Briana Churchill

Student ID: 011009463

Dr. Eric Straw

October 2, 2023

NVM2 TASK 1: CLASSIFICATION ANALYSIS

Part I: Research Question

A1. Question and Method Used to Analyze

Each individual customer has needs of their own, whether it is multiple phone lines and DSL internet, or a single line with no internet, etcetera. Therefore, it would be of little importance for this analysis to compare potential add-on services or products. Instead, it's better to take note that each customer is valued by the business regardless of their telecommunication needs and so maintaining the consumer as a customer is of utmost importance. As such, my question for this analysis is: Could the k-nearest neighbor method be used to predict which customers may be at risk for churn in an effort to maximize customer retention rates?

A2. Goal of the Data Analysis

In my most recent analysis in D208, I used a logistic regression model to assess the question: "What factors are primarily contributed to customer churn?" Unfortunately, the analysis rendered ineffective. However, the data cleaning and transformation steps used in that analysis are a great resource to start with the new research question and KNN model. Because the previous analysis would not be beneficial for the telecommunications business, the goal for this analysis is to determine if a different method could render more useful information to answer the question at hand. Finding information that could help reduce the risk of churn would be greatly beneficial to the company.

Part II: Method Justification

B1. Chosen Classification Method and Expected Outcomes

Although 10,000 data points is enough for the Naïve Bayes classification, it is recommended for larger datasets and issues I had with previous analyses relating to the limited amount of data, I thought k-nearest neighbor would be more appropriate to answer the question. The k-nearest neighbors model will seek out data points that could have a relationship or be grouped together ("neighbors") and could be utilized to make predictions. To develop the k-NN model, the data will be split into testing and training data sets. The training dataset prepares the model and allows the model to become familiar with any patterns, and to determine the best "k"

value or number of neighbors. The model will then make predictions using the test data given the best "k" value. These predictions will be analyzed to determine how well the model performs and if it could be used to implement changes within business practices.

B2. Summarize one assumption of the chosen classification method

To make predictions of churn rates based on potential factors, it would be extremely important to ensure that those potential factors are reliable. One assumption of the k-nearest neighbors method is that "similar things exist in proximity to each other." (Straw, 2023) As such, if any of the features tested in the method exist in proximity to the target (churn), then there is predictive power present, and the model could be used in a beneficial manner to the business.

B3. Packages or libraries used in Python

The packages and libraries used and their importance to the analysis is described below:

- Numpy
  - Transforms the data through mathematical equations.
- Pandas
  - Provides a logical structure, otherwise known as the data frame
  - Used to create dummy variables
  - Also used to save the data to CSV
- Sklearn
  - Used to normalize the data to better understand the data in the regression models (Preprocessing & Standard scaler)
  - Splits the data into train and test sets (train test split)
  - Facilitates KNN model (KNeighborsClassifier)
  - Implement confusion matrix, accuracy score & F1 score
  - Determine the area under the curve score (ROC [curve] & AUC score)
- Matplotlib_pyplot
  - Used to plot the ROC curve

Part III: Data Preparation

C1. One data preprocessing goal

With knowledge previously learned in D208, one data preparation goal for this analysis is to scale and standardize the data through preprocessing and standard scaler. Given the various differences in values within columns used, standardizing the data allows it to be read and understood more easily.

C2. Initial variables and their classification

Categorical	Continuous/Numeric
Area (Rural, Suburban, Urban)	Zip code
Marital	Children
Gender	Age
Techie	Income
Contract	Tenure
Portable Modem	Bandwidth Usage
Internet Service	Monthly Charge
Tablet	Service Outages

C3. Explain the steps used to prepare the data and identify the code segment for *each* step

After cleaning the data using scripts used in D207 and D208, I altered the data quite a bit further to prepare it for the analysis. Starting with code previously used to convert column data types, map values, create dummy variables, and lastly to drop unnecessary variables.

Initially the "Area" column was changed to string and front filled with zeros to correct the zip codes that are < 5 characters. Then updated to be data type category as can be seen below.

```
df['Zip'] = df['Zip'].astype("str").str.zfill(5)
df["Area"] = df["Area"].astype("category")
```

The columns with categorical variables were all converted to data type category. The variables and code used are:

- Marital
  - df["Marital"] = df["Marital"].astype("category") # Convert Gender column to category from object
- Gender
  - df["Gender"] = df["Gender"].astype("category")
- Internet (previously InternetService)
  - df["Internet"] = df["Internet"].astype("category")
- Contract
  - df["Contract"] = df["Contract"].astype("category")
- Techie
  - df["Techie"] = df["Techie"].astype("category")
- Modem (previously Port_modem)
  - df["Modem"] = df["Modem"].astype("category")
- Tablet
  - df["Tablet"] = df["Tablet"].astype("category")

- Streaming TV
    - df["Streaming_TV"] = df["Streaming_TV"].astype("category")
- Streaming Movies
    - df["Streaming_Movies"] = df["Streaming_Movies"].astype("category")
- Payment Method
    - df["Payment_Method"] = df["Payment_Method"].astype("category")
- Paperless Billing
    - df["Paperless_Billing"] = df["Paperless_Billing"].astype("category")
- Churn
    - df["Churn"] = df["Churn"].astype("category")

After the categorical variables were taken care of, all the numeric variable columns were converted to data type int.

- Income
    - df["Income"] = df["Income"].astype(int)
- Children
    - df["Children"] = df["Children"].astype(int)
- Age
    - df["Age"] = df["Age"].astype(int)
- Tenure
    - df["Tenure"] = df["Tenure"].astype(int)
- Bandwidth Usage (previously Bandwidth_GB_Year)
    - df["Bandwidth_Usage"] = df["Bandwidth_Usage"].astype(int)
- Equipment failures (previously known as yearly_equip_failure)
- df["Equipment_Failures"] = df["Equipment_Failures"].astype(int)
- Outages (previously known as Outage_sec_perweek)
- df["Outages"] = df["Outages"].astype(int)
- Monthly Charge
    - df["Monthly_Charge"] = df["Monthly_Charge"].astype(int)

Now that all data types have been converted, categorical variables with yes/no data points were mapped to 1 or 0. The applicable variables are Churn, Techie, Tablet and Modem. The scripts used to implement this step is as follows:

```
Cd_map = {'Yes': 1, 'No': 0};
convert = ["Churn" , "Techie" , "Tablet" , "Modem"] df[convert] = df[convert].replace(Cd_map)
```

For the remaining categorical variables, I implemented one-hot encoding to create dummy variables. The affected columns are Area, Gender, Marital, Contract and Internet. The scripts used to implement this:

```
dummy_columns = ["Area" , "Gender" , "Marital", "Contract", "Internet" ]
df = pd.get_dummies(data=df, columns=dummy_columns, drop_first=False)
```

Lastly, before creating the KNN model, all unnecessary columns needed to be dropped. There were quite a few and can be seen in the code used here:

```
Cdrop = ["Customer_id", "Interaction", "UID", "City", "County", "Lat", "Lng",
    "Payment_Method" , "TimeZone", "Email", "Contacts", "Phone", "Paperless_Billing",
    "State", "Population", "Job" , "Phone" , "Multiple" , "Online_Security" , "Online_Backup"
    ,"Device_Protection" , "Tech_Support" , "Streaming_TV" , "Streaming_Movies" ,"Item1" ,
    "Item2" , "Item3" , "Item4" , "Item5" ,"Item6" , "Item7" , "Item8"]
df.drop(columns=Cdrop, axis=1, inplace=True)
```

C4. Copy of the cleaned data set

The cleaned data set CSV file associated with this analysis is attached to the submission of this assessment.

Part IV: Analysis

D1. Split the data into training and test data sets and provide the files

The split data sets and a CSV file for each associated with this analysis are attached to the submission of this assessment.

D2. Describe the analysis technique used

With a mixture of scripts and instruction provided by Datacamp, StackOverflow and Simplilearn which are cited below, I attempted the k-nearest neighbors model. After trial and error, I used snippets of code and information from all sources to come to the best scripts for the data I was using. After splitting the data as previously discussed, I standardized the data using StandardScaler. With varying k values, I created an initial KNN classifier. Using cross validation with instruction from Datacamp, I found the best number of neighbors for my data, which was 1. Using this information, I trained my final model and made predictions. Afterwards I retrieved the accuracy score, f1 score, and confusion matrix. Lastly, I computed the ROC/AUC score and plotted the ROC curve. These scripts and their outputs can be seen in the screengrabs below:

```
# Begin anaylsis by separating the X variables (features) from the Y variable (target)
Xdf = df.drop(["Churn"], axis=1)
Ydf = df["Churn"]
```

```
# Print X variables to verify changes made in data transformation steps
Xdf
```

CaseOrder	Zip	Children	Age	Income	Outages	Equipment_Failures	Techie	Modem	Tablet	Tenure	...	Marital_Married	Marital_Never	Marital_Separated	Mari
1	99927	0	68	28561	7	1	0	1	1	6	...	0	0	0	
2	48661	1	27	21704	11	1	1	0	1	1	...	1	0	0	
3	97148	4	50	9609	10	1	1	1	0	15	...	0	0	0	
4	92014	1	48	18925	14	0	1	0	0	17	...	1	0	0	
5	77461	0	83	40074	8	1	0	1	0	1	...	0	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9996	05758	3	23	55723	9	0	0	1	1	68	...	1	0	0	
9997	37042	4	48	34129	6	0	0	0	0	61	...	0	0	0	
9998	79061	1	48	45983	6	0	0	0	0	47	...	0	1	0	
9999	30117	1	39	16667	12	0	0	0	1	71	...	0	0	1	
10000	30523	1	28	9020	11	0	0	1	0	63	...	0	1	0	

10000 rows × 29 columns

```
# View Y variables to verify changes made in data transformation steps
Ydf
```

```
CaseOrder
1        0
2        1
3        0
4        0
5        1
        ..
9996     0
9997     0
9998     0
9999     0
10000    0
Name: Churn, Length: 10000, dtype: category
Categories (2, int64): [0, 1]
```

```
# Identify X and y for train/test split
X = Xdf
y = Ydf
# Split the data into train and test sets, 70% train 30% test, maintain proportions by stratifying the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, test_size=0.3, random_state = 42, stratify = y)
```

```
# Save each data set to CSV
X_train_df = pd.DataFrame(X_train)
X_test_df = pd.DataFrame(X_test)
y_train_df = pd.DataFrame(y_train)
y_test_df = pd.DataFrame(y_test)
X_train_df.to_csv('X_train.csv', index=False)
X_test_df.to_csv('X_test.csv', index=False)
y_train_df.to_csv('y_train.csv', index=False)
y_test_df.to_csv('y_test.csv', index=False)
```

```
# Normalize/scale the features for KNN
s_X = StandardScaler()
X_train = s_X.fit_transform(X_train)
X_test = s_X.transform(X_test)
```

```
# Identify k values
k_values = list(range(1, 50))
scores = []
```

```
# Identify k values
k_values = list(range(1, 50))
scores = []
```

```
# Create KNN classifier with varying values of k
for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k, p=2, metric='euclidean')
# Compute cross-validation and mean score
score = np.mean(cross_val_score(knn, X_train, y_train, cv=5))
scores.append(score)
# Find the index of the best score for k value
best_index = np.argmax(scores)
best_k = k_values[best_index]
```

```
# Train final classifier
final_knn = KNeighborsClassifier(n_neighbors=best_k, p=2, metric='euclidean')
final_knn.fit(X_train, y_train)
```

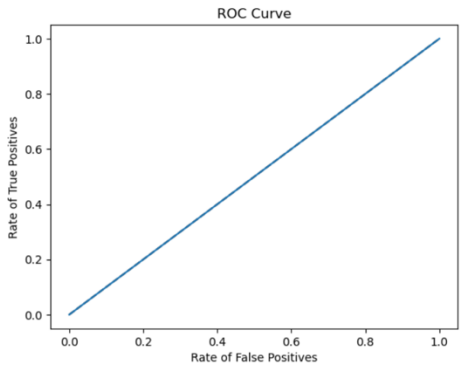
```
▾ KNeighborsClassifier
KNeighborsClassifier(metric='euclidean', n_neighbors=1)
```

```
# Make predictions using the final model
y_pred = final_knn.predict(X_test)
```

```
# Evaluate the model using accuracy, f1, confusion matrix and roc/auc
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
cmat = confusion_matrix(y_test, y_pred)
roc_auc = roc_auc_score(y_test, final_knn.predict_proba(X_test)[:, 1])
# Print scores
print("Accuracy:", accuracy)
print("F1 Score:", f1)
print("Confusion Matrix:\n", cmat)
print("ROC AUC:", roc_auc)
```

```
Accuracy: 0.7733333333333333
F1 Score: 0.5663265306122448
Confusion Matrix:
 [[1876  329]
 [ 351  444]]
ROC AUC: 0.7046421084156932
```

```
# Plot ROC Curve
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr)
plt.xlabel('Rate of False Positives')
plt.ylabel('Rate of True Positives')
plt.title('ROC Curve')
plt.show()
```



D3. Provide the code used to perform the classification analysis

In addition to being seen in the images above, here are the scripts used for this portion of the analysis:

```
# Begin analysis by separating the X variables (features) from the Y variable (target)
Xdf = df.drop(["Churn"], axis=1)
Ydf = df["Churn"]

# Print X variables to verify changes made in data transformation steps
Xdf

# View Y variables to verify changes made in data transformation steps
Ydf

# Identify X and y for train/test split
X = Xdf
y = Ydf
# Split the data into train and test sets, 70% train 30% test, maintain proportions by stratifying
    the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, test_size=0.3,
    random_state = 42, stratify = y)

# Save each data set to CSV
X_train_df = pd.DataFrame(X_train)
X_test_df = pd.DataFrame(X_test)
y_train_df = pd.DataFrame(y_train)
y_test_df = pd.DataFrame(y_test)
X_train_df.to_csv('X_train.csv', index=False)
X_test_df.to_csv('X_test.csv', index=False)
```

```
y_train_df.to_csv('y_train.csv', index=False)
y_test_df.to_csv('y_test.csv', index=False)

# Normalize/scale the features for KNN
s_X = StandardScaler()
X_train = s_X.fit_transform(X_train)
X_test = s_X.transform(X_test)

# Identify k values
k_values = list(range(1, 50))
scores = []

# Create KNN classifier with varying values of k
for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k, p=2, metric='euclidean')
# Compute cross-validation and mean score
score = np.mean(cross_val_score(knn, X_train, y_train, cv=5))
scores.append(score)
# Find the index of the best score for k value
best_index = np.argmax(scores)
best_k = k_values[best_index]

# Train final classifier
final_knn = KNeighborsClassifier(n_neighbors=best_k, p=2, metric='euclidean')
final_knn.fit(X_train, y_train)

# Make predictions using the final model
y_pred = final_knn.predict(X_test)

# Evaluate the model using accuracy, f1, confusion matrix and roc/auc
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
cmat = confusion_matrix(y_test, y_pred)
roc_auc = roc_auc_score(y_test, final_knn.predict_proba(X_test)[:, 1])

 #print scores
print("Accuracy:", accuracy)
print("F1 Score:", f1)
print("Confusion Matrix:\n", cmat)
print("ROC AUC:", roc_auc)

# Plot ROC Curve
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr)
plt.xlabel('Rate of False Positives')
```

```
plt.ylabel('Rate of True Positives')
plt.title('ROC Curve')
plt.show()
```

Part V: Data Summary and Implications

E1. Accuracy and AUC of classification model

As seen in the above screengrabs, the accuracy score has a value of 0.773. The area under the curve value is 0.7046. The accuracy score indicates that the model correctly classified around 77.3% of the instances in the dataset. This seems promising, as it seems that the model is more accurate than not for most instances.

When taking the AUC score into consideration, the model's efficiency is still quite promising. Before using the model for any business purposes, it would be vital to review the AUC score in comparison to the accuracy score. With an AUC value of 0.7046, this would indicate that the model discriminates between positive and negative cases better than random chance or guessing.

E2. Results and implications of classification analysis

As previously learned in D208, the F1-score is the "harmonic mean" of precision and recall. It evaluates false positives and false negatives and is used for imbalanced datasets. The higher the F1 score, the better the balance is between precision and recall. (2023, Kumar) In this case it is 0.5663, which would suggest an appropriate balance between the precision and recall. The confusion matrix results give information about the model's classification results, there are four values, 1876, 329, 351 and 444. There were 1,876 instances that were correctly predicted as negative (customer has not churned). 329 instances of false negatives, in which the instances were incorrectly predicted as negative. 351 instances were incorrectly predicted as negative, otherwise known as false negatives. 444 instances correctly predicted as positive (customer churned).

E3. One limitation of the data analysis

One limitation of this analysis is that of similarity to the limitations discussed in previous assessments. Specifically, the dataset size. Fortunately, the model rendered more accurate and dependable than previous models. At first look, the size of the dataset during this method seems to not be an issue this time around. Training models and making predictions with said models requires appropriate discrimination between instances. It is important to note that with such a small dataset, there is the possibility of sampling bias. Using this model with 10,000 data points to potentially make predictions for a customer base of 100,000 could cause skewed analysis and be more harmful than helpful for business.

E4. Recommended course of action

The classification results are promising, overall implicating that the model is better at predicting than random chance. These results are much better than previous models and utilizing the k-nearest neighbor method may prove beneficial to the business. However, the model could still be improved to be more accurate. For instance, 351 instances were missed cases. Missed cases cost the business money and customers, getting this number down to ensure more accuracy could make the model much more effective. As such, my recommendation would be to establish a feedback loop to gather actual outcomes of the predictions. Over time this data would be used to reassess and retrain the model to make improvements. The feedback loop would continuously update and improve the model and the business could highly benefit from its predictions.

Part VI: Demonstration

F. Panopto Video

The Panopto video recorded for this assessment can be found in the corresponding folder for this course.

G. List the web sources used to acquire data or segments of third-party code

Chelaru, Mihai. (2018, October 20). Implementing ROC Curves for K-NN machine learning algorithm using python and Sciki Learn. Retrieved from https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-algorithm-using-python-and-sci

Churchill, Briana. (2023, July 14). *Performance Assessment: Exploratory Data Analysis (OEM2).* Assignment for MS Data Analytics Course D207. Western Governors University.

Churchill, Briana. (2023, September 3). *Performance Assessment: Predicting Modeling Task 2.* Assignment for MS Data Analytics Course D208. Western Governors University.

Datacamp. (2023, February). K-Nearest Neighbors (KNN) Classification with scikit-learn. Retrieved from https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn

Simplilearn. (2018, June 8). KNN Algorithm In Machine Learning | KNN Algorithm Using Python | K Nearest Neighbor | Simplilearn. [Video file]. YouTube. https://www.youtube.com/watch?v=4HKqjENq9OU

H. Acknowledge sources

Kumar, A. (2023, Mar 17). *Accuracy, Precision, Recall & F1-Score – Python Examples.* https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/

Scikit-learn. (2011). Nearest Neighbors. In Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830. https://scikit-learn.org/stable/modules/neighbors.html

Shmueli, Galit. (2015 August 19). Categorical predictors: how many dummies to use in regression vs. k-nearest neighbors. BzST | Business Analytics, Statistics, Teaching. https://www.bzst.com/2015/08/categorical-predictors-how-many-dummies.html

Simplilearn. (2018, June 8). KNN Algorithm In Machine Learning | KNN Algorithm Using Python | K Nearest Neighbor | Simplilearn. [Video file]. YouTube. https://www.youtube.com/watch?v=4HKqjENq9OU

Stack Exchange. (2020). K-Nearest Neighbor Classifier Best K Value. Data Science Stack Exchange. https://datascience.stackexchange.com/questions/69415/k-nearest-neighbor-classifier-best-k-value

Straw, E. (2023). *D209 Data Mining 1 Task 1 Cohort* [Unpublished document]. Western Governors University.

Straw, E. (2023). *Dr. Straw's Tips for Success in D209* [Unpublished document]. Pg 19. Western Governors University.