation.
https://scikit-learn.org/stable/auto_examples/model_selection/plot_grid_search_digits.html

H. Acknowledged Sources

Zendesk (2021). How To Calculate Customer Churn Rate.
https://www.zendesk.com/blog/customer-churn-rate/

Peter Grant (2019). Introducing K-Nearest Neighbors.
https://towardsdatascience.com/introducing-k-nearest-neighbors-7bcd10f938c5

Onel Harrison (2018). Machine Learning Basics With The K-Nearest Neighbors Algorithm.
https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761