D209 - Data Mining I

#### **Performance Assessment - Task 1: Classification Analysis**

Samuel Spencer

Western Governors University

Dr. Keiona Middleton

November 16, 2022

**Customer Churn Problem**

According to our data dictionary, “Customer churn is defined as the percentage of customers who stopped using a provider’s product or service during a certain time frame. In this highly competitive market, some telecommunications industries can experience average annual churn rates as high as 25 percent. Given that it costs 10 times more to acquire a new customer than to retain an existing one, customer retention has now become even more important than customer acquisition.”

There are several factors that contribute to customer churn that are well known such as:

* Price
* Competition
* Value
* Poor customer service

But my goals for this project is to analyze the variables that I’ll touch on later in the project and predict whether the customer is at a higher risk of churn, based on their profile.

## Part I - Research Question

### A1: Proposal of Question

The central research question addressed by this analysis is to determine:

Can we determine if a customer is considered a techie based on demographic information provided in the given data set?

### A2: Defined Goal

Although churn is not being addressed in this research as far as a variable we are including, the purpose of the research is ultimately to reduce churn with our telecom company. My intention with this analysis is to conclude whether we can predict and classify individuals based on the outlined variables (XXXX). To classify our customer, we will be using k-NN or K Nearest Neighbor.

## Part II - Method Justification

### B1: Explanation of Classification Method

K-Nearest Neighbors or commonly referred to as KNN is a rather simple algorithm, but it is equally as impactful as other algorithms. KNN works by classifying various new data points based on distance. With a KNN model, it assumes that any data with similar traits would be closer together, or neighbors. You can visualize this in the real world as well, if you go to the grocery store, items that share similar data would be grouped together such as fruit, veggies, grain etc. all these items are arranged in proximity to each other in grocery stores.

My expected outcome is classifying the data based on distance to predict if someone is a techie, Y/N

### B2: Summary of Method Assumptions

### KNN models assume that data that is close to each other are highly similar, and a strong relationship is likely on the other hand, if data is far apart, they are nothing alike.

### B3: Packages/Libraries List

The following Python libraries and packages will be utilized in this analysis:

* Pandas
* NumPy
* KNN Classifier
* MatPlotLib
* Seaborn

**Pandas**

* Pandas is a standard import for most data analysis. This will assist us in cleaning and filtering the data.

**KNeighborsClassifier**

* For our analysis, this package will be used to test the KNN algorithm.

**NumPy**

* NumPy allows us to work with arrays of data, and it works in conjunction with pandas for manipulation of mathematical functions.

**Matplotlib & Seaborn**

* Both tools are power visualization tools. What they do in our analysis is allows us to visually represent our data.

Part III - Data Preparation

### C1: Data Preprocessing

Before I can do analysis with our categorical variables, they’ll need to be encoded to numerical values.

### C2: Dataset Variables

The variables that are used for our analysis are listed below:

|  |  |
| --- | --- |
|  | DataType |
| Tenure | categorical |
| Age | numeric |
| Contacts | numeric |
| Income | categorical |
| Bandwidth\_GB\_Year | categorical |
| Yearly\_equip\_failure | numeric |
| Tablet | numeric |
| Gender | numeric |
| Techie | numeric |

### C3: Steps for Analysis

These are the steps taken for us to have a clean data set for our analysis:

* **Step 1 - Load in libraries and dataset**
  + This is the initial step involved with all data analysis.it involves importing the necessary libraries so that we can manipulate and visualize the data. *The necessary libraries were mentioned at B3.*

# import standar packages

import numpy as np

import pandas as pd

from sklearn.neighbors import KNeighborsClassifier

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

# Importing the data

df = pd.read\_csv('/Users/Spence604/Library/CloudStorage/OneDrive-WesternGovernorsUniversity/Docs/Churn Data/churn\_clean.csv')

* **Step 2 – Check for null and dupes**
  + This is a critical step in data cleaning, we must locate and normalize any null or duplicates if there are any, but there aren’t any in our data set

print("Are there any columns with null values: " + str(df.isnull().all(axis=1).any()))

df.duplicated()

* **Step 3 – Data frame with only necessary variables**
  + We need to drop variables that are irrelevant to our analysis and create a new dataframe with just necessary variables

newdf = df[['Age', 'MonthlyCharge', 'Yearly\_equip\_failure','Churn']].copy()

#view new dataframe

Newdf.head()

* **Step 4 -Export new df**
* **Step 5 - Solve for optimal n\_neighbors**
  + Lastly, in order to optimize the model, an algorithm is used to test using a n\_neighbors value of 1-20. A plot is then generated to view the knn.score value on the y-axis and the n\_neighbors value on the x-axis to identify the optimal value.

### C4: Cleaned Dataset

Please see attached, cleaned dataset included in Task 1 submission.

## Part IV - Analysis

### D1: Splitting the Data

We will use the module train\_test\_split from the sklearn library to split the dataset. This tool makes the process of splitting quite simple. The split percentage will be 80/20 with 80% of the total observations utilized in training the model, while the remaining 20% will be reserved for testing the model. Additionally, as described in the code below, a random seed of 42 was chosen for the randomization. Seed 42 was chosen at random as well.

# Declare X as df1 Initial\_days and VitD\_levels  
# without target feature ReAdmis  
X = df1.drop(['ReAdmis'],  
 axis=1)  
  
# Declare y as df1 Readmis target feature   
y = df1.ReAdmis  
  
# Split data using 80/20 split and seed 42  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=.2, random\_state=42)  
  
# Set kNN k=7 and fit to training data  
knn = KNeighborsClassifier(7)  
knn.fit(X\_train,  
 y\_train)  
  
# Test kNN on test data  
knn.score(X\_test,  
 y\_test)

### D2: Output & Intermediate Calculations

The present analysis consists primarily of the algorithm and determining the optimal value for n\_neighbors. After first preparing the data for analysis, minor EDA steps, such as preparing descriptive statistics and visualizing the data through the use of boxplots on the selected features, are taken to gain a bit of perspective prior to modelling. After brief EDA, a preliminary attempt to fit the training data to the model is performed. The value of is arbitrarily selected as a first step prior to running further analysis on optimizing . Finally, the optimization analysis is performed Below are examples of the above-mentioned intermediate steps:

### D3: Code Execution

#### Step 1 - Load in libraries and dataset

# Load in libraries needed  
import pandas as pd  
from sklearn.datasets import load\_iris  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import confusion\_matrix, roc\_curve, auc  
import plotly.express as px  
import matplotlib.pyplot as plt  
  
# Read medical dataset into dataframe as df  
df = pd.read\_csv('./data/medical\_clean.csv')  
  
# Show summary of dataframe including dtypes and counts  
df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 10000 entries, 0 to 9999  
Data columns (total 50 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 CaseOrder 10000 non-null int64   
 1 Customer\_id 10000 non-null object   
 2 Interaction 10000 non-null object   
 3 UID 10000 non-null object   
 4 City 10000 non-null object   
 5 State 10000 non-null object   
 6 County 10000 non-null object   
 7 Zip 10000 non-null int64   
 8 Lat 10000 non-null float64  
 9 Lng 10000 non-null float64  
 10 Population 10000 non-null int64   
 11 Area 10000 non-null object   
 12 TimeZone 10000 non-null object   
 13 Job 10000 non-null object   
 14 Children 10000 non-null int64   
 15 Age 10000 non-null int64   
 16 Income 10000 non-null float64  
 17 Marital 10000 non-null object   
 18 Gender 10000 non-null object   
 19 ReAdmis 10000 non-null object   
 20 VitD\_levels 10000 non-null float64  
 21 Doc\_visits 10000 non-null int64   
 22 Full\_meals\_eaten 10000 non-null int64   
 23 vitD\_supp 10000 non-null int64   
 24 Soft\_drink 10000 non-null object   
 25 Initial\_admin 10000 non-null object   
 26 HighBlood 10000 non-null object   
 27 Stroke 10000 non-null object   
 28 Complication\_risk 10000 non-null object   
 29 Overweight 10000 non-null object   
 30 Arthritis 10000 non-null object   
 31 Diabetes 10000 non-null object   
 32 Hyperlipidemia 10000 non-null object   
 33 BackPain 10000 non-null object   
 34 Anxiety 10000 non-null object   
 35 Allergic\_rhinitis 10000 non-null object   
 36 Reflux\_esophagitis 10000 non-null object   
 37 Asthma 10000 non-null object   
 38 Services 10000 non-null object   
 39 Initial\_days 10000 non-null float64  
 40 TotalCharge 10000 non-null float64  
 41 Additional\_charges 10000 non-null float64  
 42 Item1 10000 non-null int64   
 43 Item2 10000 non-null int64   
 44 Item3 10000 non-null int64   
 45 Item4 10000 non-null int64   
 46 Item5 10000 non-null int64   
 47 Item6 10000 non-null int64   
 48 Item7 10000 non-null int64   
 49 Item8 10000 non-null int64   
dtypes: float64(7), int64(16), object(27)  
memory usage: 3.8+ MB

#### Step 2 - Subset data & initial EDA

# Sub-select for pre-determined features  
df1 = df.loc[:, ['Initial\_days',  
 'VitD\_levels',  
 'ReAdmis']]  
  
# Summary statistics of the two predictor features  
df1.describe()

Initial\_days VitD\_levels  
count 10000.000000 10000.000000  
mean 34.455299 17.964262  
std 26.309341 2.017231  
min 1.001981 9.806483  
25% 7.896215 16.626439  
50% 35.836244 17.951122  
75% 61.161020 19.347963  
max 71.981490 26.394449

# Show boxplots of Initial\_days and VitD\_levels features  
df1boxplts, axes = plt.subplots(nrows=1, ncols=2)  
df1.boxplot('Initial\_days', ax=axes[0])  
df1.boxplot('VitD\_levels', ax=axes[1])

Chart, box and whisker chart

Description automatically generated

# Show scatterplot of Initial\_days and VitD\_levels  
# with ReAdmis as color to identify categories  
fig = px.scatter(df.sample(n=200,  
 random\_state=42),  
 x='Initial\_days',  
 y='VitD\_levels',  
 color='ReAdmis',  
 template='seaborn',  
 width=800,  
 height=500)  
fig.show()

#### Chart, scatter chart Description automatically generated

#### Step 3 - Prepare subset data for analysis

# Set ReAdmis to category type for easier  
# handling with model and later visualization  
df1.ReAdmis = df1.ReAdmis.astype('category')  
df1.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 10000 entries, 0 to 9999  
Data columns (total 3 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Initial\_days 10000 non-null float64   
 1 VitD\_levels 10000 non-null float64   
 2 ReAdmis 10000 non-null category  
dtypes: category(1), float64(2)  
memory usage: 166.3 KB

# Declare X as df1 Initial\_days and VitD\_levels  
# without target feature ReAdmis  
X = df1.drop(['ReAdmis'],  
 axis=1)  
  
# Declare y as df1 Readmis target feature   
y = df1.ReAdmis  
  
# Split data using 80/20 split and seed 42  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=.2, random\_state=42)

#### Step 4 - Set kNN, fit, and test

# Set kNN k=7 and fit to training data  
knn = KNeighborsClassifier(7)  
knn.fit(X\_train,  
 y\_train)  
  
# Test kNN on test data  
knn.score(X\_test,  
 y\_test)

0.982

# Set kNN k=7 and fit to training data  
knn = KNeighborsClassifier(7)  
knn.fit(X\_train,  
 y\_train)  
  
# Test kNN on test data  
knn.score(X\_test,  
 y\_test)

0.982

#### Step 5 - Solve for optimal n\_neighbors

# Loop through attempts to fit kNN using k  
# value of 1 through 20 to identify optimal k  
num\_k = []  
knnscore = []  
for i in range(1,21):  
 num\_k.append(i)  
 knn = KNeighborsClassifier(n\_neighbors=i)  
 knn.fit(X\_train,y\_train)  
 knnscore.append(knn.score(X\_test,  
 y\_test))  
  
# Create df of each k value and corresponding score  
pltscore = pd.DataFrame({'num\_k': num\_k,  
 'knnscore': knnscore})  
  
# Plot score by k value  
fig = px.line(pltscore,  
 x='num\_k',  
 y='knnscore',  
 width=700,  
 height=500)  
fig.show()

Chart, scatter chart

Description automatically generated

As demonstrated above, it seems that a of 13 is a decent value to stick with based on the scoring method we've implemented. We will perform additional analysis on accuracy below.

## Part V - Data Summary & Implications

### E1: Accuracy & AUC

The accuracy score of this analysis is 0.7312. This means that there is an 73.12% chance of predicting correctly whether a customer will churn or not based on the variables we inputted, being Age, Tenure, Bandwidth Usage, and Monthly Charge.

### E2: Results & Implications

As mentioned above, our model performed incredibly well in classifying readmission based on the features Initial\_days and VitD\_levels. As a result, our analysis supports the rejection of the null hypothesis () which states:

*The recorded vitamin D level (VitD\_levels) and length of initial hospitalization (Initial\_days) features from the medical readmission dataset have no statistically significant predictive power to classify a given patient's readmission status (ReAdmis).*

Furthermore, our analysis supports the acceptance of the alternate hypothesis () which states:

*The recorded vitamin D levels (VitD\_levels) and length of initial hospitalization (Initial\_days) features from the medical readmission dataset do classify a given patient's readmission status (ReAdmis) in a statistically significant way.*

Therefore, our initial research question (Can a patient's readmission (ReAdmis) status (Yes/No) be accurately classified given their recorded vitamin D level (VitD\_levels) and number of days initially hospitalized (Initial\_days)?) is answered in the affirmative based on the results of our above analysis.

It seems evident that the selected features provide for the ability to classify with a high level of accuracy the readmission status of a given patient. With regard to implications, though the analysis does not explicitly provide the insight into whether or not these features are also predictive of readmission status, it is reasonable to preliminarily assume that as a patient's initial stay lengthens and their vitamin D level is relatively low, the probability that patient will be readmitted increases. This would imply that, if the goal is to reduce readmissions a much as possible, a sensible course of action could include measures to shorten initial length of stay as much as possible and support vitamin D levels if low.

### E3: Limitations

One limitation of the analysis is the nature of the distribution of several features, including initial length of hospitalization (Initial\_days). As the distribution of this feature is not normal and is heavily binomial, the reliability of the analysis becomes more difficult to ascertain. In terms of assumptions, normal distribution is not indicated for and therefore the classification modelling should not suffer from a non-normal distribution. However, another limitation of the Initial\_days feature is that it is somewhat obvious that the longer a patient is initially hospitalized, the more likely they are to be readmitted. Though, often presumptions are proven incorrect upon further analysis, therefore this analysis is at least confirmatory.

### E4: Course of Action

As a result of the analysis performed, it is recommended that the organization take actions to limit a patient's initial length of stay and support a patient's vitamin D level if low. Leadership should consult with physicians on safe and effective ways to shorten hospitalizations and address low vitamin D levels. It is possible this would lead to lower rates of readmission. Additionally, further analysis and inquiry is recommended to better understand and enhance the data gathering process. In particular, investigating the integrity and reliability of the Initial\_days feature is recommended. Ensuring that the underlying data is as reliable as possible will lead to better and more useful analysis in the future.

### H: Sources

Hajian-Tilaki, K. P. (2013). Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation. *Caspian J Intern Med, 4*(2), 627–635.

Pedregosa, F. V. (2011, October). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Powers, D. M. (2008). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies, 2*(1).