Performance Assessment for

D209: Data Mining I

Task 1 Attempt 2

Drew Mendez

MSDA Western Governors University

D209: Data Mining I

Dr. Eric Straw

August 21, 2024

D209_PA_MendezD_Task1_Attempt2

August 21, 2024

1 Part I: Research Question

1.1 A. Purpose of Data Mining Report

1.1.1 A1. Research Question

Can the k-Nearest Neighbor method be used to predict whether or not a customer is at risk of churn?

1.1.2 A2. Goal of the Data Analysis

The primary goal of this analysis is to develop a machine learning model by applying the k-Nearest Neighbor method to identify the features associated with churn. The results of this analysis will be used to inform a recommended course of action.

2 Part II: Method Justification

2.1 B. Reasons for Chosen Classification Method

2.1.1 B1. How the kNN method Analyzes the Data and Expected Outcomes

The k-Nearest Neighbors method identifies the k closest data points (or neighbors) to a new data point based on a chosen distance metric, such as Euclidean distance. kNN can be used to predict the class of a new data point based on the majority class among its k neighbors, and as such, it can be used for classification tasks such as identifying whether or not a customer will churn given a particular set of features (Elleh).

2.1.2 B2. Assumptions of the kNN Method

- The k-nearest neighbors classification method assumes that similar things exist in proximity to each other.
- If a data point is far away from another group, it is dissimilar to those data points.
- The algorithm depends on this assumption being true enough for the algorithm to be useful.
- The algorithm classifies new data points based on how the neighbors are classified.

(Elleh)

2.1.3 B3. Packages and Libraries Used to Support Analysis

Packages/Libraries	Method/Function	Usage
Pandas	`.isnull`, `.duplicated`, and `.sum`	used to provide important basic functionality
Pandas	`.quantile`	used in outlier detection
matplotlib.pyplot	`title` and `show`	used to generate figures
Seaborn	`boxplot`	used to observe the distributions of quantitative variables
sklearn.preprocessing	`MinMaxScaler`	used to scale the data to the 0-1 range
sklearn.feature_selection	`SelectKBest`, `f_classif`	used to determine the best features for the reduced model
sklearn.model_selection	`train_test_split`	used to split data into test set and training set
sklearn.neighbors	`KNeighborsClassifier`	main tool of the analysis
sklearn.model_selection	`GridSearchCV`	used to determine the best value of k
sklearn.metrics	`confusion_matrix`	used to generate the confusion matrix for the model
sklearn.metrics	`roc_auc_score`	used to complete the Area Under the Curve score for the model
sklearn.metrics	`roc_curve`	used to plot the Receiver Operating Characteristic curve of the model
sklearn.metrics	`classification_report`	used to generate a summary report of the metrics of the model

3 Part III: Data Preparation

3.1 C. Data Preparation

3.1.1 C1. One Data Preprocessing Goal

A crucial goal of data preprocessing is the re-expression of categorical variables using one-hot encoding, a process by which categorical data is re-encoded using 0 and 1 in newly created columns. For example, the categorical variables `Gender` has three levels: Male, Female, and Nonbinary. Applying one-hot encoding to the `Gender` variable creates three new binary features, with each row having a 1 in the column corresponding to its original category and a 0 in the other columns:

Original Category	Male	Female	Nonbinary
Male	1	0	0
Female	0	1	0
Nonbinary	0	0	1

This can be achieved using the Pandas function `get_dummies()`. One-hot encoding of the categorical variables enables the kNN classifer to handle data that would otherwise be unusable in this analysis.

3.1.2 C2. Initial Data Set Variables

The dependent variable for this analysis will be the binary categorical variable `Churn`.

The initial explanatory variables used to perform the analysis are:

- all thirteen numeric variables
  - `Lat`, `Lng`, `Population`, `Children`, `Age`, `Income`, `Outage_sec_perweek`, `Email`, `Contacts`, `Yearly_equip_failure`, `Tenure`, `MonthlyCharge`, `Bandwidth_GB_Year`
- all twelve re-expressed binary categorical variables
  - `Techie`, `Port_modem`, `Tablet`, `Phone`, `Multiple`, `OnlineSecurity`, `OnlineBackup`, `DeviceProtection`, `TechSupport`, `StreamingTV`, `StreamingMovies`, `PaperlessBilling`
- all six categorical variables re-expressed with one-hot encoding
  - `Area`, `Marital`, `Gender`, `Contract`, `InternetService`, `PaymentMethod`
- all eight ordinal categorical variables
  - `Item1`, `Item2`, `Item3`, `Item4`, `Item5`, `Item6`, `Item7`, `Item8`

3.1.3 C3. Explanation of Each Step to Prepare the Data

Much of the code to prepare the data was adapted from my D208 Task 2 PA. The steps to prepare the data are:

- detect duplicates, missing values, and outliers
- treatment of NAs and outliers (retention of reasonable outliers)
- binary encoding re-expression of thirteen binary variables
- type casting thirteen binary variables
- one-hot encoding re-expression of six categorical variables
- scaling data to within 0 - 1 so every feature is on the same scale
- identify the best features using `SelectKBest`

[1]:
```
## The following packages were necessary for this analysis:

import pandas as pd # .read_csv(), .duplicated(), .sum(), .isnull(),
import numpy as np # quantile()
import matplotlib.pyplot as plt
import seaborn as sns # boxplot()
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.model_selection import train_test_split
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
```

```
from sklearn.metrics import classification_report
```

[2]:
```
## C3 The following cells include the annotated code used to prepare the data.
# See code attached, in D209_PA_MendezD_Task1_Attempt2.ipynb

# Load the data into a data frame with Pandas' .read_csv() function
df = pd.read_csv('/Users/drewmendez/Documents/WGU/D209/churn_d209/churn_clean.
 ↪csv')



def printDupesNulls(data_frame):
# Detect duplicates with Pandas' .duplicated method chained with .sum() method.
# Identify missing values in the data frame with Pandas' .isnull() method,
# then sum the resulting series with the .sum() method

    duplicate_count = data_frame.duplicated().sum()
    missing_values_count = data_frame.isnull().sum()
    print('Number of duplicate rows:', duplicate_count)
    print("Number of missing values per variable:")
    print(missing_values_count)


def boxplotOutliers(data_frame, col_name):
# Visualize outliers using boxplot() from matplotlib
# First and third quartiles, Q1 and Q3, are found using .quantile() from Pandas,
# then the interquartile range is found using IQR = Q3 - Q1.
# The upper whisker of the boxplot is found using max = Q3 + 1.5 * IQR.
# The lower whisker of the boxplot is found using min = Q1 - 1.5 * IQR.
# The .sum() method returns the count of observations greater than the max or␣
 ↪less than the min.
# The .round() method rounds the outlier count to two decimals.
# If loop to print corresponding outputs

    sns.boxplot(data = data_frame, x = col_name)
    plt.title(f'Boxplot of {col_name}')
    plt.show()

    Q1 = data_frame[col_name].quantile(0.25)
    Q3 = data_frame[col_name].quantile(0.75)
    IQR = Q3 - Q1
    maximum = round(Q3 + 1.5 * IQR, 2)
    minimum = round(Q1 - 1.5 * IQR, 2)
    outlier_count_up = (data_frame[col_name] > maximum).sum()
    outlier_count_low = (data_frame[col_name] < minimum).sum()


    if outlier_count_up > 0:
```

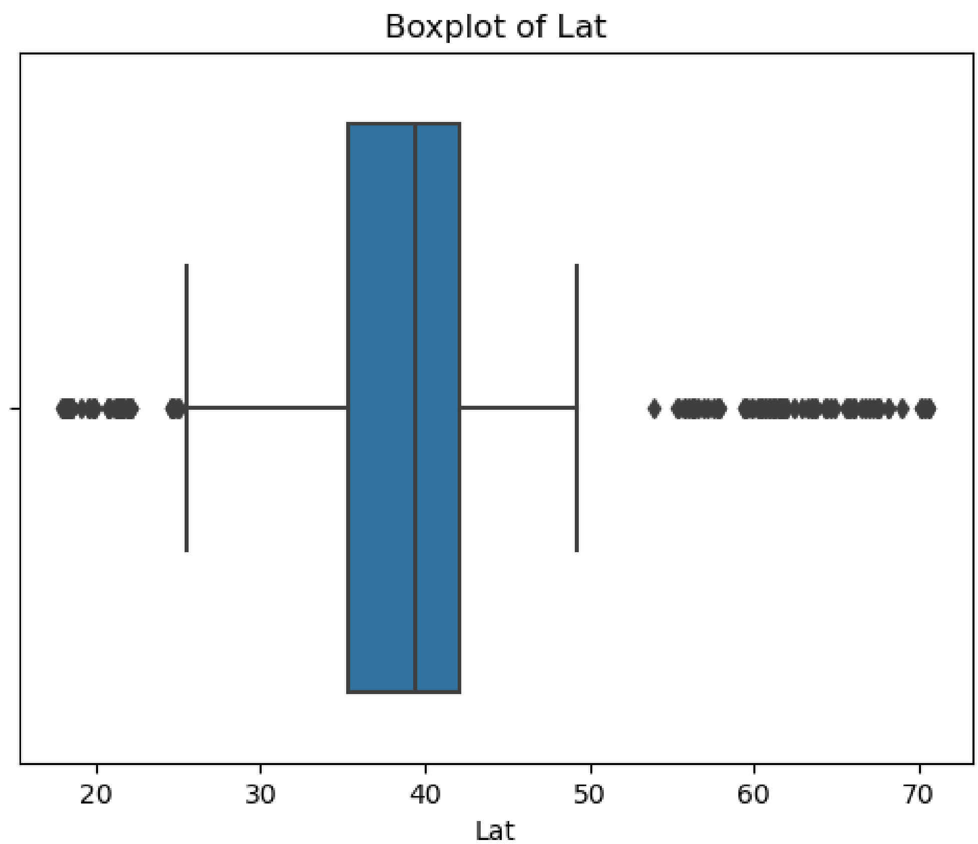
```
        if outlier_count_low > 0:
            print(f'For the `{col_name}` variable, all observations greater␣
↪than {maximum} or less than {minimum} are considered outliers.')
            print(f'The count of observations greater than {maximum} is␣
↪{outlier_count_up}.')
            print(f'The count of observations less than {minimum} is␣
↪{outlier_count_low}.')
        if outlier_count_low == 0:
            print(f'For the `{col_name}` variable, all observations greater␣
↪than {maximum} are considered outliers.')
            print(f'The count of observations greater than {maximum} is␣
↪{outlier_count_up}.')
    if outlier_count_up == 0:
        if outlier_count_low > 0:
            print(f'For the `{col_name}` variable, all observations less than␣
↪{minimum} are considered outliers.')
            print(f'The count of observations less than {minimum} is␣
↪{outlier_count_low}.')
        if outlier_count_low == 0:
            print(f'There are no outliers for the `{col_name}` variable.')
```

[3]:
```
## C3 Detection of Duplicates and Missing Values

printDupesNulls(df)
```

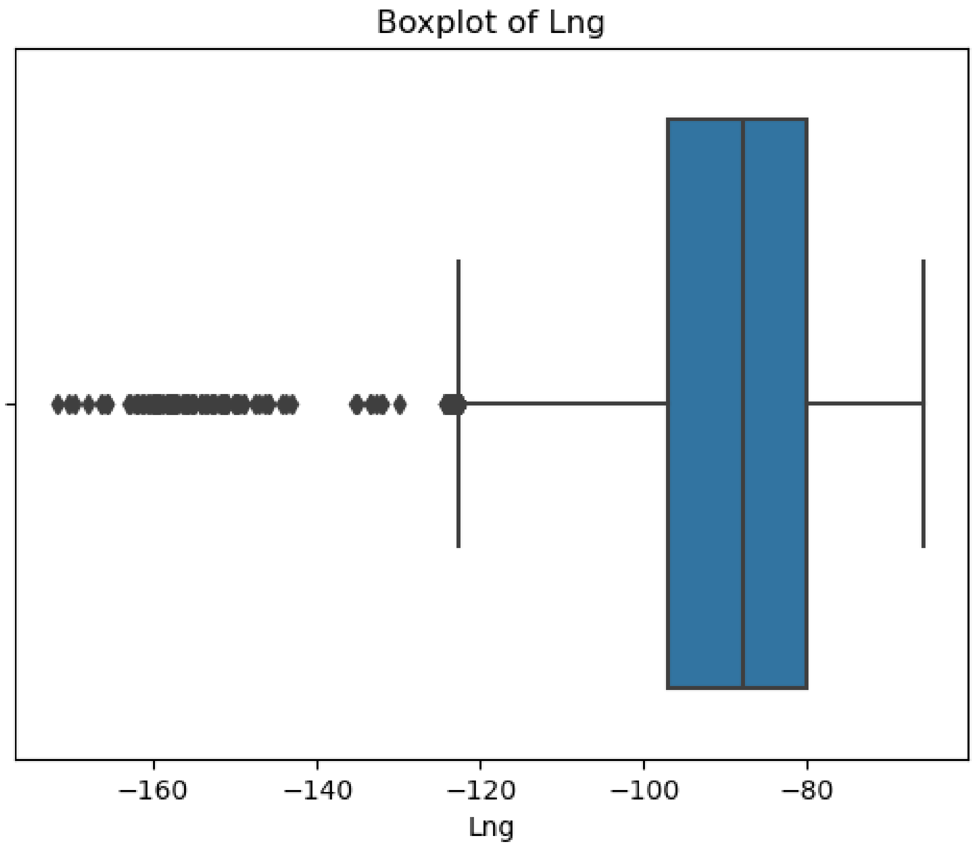
```
Number of duplicate rows: 0
Number of missing values per variable:
CaseOrder                   0
Customer_id                 0
Interaction                 0
UID                         0
City                        0
State                       0
County                      0
Zip                         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       2129
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
Item1                    0
Item2                    0
Item3                    0
Item4                    0
Item5                    0
Item6                    0
Item7                    0
Item8                    0
dtype: int64
```

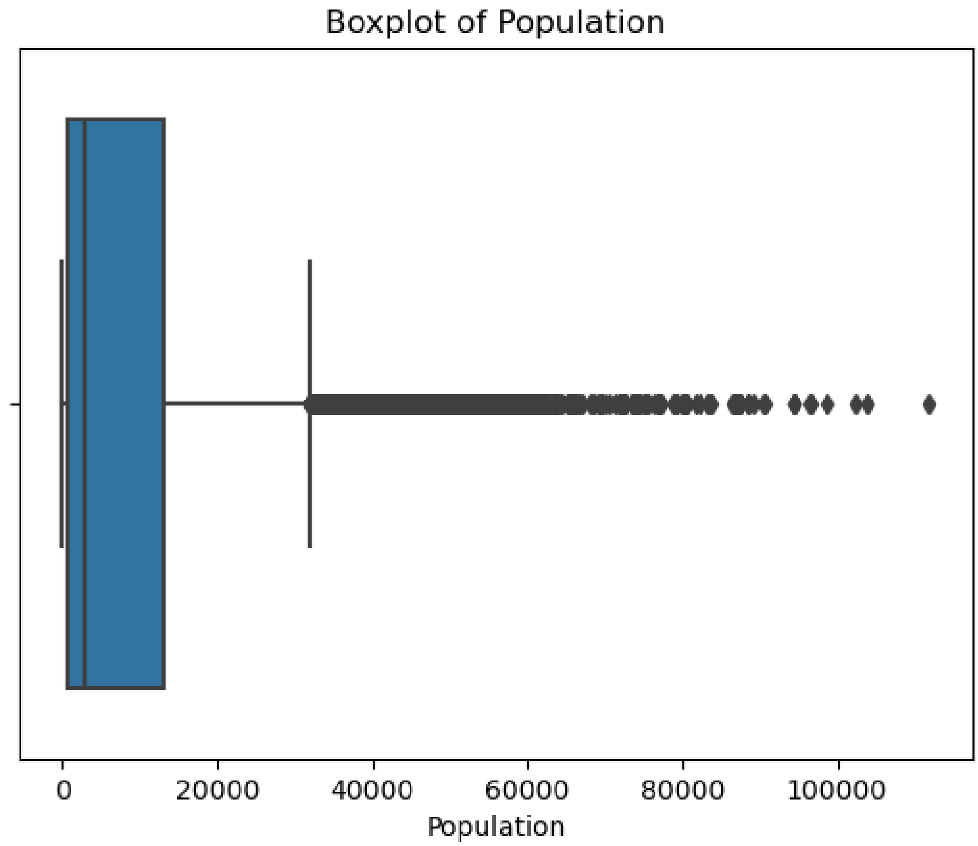
[4]:
```
## C3 Detection of Outliers

numericVars = df[['Lat', 'Lng', 'Population', 'Children', 'Age', 'Income',␣
 ↪'Outage_sec_perweek', 'Email',
          'Contacts', 'Yearly_equip_failure', 'Tenure', 'MonthlyCharge',␣
 ↪'Bandwidth_GB_Year']]

for col in numericVars:
    boxplotOutliers(df, col)
```



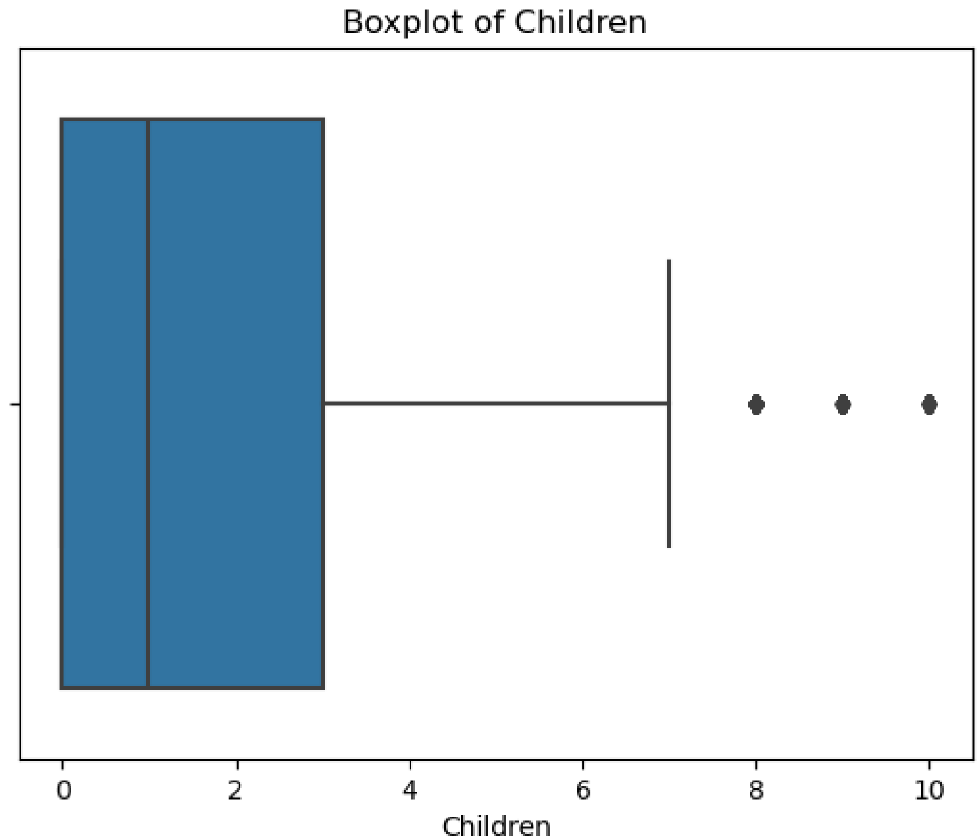
For the `Lat` variable, all observations greater than 52.25 or less than 25.19 are considered outliers.
The count of observations greater than 52.25 is 77.
The count of observations less than 25.19 is 81.



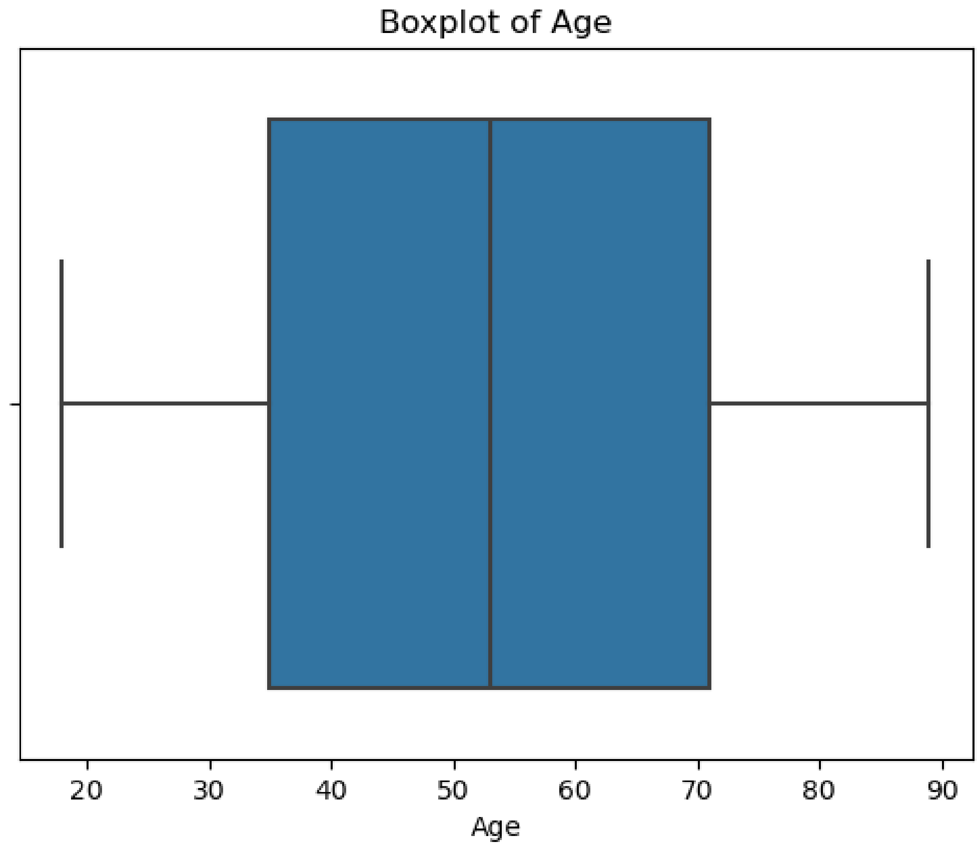
For the `Lng` variable, all observations less than -122.57 are considered outliers.
The count of observations less than -122.57 is 273.



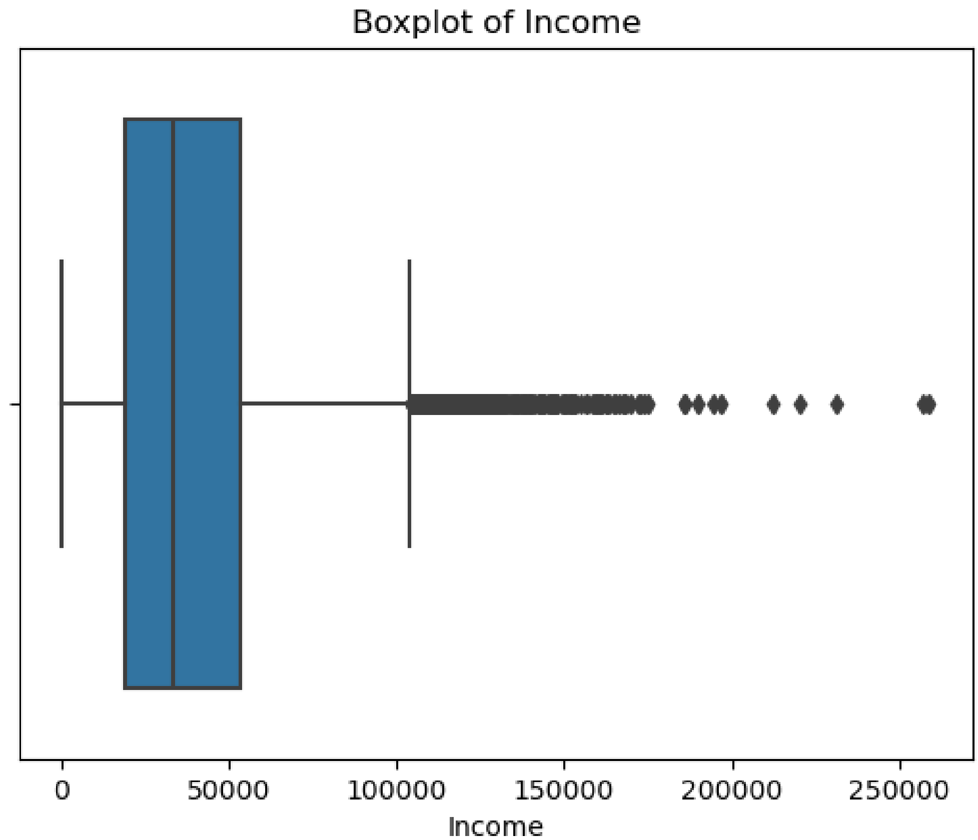
For the `Population` variable, all observations greater than 31813.0 are considered outliers.
The count of observations greater than 31813.0 is 937.



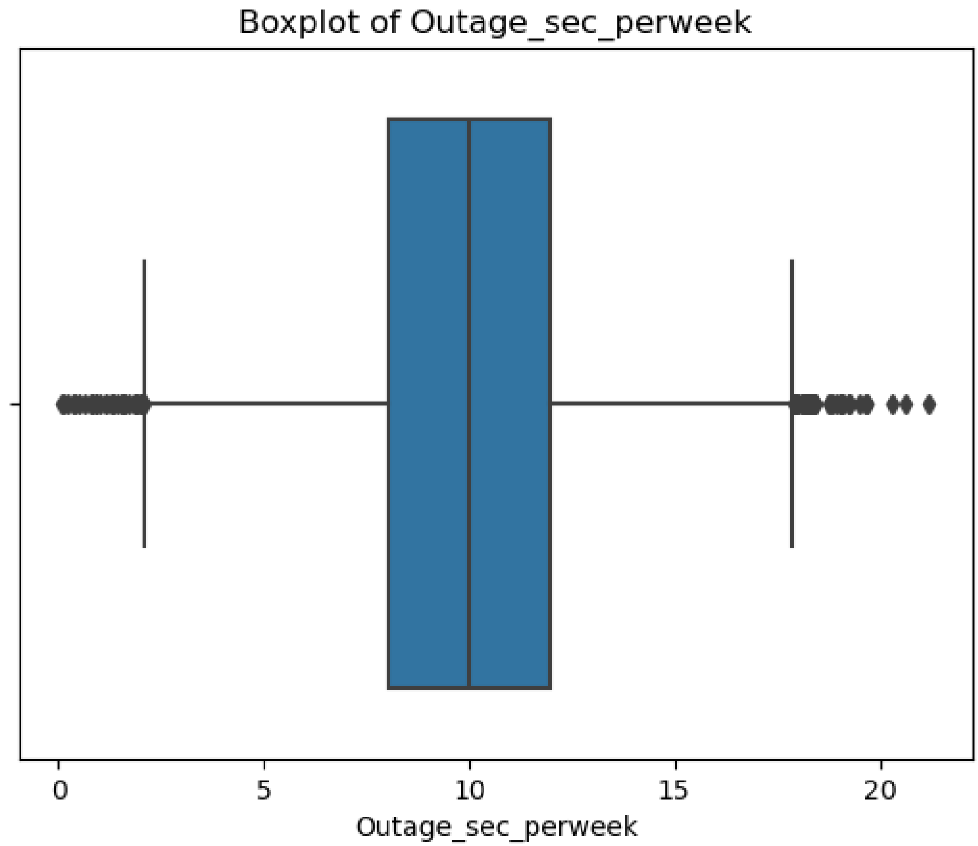

```
For the `Children` variable, all observations greater than 7.5 are considered
outliers.
The count of observations greater than 7.5 is 401.
```



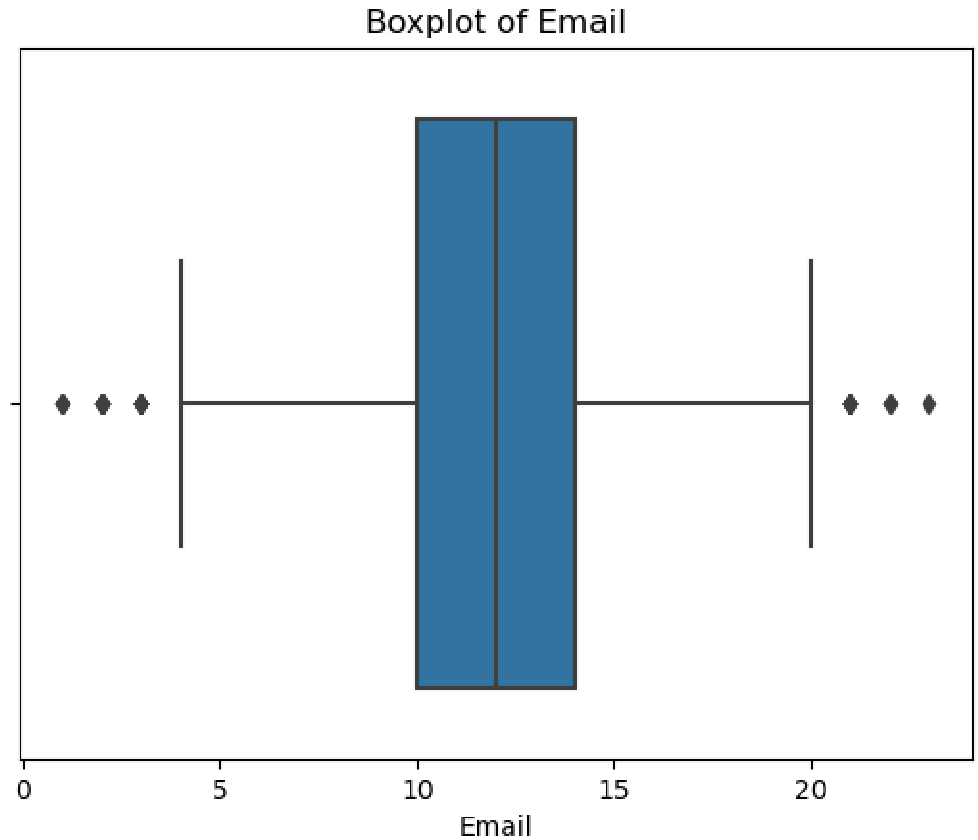
There are no outliers for the `Age` variable.



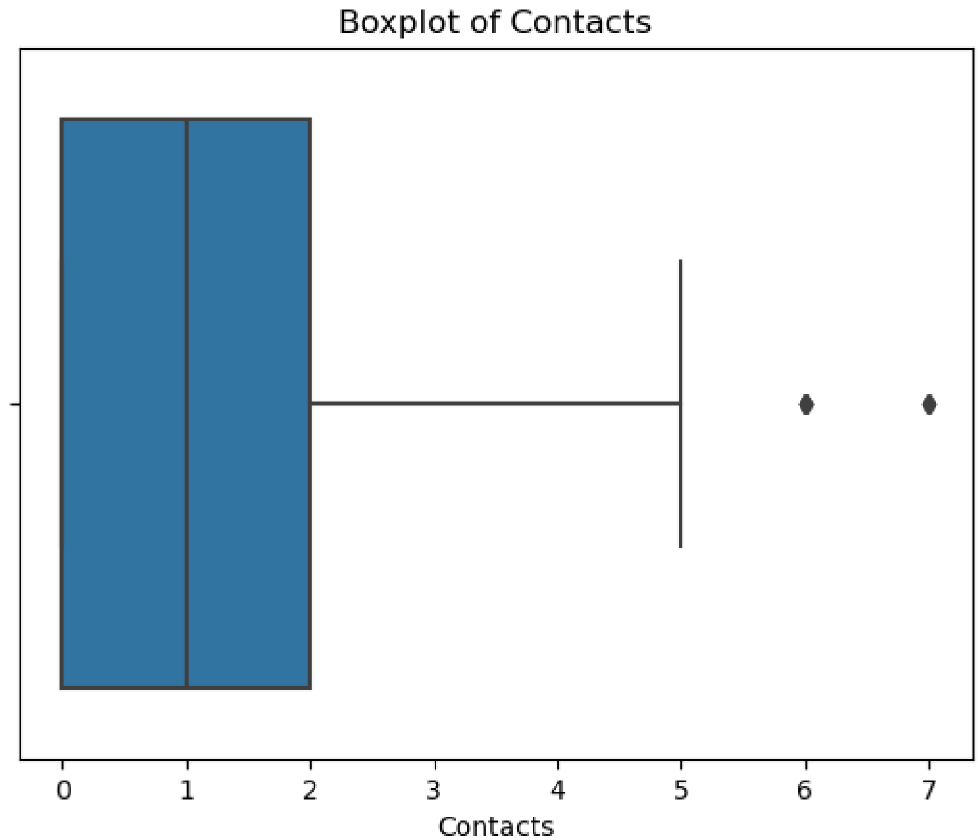

```
For the `Income` variable, all observations greater than 104278.35 are
considered outliers.
The count of observations greater than 104278.35 is 336.
```



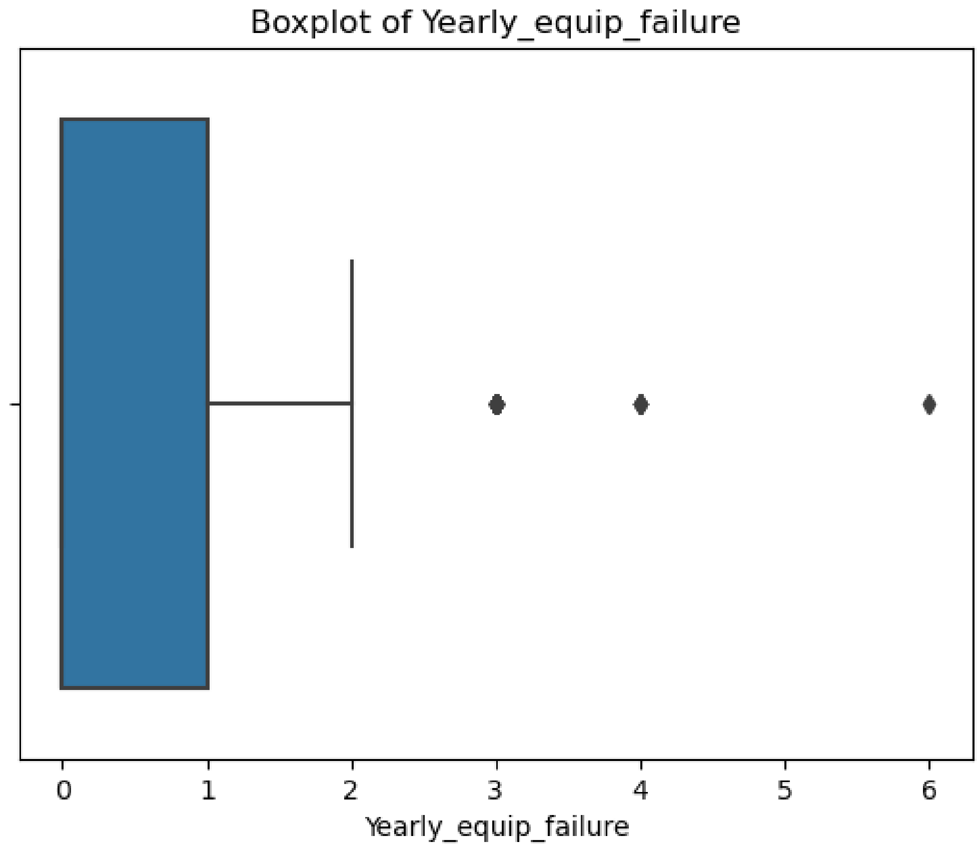
For the `Outage_sec_perweek` variable, all observations greater than 17.9 or less than 2.09 are considered outliers.
The count of observations greater than 17.9 is 43.
The count of observations less than 2.09 is 33.



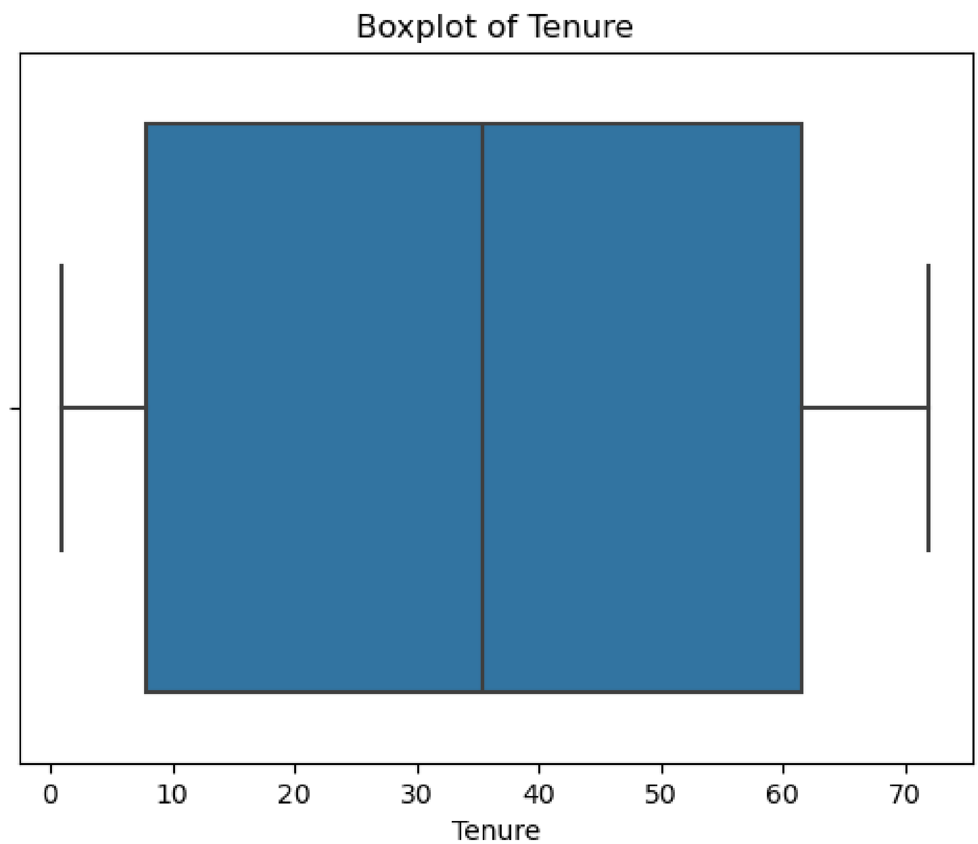

```
For the `Email` variable, all observations greater than 20.0 or less than 4.0
are considered outliers.
The count of observations greater than 20.0 is 15.
The count of observations less than 4.0 is 23.
```



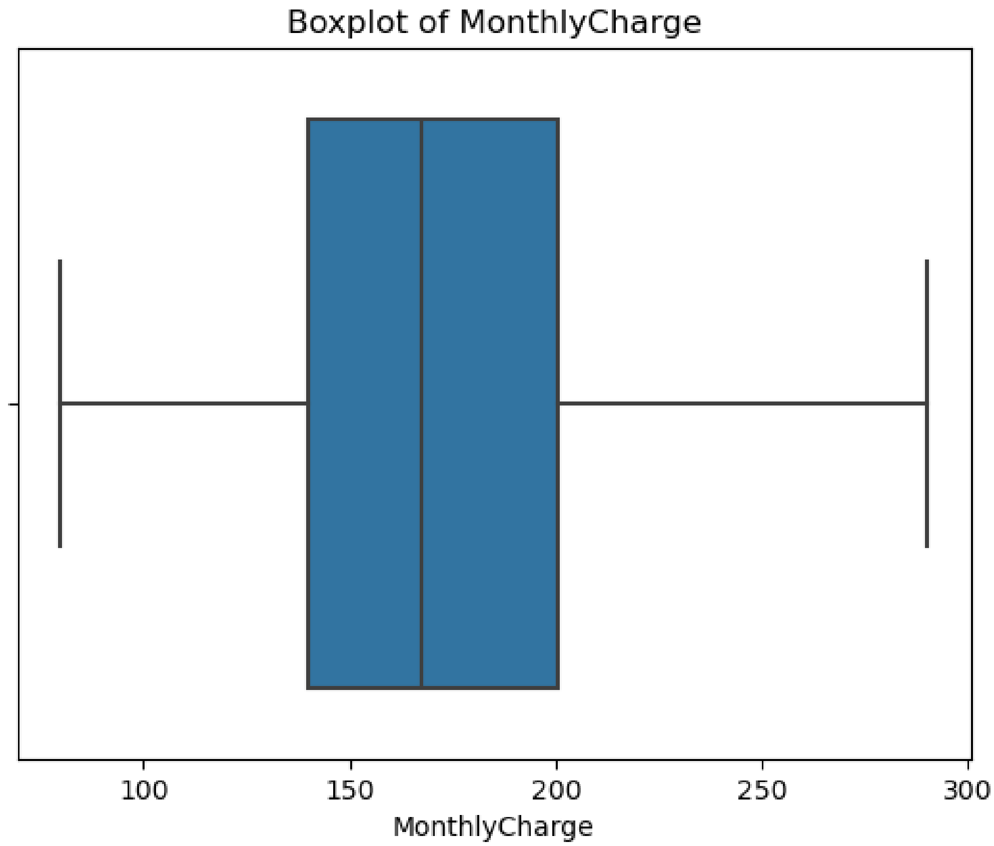

```
For the `Contacts` variable, all observations greater than 5.0 are considered
outliers.
The count of observations greater than 5.0 is 8.
```



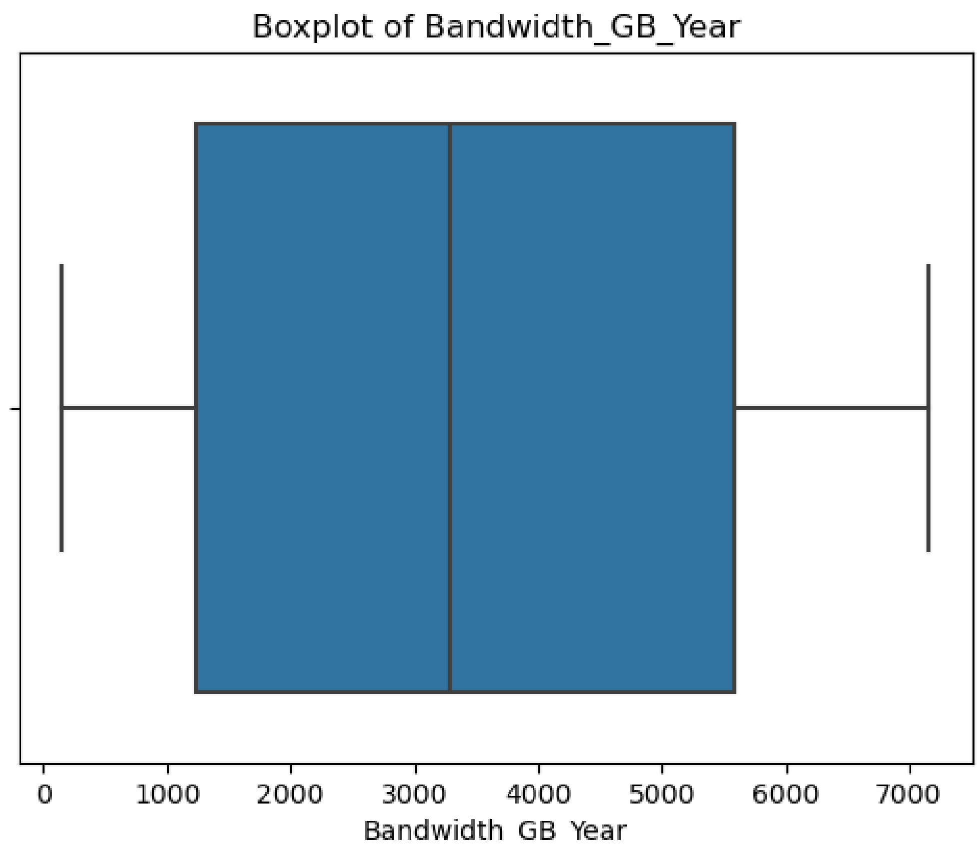

```
For the `Yearly_equip_failure` variable, all observations greater than 2.5 are
considered outliers.
The count of observations greater than 2.5 is 94.
```



There are no outliers for the `Tenure` variable.



There are no outliers for the `MonthlyCharge` variable.



There are no outliers for the `Bandwidth_GB_Year` variable.

[5]:
```
## C3 Treatment of Missing Values

# Since the 'InternetService' variable has 'None' as one of its options,
# it is necessary to impute 'None'

df['InternetService'].fillna('None', inplace=True)

# Verify that 'None' no longer appears as 'Null'
print('Number of `InternetService` nulls:', df['Tenure'].isnull().sum())
```

```
Number of `InternetService` nulls: 0
```

[6]:
```
## C3 Binary Encoding Re-expression of the Thirteen Binary Variables and Type
 ↪Casting

# # It was necessary to keep a copy of the Churn variable as strings for the
 ↪bivariate graphs
# df['ChurnStr'] = df['Churn'].copy()
```

```
# Create a list of the columns that will be encoded
binaryList = ['Churn', 'Techie', 'Port_modem', 'Tablet', 'Phone',
                  'Multiple', 'OnlineSecurity', 'OnlineBackup',␣
 ↪'DeviceProtection',
                  'TechSupport', 'StreamingTV', 'StreamingMovies',␣
 ↪'PaperlessBilling']

binaryDict = {'Yes': 1, 'No': 0}

# Run a loop that replaces all 'Yes' with 1 and 'No' with 0 for each column in␣
 ↪the list above
for col in binaryList:
    df[col] = df[col].replace(binaryDict)

binaryVars = df[['Techie', 'Port_modem', 'Tablet', 'Phone',
                  'Multiple', 'OnlineSecurity', 'OnlineBackup',␣
 ↪'DeviceProtection',
                  'TechSupport', 'StreamingTV', 'StreamingMovies',␣
 ↪'PaperlessBilling']]

# Type Casting Binary Variables
binaryVars = binaryVars.astype('category')

df['Churn'] = df['Churn'].astype('category')
```

[7]:
```
## C3 One-Hot Encoding Re-expression of the Six Categorical Variables

# Data frame of categorical variables
catVars = df[['Area', 'Marital', 'Gender', 'Contract', 'InternetService',␣
 ↪'PaymentMethod']]

# Create additional data frame from variables being re-expressed
oneHotVars = catVars

# Apply one-hot encoding
oneHotVars = pd.get_dummies(oneHotVars, drop_first = False, dtype = int)

# Assign Ordinal Variables
ordinalVars = df[['Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6',␣
 ↪'Item7', 'Item8']]
```

[8]:
```
## C3 Scaling

# Assign independent variable, X, and dependent variable, y
y = df['Churn']
X = pd.concat([numericVars, binaryVars, oneHotVars, ordinalVars], axis = 1)
```

```
# Scaling the Data
X = pd.DataFrame(MinMaxScaler().fit_transform(X), columns = X.columns)
```

[9]:
```
## C3 Feature Selection

# Use SelectKBest to determine which features to include in the reduced model

features = SelectKBest(f_classif, k = 'all')
features.fit_transform(X, y)
pvals = pd.DataFrame({'Feature' : X.columns, 'p_value' : features.pvalues_}).
 ↪sort_values('p_value')
pvals[pvals['p_value'] < 0.05]
```

[9]:

	Feature	p_value
10	Tenure	0.000000e+00
11	MonthlyCharge	0.000000e+00
12	Bandwidth_GB_Year	0.000000e+00
23	StreamingMovies	5.393071e-192
36	Contract_Month-to-month	1.236727e-163
22	StreamingTV	2.414257e-120
38	Contract_Two Year	3.019204e-72
37	Contract_One year	2.359068e-44
17	Multiple	5.642495e-40
39	InternetService_DSL	7.391267e-21
13	Techie	2.408802e-11
40	InternetService_Fiber Optic	4.873098e-09
20	DeviceProtection	1.578944e-08
19	OnlineBackup	4.339213e-07
41	InternetService_None	1.599912e-04
44	PaymentMethod_Electronic Check	2.774461e-03
34	Gender_Male	5.011402e-03
33	Gender_Female	6.887623e-03
16	Phone	8.543973e-03

3.1.4 C4. Copy of the Cleaned Data Set

See code attached, in D209_PA_MendezD_Task1_Attempt2.ipynb.

[10]:
```
## C4 Prepared Data Frame

X_reduced = X[['Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year',␣
 ↪'StreamingMovies', 'Contract_Month-to-month', 'StreamingTV',
               'Contract_Two Year', 'Contract_One year', 'Multiple',␣
 ↪'InternetService_DSL', 'Techie', 'InternetService_Fiber Optic',
               'DeviceProtection', 'OnlineBackup', 'InternetService_None',␣
 ↪'PaymentMethod_Electronic Check',
```

```
              'Gender_Male', 'Gender_Female', 'Phone']]

df_prep = pd.concat([y, X_reduced], axis = 1)

df_prep.to_csv('D209_PA_MendezD_Task1_Attempt2.csv', sep = ',', encoding =␣
 ↪'utf-8', index = False)
```

4 Part IV: Analysis

4.1 D. Perform and Report the Data Analysis

4.1.1 D1. Split Data into Training and Testing Data Sets

[11]:
```
## D1 The following cells include the annotated code used to split the data.
# See code attached, in D209_PA_MendezD_Task1_Attempt1.ipynb

X = X_reduced

# Split the data into four data frames, two training and two testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42, stratify=y)

# Create CSVs from above data frames
X_train.to_csv('D209_PA_MendezD_Task1_X_train.csv', sep = ',', encoding =␣
 ↪'utf-8', index = False)
X_test.to_csv('D209_PA_MendezD_Task1_X_test.csv', sep = ',', encoding =␣
 ↪'utf-8', index = False)
y_train.to_csv('D209_PA_MendezD_Task1_y_train.csv', sep = ',', encoding =␣
 ↪'utf-8', index = False)
y_test.to_csv('D209_PA_MendezD_Task1_y_test.csv', sep = ',', encoding =␣
 ↪'utf-8', index = False)
```

4.1.2 D2. Description of the Analysis Technique

To perform the kNN classification, it was necessary to determine the appropriate value of `k` for these features, which can be done using `GridSearchCV` (Boorman, n.d.). By specifying a range of 1-50 to search over for the `n_neighbors` parameter, we can use `GridSearchCV` to setup a grid search with cross-validation to find the best value for `n_neighbors`. After finding this to be 27, it was possible to proceed with the kNN classification.

kNN classification was performed using `n_neighbors = 27`, and the training data was fit to the model. Then the confusion matrix, training and testing accuracy, and other associated metrics were calculated. The training and testing accuracy were found to be 0.8805 and 0.8755, respectively. Finally, the Receiver Operating Characteristic curve was generated and the Area Under the Curve score for the model was found to be 0.9375.

4.1.3 D3. Code to Perform the Classification Analysis

See code attached, in D209_PA_MendezD_Task1_Attempt2.ipynb.

[12]:
```
## D3 Code adapted from DataCamp material by George Boorman

# Determine best number of neighbors, from k = 1 to k = 50
param_grid = {'n_neighbors' : np.arange(1, 50)}

# Instantiate KNeighborsClassifier object and GridSearchCV object
knn = KNeighborsClassifier()
knn_cv = GridSearchCV(knn, param_grid, cv = 5)

# Fit to training data
knn_cv.fit(X_train, y_train)

# Find best parameter
knn_cv.best_params_
```

[12]:
```
{'n_neighbors': 27}
```

[13]:
```
# D3 Find best score
knn_cv.best_score_
```

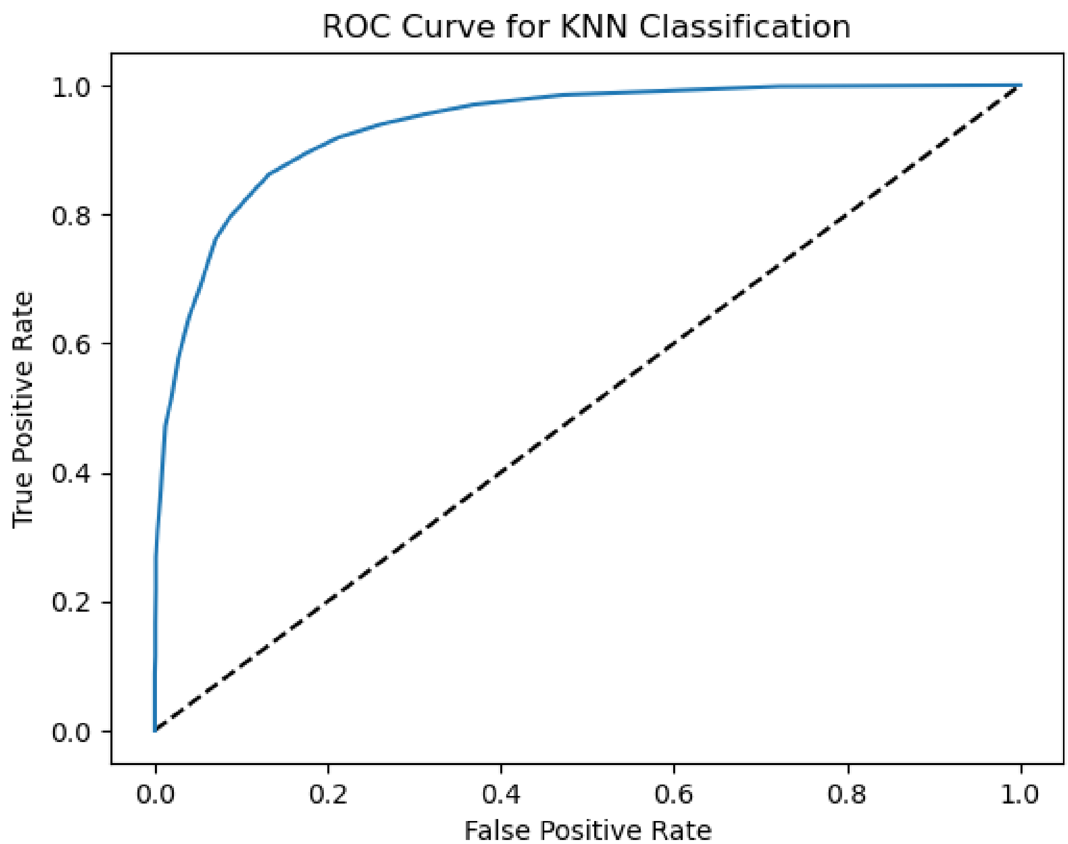
[13]:
```
0.866625
```

[14]:
```
## D3 Code adapted from DataCamp material by George Boorman

# Perform KNN using the value of k = 27 as found above
knn = KNeighborsClassifier(n_neighbors = 27)

# Fit to the training data
knn.fit(X_train, y_train)

# Generate y_pred for Confusion Matrix
y_pred = knn.predict(X_test)
conf = confusion_matrix(y_test, y_pred)

# Training and Testing Accuracy Scores
train_acc = knn.score(X_train, y_train)
test_acc = knn.score(X_test, y_test)

print('The confusion matrix is: ')
print(conf)
print(f'The training accuracy of this KNN classification is {train_acc}.')
print(f'The testing accuracy of this KNN classification is {test_acc}.')
print(classification_report(y_test, y_pred))
```

```
The confusion matrix is:
[[1412   58]
 [ 191  339]]
The training accuracy of this KNN classification is 0.8805.
The testing accuracy of this KNN classification is 0.8755.
              precision    recall  f1-score   support

           0       0.88      0.96      0.92      1470
           1       0.85      0.64      0.73       530

    accuracy                           0.88      2000
   macro avg       0.87      0.80      0.83      2000
weighted avg       0.87      0.88      0.87      2000
```

[15]:

```
## D3 ROC Curve and AUC. Code adapted from DataCamp material by George Boorman

y_pred_prob = knn.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for KNN Classification')
plt.show()

print(f'The Area Under the Curve (AUC) is: {roc_auc_score(y_test,␣
 ↪y_pred_prob)}\n')
```




```
The Area Under the Curve (AUC) is: 0.9375067385444744
```

5 Part V: Data Summary and Implications

5.1 E. Summarize your data analysis by doing the following:

5.1.1 E1. Accuracy and AUC Scores

As shown above, the training accuracy and testing accuracy scores were 0.8805 and 0.8755, respectively. The training accuracy score shows that the model accurately predicted 88.05% of the data on which it was trained, that is, the model can accurately classify about 88% of the training data. The testing score shows that the model can accurately predict 87.55% of new data, which indicates that the model has adequately captured the underlying patterns of the data (Accuracy vs. precision vs. recall in machine learning). With an AUC of 0.9375, the model's predictive ability is significantly better than random guessing, which corresponds to AUC = 0.5 (How to explain the ROC curve and ROC AUC score?).

5.1.2 E2. Results and Implications of the Classification Analysis

Since the model was able to accurately predict 87.55% of new data, the model may have the potential to predict whether a customer will churn. Furthermore, the AUC score produced was in the reasonable range, which reaffirms the model's predictive ability. However, the model may not be suited for deployment by the stakeholders, as it could be improved upon.

5.1.3 E3. One Limitation of the Data Analysis

A possible limitation of the analysis was the choice to retain outliers. It is possible that the model may have been more accurate if the outliers were treated with imputation of statistical measures.

5.1.4 E4. Recommended Course of Action

Since many of the features deemed to be influential to churn are services provided by the stakeholders, my recommended course of action would be to use this information to develop programs that will retain customers. For example, since features such as `MonthlyCharge`, `StreamingTV`, and `Multiple` were found to influence churn, programs like incentives or discounts could be offered to encourage customer retention.

6 Part VI: Demonstration

6.1 F. Panopto Video

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=3890cbfa-e4f9-4a29-a416-b1d10161fbdb

6.2 G. Acknowledgement of Web Sources

Boorman, G. (n.d.). *Supervised Learning with scikit-learn.* DataCamp. https://app.datacamp.com/learn/courses/supervised-learning-with-scikit-learn

scikit-learn . (2019). sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.22.1 documentation. Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

6.3 H. Acknowledgement of Sources

Accuracy vs. precision vs. recall in machine learning: what's the difference? (n.d.). EvidentlyAI. https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall

How to explain the ROC curve and ROC AUC score? (n.d.). EvidentlyAI. https://www.evidentlyai.com/classification-metrics/explain-roc-curve

Elleh, F. D209 Task 1: Expectations and Data Preprocessing - Python. https://westerngovernorsuniversity.sharepoint.com/:p:/r/sites/DataScienceTeam/Shared%20Documents/Graduat