D209 Task 1: Classification Analysis

Part I: Research Question

A1. Research Question

Which customer variables are at the highest risk of churn using the k-nearest neighbor (KNN) method?

A2. Data Analysis Goal

The goal of this data analysis is to develop a machine learning model using the k-nearest neighbor (KNN) method to gain more insight to determine which variables from the churn dataset are at the highest risk of customer churn.

Part II: Method Justification

B1. Classification Method Explanation

The k-nearest neighbor method stores the similarity distances of earlier data points to classify the new ones by determining the *k*, the number of closest neighbors used to classify the new data points to predict the label of any data point.

Therefore, the expected outcome is to classify the test data points according to their closest neighbors (Kumar, 2020).

B2. Method Assumption Summary

An assumption of k-NN classification method is if a data point is in close proximity to another group, it is similar to those data points (Harrison, 2019).

B3. Packages & Libraries Used

I will be using Python to perform this data analysis because of the consistent syntax that makes it easy to learn and follow along, the flexibility to create and learn new things, and all of the libraries and packages that it has to offer. For example, I will be using the following libraries and packages for my analysis (R or Python 2023):

- pandas- to load datasets
- NumPy- to work with arrays
- Sci-kit Learn- for machine learning and to transform our data
- Matplotlib- for basic plotting generally consisting of bars, lines, pies, scatter plots, and graphs
- Seaborn- for a variety of visualization patterns

Part III: Data Preparation

C1. Data Preprocessing Goal

A data preprocessing goal for the KNN model is to remove any outliers, impute any missing data, and re-express the binary categorical variables (yes/no) as dummy variables where 1 represents yes and 0 represents no.

C2. Dataset Variables

Variable Name	Data Class	Data Type
Children	Quantitative	Numeric
Income	Quantitative	Numeric
Churn	Qualitative	Categorical
Outage_sec_perweek	Quantitative	Numeric
Email	Quantitative	Numeric
Contacts	Quantitative	Numeric
Yearly_equip_failure	Quantitative	Numeric
Techie	Qualitative	Categorical
Contract	Qualitative	Categorical
Port_modem	Qualitative	Categorical
Tablet	Qualitative	Categorical
InternetService	Qualitative	Categorical
Phone	Qualitative	Categorical
Multiple	Qualitative	Categorical
OnlineSecurity	Qualitative	Categorical
OnlineBackup	Qualitative	Categorical
DeviceProtection	Qualitative	Categorical
TechSupport	Qualitative	Categorical
StreamingTV	Qualitative	Categorical
StreamingMovies	Qualitative	Categorical
Tenure	Quantitative	Numeric
MonthlyCharge	Quantitative	Numeric
Bandwidth_GB_Year	Quantitative	Numeric
Item1	Quantitative	Numeric
Item2	Quantitative	Numeric
Item3	Quantitative	Numeric
Item4	Quantitative	Numeric
Item5	Quantitative	Numeric
Item6	Quantitative	Numeric
Item7	Quantitative	Numeric
Item8	Quantitative	Numeric

C3. Analysis Steps

1. Import any necessary libraries and packages.
2. Load dataset into pandas data frame using read_csv command. The data frame is named "df".
3. Explore the dataset in order to determine how to evaluate the input data by using the head() command to print the first 5 rows of the dataset.
4. Rename the survey columns to describe the variables better.
5. Print column names to check corrections made.
6. Calculate the total null values and total duplicate values in the dataset. If there are not any, the values will be shown as 0.
7. Drop columns that are unnecessary for the analysis.
8. Use the head() command to look at what data is left.
9. Find the summary statistics for the variables used in the analysis.
10. Create the univariate visualizations for all the predicting variables and the target variable.
11. Drop either Tenure or Bandwidth_GB_Year because of multicollinearity from a previous analysis.
12. Reformat the columns to have 3 decimal places.
13. Create dummy variables where "Yes" is represented by 1 & "No" is represented by 0.
14. Drop the original categorical columns so only dummy categorical columns are left.
15. Check the new data frame to make sure all the data transferred correctly.
16. Extract the cleaned dataset.

C4. Cleaned Dataset

The cleaned dataset will be attached to submission as "knn_clean.csv".

Part IV: Analysis

D1. Splitting the Data

The csv files of the split data into training and testing sets will be attached to the submission as "X_train.csv", "X_test.csv", "y_train.csv", and "y_test.csv".

```
# Split the data into X & y
y = df.DummyChurn
X = df[['Tenure', 'DummyStreamingMovies', 'DummyStreamingTV', 'DummyContract',
       'DummyMultiple', 'DummyTechie', 'DummyInternetService',
       'DummyDeviceProtection', 'DummyOnlineBackup', 'DummyPhone']]
```

```
# Create the training & test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=1)
```

```
# Save the training and testing sets as csv files
pd.DataFrame(X_train).to_csv('X_train.csv')
pd.DataFrame(X_test).to_csv('X_test.csv')
pd.DataFrame(y_train).to_csv('y_train.csv')
pd.DataFrame(y_test).to_csv('y_test.csv')
```

D2. Output & Intermediate Calculations

The first analysis technique I used to appropriately analyze the data was by using SelectKBest from scikit-learn to select the features with a significant p-value and build an initial model.

```
# Create a new dataframe with the selected features
X_skbest = X[features_to_keep]

# Calculate the VIF to check for multicollinearity
vif = pd.DataFrame()
vif["Feature"] = X_skbest.columns
vif["VIF"] = [variance_inflation_factor(X_skbest.values, i) for i in range(X_skbest.shape[1])]

# Print the VIFs
print(vif)
```

```
                 Feature        VIF
0          MonthlyCharge  59.123125
1                 Tenure   2.646306
2    DummyStreamingMovies   5.290966
3        DummyStreamingTV   4.325467
4           DummyContract   1.319610
5           DummyMultiple   3.228777
6             DummyTechie   1.201250
7    DummyInternetService   2.600339
8   DummyDeviceProtection   2.073906
9       DummyOnlineBackup   2.544588
10             DummyPhone   9.090603
```

```
# Run the model after the removal of "MonthlyCharge" since it had a high VIF (greater than 10)
X_skbest = X[features_to_keep].drop(columns="MonthlyCharge")
vif = pd.DataFrame()
vif["Feature"] = X_skbest.columns
vif["VIF"] = [variance_inflation_factor(X_skbest.values, i) for i in range(X_skbest.shape[1])]

print(vif)
```

```
               Feature      VIF
0               Tenure  2.456814
1   DummyStreamingMovies  1.855791
2       DummyStreamingTV  1.857582
3          DummyContract  1.302950
4          DummyMultiple  1.772117
5            DummyTechie  1.189112
6   DummyInternetService  1.709921
7  DummyDeviceProtection  1.694815
8      DummyOnlineBackup  1.743520
9              DummyPhone  4.883828
```

```
X_skbest.columns
```

```
Index(['Tenure', 'DummyStreamingMovies', 'DummyStreamingTV', 'DummyContract',
       'DummyMultiple', 'DummyTechie', 'DummyInternetService',
       'DummyDeviceProtection', 'DummyOnlineBackup', 'DummyPhone'],
      dtype='object')
```

The next step is to calculate the variance inflation factors of the selected features to check for multicollinearity. If any of the VIFs are greater than 10, it will be removed from the analysis. In this case, MonthlyCharge had a much greater VIF, so it was removed. The model was run again to make sure the VIFs were less than 10.

```
# Set the predictor variables & target variable
y = churn_df["DummyChurn"]
X = churn_df.drop(columns=["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
print("Selected Features:")
print(features_to_keep)
```

```
Selected Features:
7             MonthlyCharge
6                    Tenure
28     DummyStreamingMovies
27         DummyStreamingTV
17            DummyContract
22            DummyMultiple
16              DummyTechie
20     DummyInternetService
25    DummyDeviceProtection
24        DummyOnlineBackup
21               DummyPhone
Name: Feature, dtype: object
```

After the features to keep were determined, the next step is to determine the best k-value for KNN classification. To do so, I chose the k range of 1 to 30 and used the GridSearchCV() method to define the parameter range. This method is referred to as hyperparameter tuning where a model is built for every possible combination of the provided hyperparameters and the model with the best results is the one chosen (Jordan, 2018).

```
# Determine the best k value
knn = KNeighborsClassifier()
k_range = list(range (1,30))
param_grid = dict(n_neighbors=k_range)

#Define the parameter range
knn_cv = GridSearchCV(knn,param_grid,cv=10,scoring='accuracy',return_train_score=False,verbose=1)

#Fit the model for grid search
knn_cv.fit(X_train, y_train)

print('The best k value is: ')
print(knn_cv.best_params_)

# Calculate the accuracy
print('The accuracy is: ')
print(knn_cv.best_score_)
```

```
Fitting 10 folds for each of 29 candidates, totalling 290 fits
The best k value is:
{'n_neighbors': 23}
The accuracy is:
0.854875
```

Then, the next step of the intermediate calculations is to fit the KNN model using the grid search results from above where the best k value is 23.

```
# Fit the KNN model using grid search result of k = 23
knn = KNeighborsClassifier(n_neighbors=23)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)
y_pred_prob = knn.predict_proba(X_test)[:,1]
```

After the KNN model was fit, the classification report of the model is generated as well as a confusion matrix to summarize the performance of the classification model that has been created. As shown below, the confusion shows out of 2000 observations, 1683 were classified correctly while 317 were classified incorrectly. Hence why the accuracy of the model is 84.15%.

```
# Print the classification report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.92      0.89      1442
           1       0.76      0.64      0.69       558

    accuracy                           0.84      2000
   macro avg       0.81      0.78      0.79      2000
weighted avg       0.84      0.84      0.84      2000
```

```
# Print the confusion matrix of the model
y_pred = knn.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

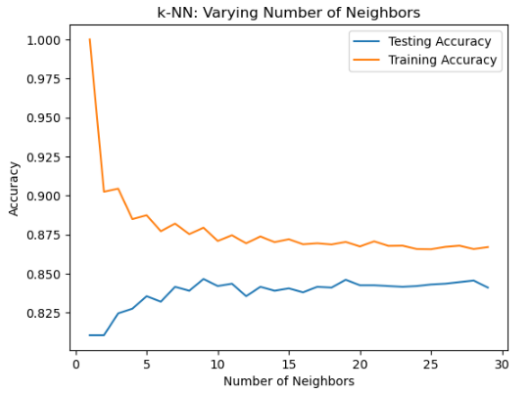
```
[[1327  115]
 [ 202  356]]
```

```
# Print the accuracy scores
print('The accuracy of the model is: ', knn.score(X_test, y_test))
print('The accuracy of the training model is: ', knn.score(X_train, y_train))
```

```
The accuracy of the model is:  0.8415
The accuracy of the training model is:  0.867875
```

The next step is to generate a model complexity curve to visualize the values of k and how they affect the model's training and testing data.

```
# Model complexity curve
neighbors = np. arange(1, 30)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

# Loop over different values of k
for i, k in enumerate (neighbors):

    #Setup a k-NN Classifier with k neighbors: knn
    knn = KNeighborsClassifier(n_neighbors=k)

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

    #Compute accuracy on the training set
    train_accuracy[i] = knn.score(X_train, y_train)

    #Compute accuracy on the testing set
    test_accuracy[i] = knn.score(X_test, y_test)


#Generate plot
plt.title ('k-NN: Varying Number of Neighbors')
plt.plot(neighbors, test_accuracy, label = 'Testing Accuracy')
plt.plot (neighbors, train_accuracy, label = 'Training Accuracy')
plt.legend()
plt.xlabel('Number of Neighbors')
plt.ylabel('Accuracy')
plt.show()
```

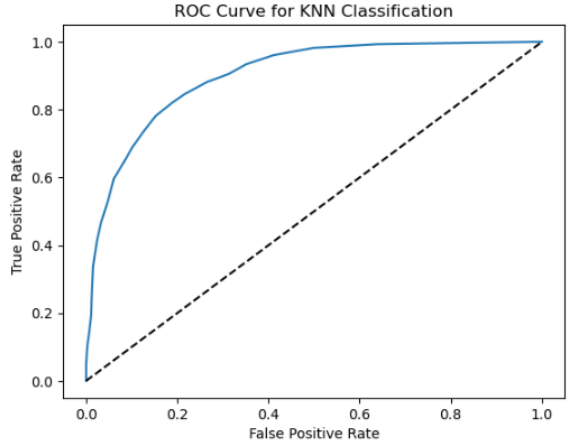


After calculating all of the above data, the last step is to compute the area under the Receiver Operating Characteristic (ROC) Curve and generate the plot for the ROC Curve. As shown below, the AUC-ROC score is 0.9. This score is considered to be very good because a score of 1.0 is the ideal score (Brownlee, 2018).

```
# Compute the area under the Receiver Operating Characteristic Curve
auc_roc = roc_auc_score(y_test, y_pred_prob)
print('AUC-ROC score: ', auc_roc)
```

```
AUC-ROC score:  0.90030709538226
```

```
# Generate plot for ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
plt.plot(fpr, tpr)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for KNN Classification')
plt.show()
```



D3. Code Execution

All of the code from part D2 can be found in provided file "D209 Task 1.ipynb".

Part V: Data Summary & Implications

E1. Accuracy & AUC

The accuracy of the model is 0.8415 which is 84.15%. Classification accuracy represents the ratio of the number of correct predictions out of all the predictions, meaning that the model created predicted the test data 84.15% correctly.

The area under the curve (AUC) represents the probability that the model will be able to distinguish between positive and negative classes if given a random positive or negative example (*Classification: Roc and AUC*). The perfect model would have an AUC of 1.0 which would mean there is a probability of 100% that the model would have predicted everything perfectly. An AUC of 0.5 is pretty much random. Hence, having an AUC of 0.90030709538226 is an indicator that the model provides a great prediction of classes.

E2. Results & Implications

I used the k-nearest neighbor algorithm to solve classification problems. The features used in the model were 'Tenure', 'DummyStreamingMovies', 'DummyStreamingTV', 'DummyContract', 'DummyMultiple', 'DummyTechie', 'DummyInternetService', 'DummyDeviceProtection', 'DummyOnlineBackup', and 'DummyPhone'. I used the SelectKBest approach to choose which features are most important to use in the model. After that, I found the variance inflation factors of the selected features to check for multicollinearity. If the VIF was greater than 10, it was removed. After that, I created the classification report to provide the precision, recall, F-score, support, and accuracy of the model. The precision for predicting both the negative (0) and positive (1) class was fairly high with the precision of predicting negative class being 0.87 and the positive being 0.76. The recall is the ability of a classifier to identify the true positives in a class. The recall for class 0 was 0.92 and class 1 was 0.64. Therefore, class 0 was nearly perfect, and class 1 was quite a bit lower, but still greater than 0.5. If it were 0.5 or lower, this would mean there were many false negatives, resulting in an imbalanced class (Kohli, 2019). Overall, it is implied that all of the values in the classification report show that this model is a good model which is further proven by the accuracy of 0.84. However, it is difficult to evaluate the accuracy of the predicted output versus the calculated output in terms of the initial research question.

E3. Limitation

One limitation of the analysis would be what I mentioned above about how it is difficult to evaluate the accuracy of the predicted output versus the calculated output in terms of the initial research question. The reason I say this is because although all of the values are saying this model is accurate with the selected features, I would initially hypothesize that certain features would be included in the analysis pertaining to what customer variables are at the

highest risk of churn. This leads me to believe that perhaps the algorithm was not executed properly which could have led to inaccurate results.

E4. Course of Action

A course of action based on the analysis results would be to have the stakeholders look into customers that do not have extra add-ons like streaming movies, TV, multiple services, internet service, device protection, online backup, phone services, etc. It seems as if the customers who are not taking full advantage of all of the add-on options that are offered by the telecommunications company are at the highest risk of churn. As a result, reaching out to these types of customers to offer them more add-ons may not only benefit the company by bringing in more revenue but also retain these customers by making them feel as if they are getting more value for the money they are spending.

Part VI: Demonstration

F. Panopto Video

G. Third-Party Code Sources

N/A

H. Sources

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

Google. (n.d.). Classification: Roc and AUC. Google. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc#:~:text=hypothetical%20perfect%20model.-,Area%20under%20the%20curve%20(AUC),positive%20higher%20than%20the%20negative.

Harrison, O. (2019, July 14). Machine learning basics with the K-nearest neighbors' algorithm. Medium. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

Jordan, J. (2018, December 5). Hyperparameter tuning for machine learning models. https://www.jeremyjordan.me/hyperparameter-tuning/

Kohli, S. (2019, November 18). Understanding a classification report for your machine learning model. Medium. https://medium.com/@kohlishivam5522/understanding-a-classification-report-for-your-machine-learning-model-88815e2ce397

Kumar, A. (2020, May 27). KNN algorithm: What? When? Why? How? Medium. https://towardsdatascience.com/knn-algorithm-what-when-why-how-41405c16c36f

I. Professional Communication

Demonstrate professional communication in the content and presentation of your submission.