**Part I: Research Question**

1. Describe the purpose of this data mining report by doing the following:
2. Using K-Nearest Neighbors analysis (KNN), can patients’ readmittance be predicted based on information available at time of admission? Information available at time of admission does not include data on a patient’s initial stay. The variables that fit these criteria include City, State, County, Zip, Lat, Lng, Population, Area, TimeZone, Job, Children, Age, Income, Marital, Gender, VitD\_levels, Soft\_drink, Initial\_admin, HighBlood, Stroke, Overweight, Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic\_rhinitis, Reflux\_esophagitis, and Asthma.
3. One goal of the analysis is to predict a patient’s readmittance based on available information at the time of admission. This classification will allow hospitals to better prioritize the care of admitted patients and potentially reduce readmission rates.

**Part II: Method Justification**

1. Explain the reasons for your chosen classification method from part A1 by doing the following:
2. KNN classification works by comparing data points in the validation or test sets with data points in the training set. The test assumes data points with the same label are grouped together and uses the labels of the closest data points to determine which label the validation or test set data point should have. This process is where the name K Nearest Neighbors comes from, as the model compares each data point to k, a constant, of the closest data points in the training set, its neighbors, and then predicts that the new data point will have the same label as most of these neighbors. We expect the model to reliably group patients by their complication risk and be able to accurately predict the complication risk of unlabeled data. If the model has poor predictive power, that will indicate that patient readmission is not grouped into identifiable groups in the training set.
3. KNN assumes that the values of the target variable are separable into distinct groups based on the predictor variables. In three-dimensional models, where there are three predictor variables and one target variable, the three-dimensional graph of the predictor variables will show that, for each label of the categorical target variable, there is one distinct grouping. By coloring the points based on their label, we assume we can draw boundaries to divide each color from any other color with a reasonable level of accuracy.
4. In Python, we will use the Pandas library to create data frames, divide data, and join data. This library allows for easy access to data based on index, as well as selection of data based on values within variables. This library is a dependency of the modeling library as well. The Numpy library is used for computations, including mean, standard deviation, square roots, and more. This library also provides tools for working with multiple slices of the same Pandas data frame or series and is a dependency of the modeling library. The Sklearn library, specifically the preprocessing and neighbors classes, is included to encode categorical variables into numeric values, normalize numeric data, split the data into training and test data, and create a KNN model on the data.

**Part III: Data Preparation**

1. Perform data preparation for the chosen data set by doing the following:
2. After the categorical variables are numerically encoded, we will use Sklearn’s MinMaxScaler method to normalize the data. This helps the model to properly interpret the values without being biased to variables with larger ranges of values.
3. We will use the following variables as predictor variables:

City (Categorical), State (Categorical), County (Categorical), Zip (Continuous), Lat (Continuous), Lng (Continuous), Population (Continuous), Area (Categorical), TimeZone (Categorical), Job (Categorical), Children (Continuous), Age (Continuous), Income (Continuous), Marital (Categorical), Gender (Categorical), VitD\_levels (Continuous), Soft\_drink (Categorical), Initial\_admin (Categorical), HighBlood (Categorical), Stroke (Categorical), Overweight (Categorical), Arthritis, Diabetes (Categorical), Hyperlipidemia (Categorical), BackPain (Categorical), Anxiety (Categorical), Allergic\_rhinitis (Categorical), Reflux\_esophagitis (Categorical), and Asthma (Categorical).

1. We built the following data preparation steps from the preparation function first presented in Task One of Predictive Modeling.

We must divide the data into four data frames, one for numeric variables, another for non-binary categorical variables, one for binary categorical variables, and another for ordinal categorical variables. The numeric variables will have their outliers defined and offending observations dropped from the data set and the remaining values normalized with sklearn’s MinMaxScaler. Identification and removal of outliers is completed in the block of code highlighted in blue. Normalization is accomplished later in the process in the code highlighted in purple. The non-binary categorical variables will be one-hot encoded using the pandas library’s get\_dummies method, highlighted in orange below. The binary categorical variables will be encoded manually, such that values of "Yes" become one and values of "No" become zero. This task is accomplished in the for loop highlighted in yellow below. The ordinal categorical variables will be manually encoded so that values representing lesser risk or population density are given lower values while values indicating higher risk or population density are given higher values. The dictionary of values to encode and the replacement of values occurs in the code highlighted in green below. Finally, these four data frames are reunited in the code highlighted in red. The full code used to generate these transformations is provided below:

def meddata\_preprocessing(data\_num, data\_enc, data\_yn, target):

for i in range(len(data\_num.columns)):

mean, std = np.mean(data\_num.iloc[:,i]), np.std(data\_num.iloc[:,i])

upper, lower = mean + 3 \* std, mean - 3 \* std

drop = [inx for inx, x in enumerate(data\_num.iloc[:, i]) if x < lower or x > upper]

for d in drop:

if d in data\_num.index:

data\_num = data\_num.drop(d)

#One-hot encode categorical variables

data\_enc = pd.get\_dummies(data\_enc, prefix=data\_enc.columns, drop\_first=True)

for col in range(len(data\_yn.columns)):

for inx, val in enumerate(data\_yn.iloc[:, col]):

if val == 'Yes':

data\_yn.iloc[inx, col] = 1

else:

data\_yn.iloc[inx, col] = 0

#Determine levels of ordinal variables

scale\_mapper = {

"Area" : {

"Rural" : 0,

"Suburban" : 0.5,

"Urban" : 1

},

"Initial\_admin" : {

"Emergency Admission" : 1,

"Observation Admission" : 0.5,

"Elective Admission" : 0

},

"Complication\_risk" : {

"Low" : 0,

"Medium" : 0.5,

"High" : 1

}

}

#Replace values with numerical equivalents specified above

for col in data\_ord.columns:

data\_ord[col] = data\_ord[col].copy().replace(scale\_mapper[col])

mm = MinMaxScaler() #Instantiate the MinMaxScaler method

data\_num[data\_num.columns] = mm.fit\_transform(data\_num) #Normalize the data

#The use of an inner join preserves the dropping of rows performed on data\_num

#The data\_yn dataframe is converted to a numeric datatype, int32, before joining

data\_clean = data\_num.copy().join(data\_enc, how='inner').join(data\_yn.astype('int32'), how='inner').join(data\_ord, how='inner').join(data\_tar, how='inner')

return data\_clean

1. A copy of the cleaned data set is provided as data\_clean.csv.

**Part IV: Analysis**

1. Perform the data analysis and report on the results by doing the following:
2. The sklearn train\_test\_split method is used to divide the data set into training data and test data. We divide the data into the data sets X-train, X\_test, y\_train, and y\_test, where the prefix X represents a matrix of predictor variables, the prefix y represents a vector of target variables, the suffix train represents a dataset used for training, and the suffix test represents a dataset used for model validation and testing. We divide the data so that 80% of the data is retained in the training sets while 20% is reserved for testing by setting the test\_size parameter equal to 0.2. In order for the model to be reproducible, the random\_state parameter is set to 42. A copy of the divided data is provided in the csv files X\_train, X\_test, y\_train, and y\_test attached to this submission.
3. A K-nearest neighbors (KNN) analysis was employed to analyze the data. KNN predicts the classification of input based on the labels of the k-closest observations in the training data, determined by the values of the predictor variables. This style of analysis assumes that the data can be grouped or clustered, with observations having the same target label also having similar values in predictor variables. The sklearn method KNeighborsClassifier is used to implement a KNN analysis. While we can adjust parameters of the KNeighborsClassifier to achieve higher accuracy, we used the default values for this analysis.
4. X = data\_clean.drop(labels=['ReAdmis'], axis=1).copy()

y = data\_clean['ReAdmis'].copy()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

knn = KNeighborsClassifier()

knn.fit(X\_train, y\_train)

y\_pred = knn.predict(X\_test)

print(classification\_report(y\_test, y\_pred))

print(confusion\_matrix(y\_test, y\_pred))

prob = knn.predict\_proba(X\_test)

roc\_auc\_score(y\_test.values, prob[:, 1], multi\_class='ovo')

**Part V: Data Summary and Implications**

1. Summarize your data analysis by doing the following:
2. The model has an accuracy score of 0.57, as shown in the screenshot below. This accuracy score implies that the model does not predict the correct labels for the testing data well. The confusion matrix confirms this suspicion, showing that 1105 observations in the test data were correctly predicted while 345 positive, “Yes” observations were falsely predicted as negatives and 478 negative, “No” observations were incorrectly predicted as positives.

The area-under-the-curve, or roc\_auc\_score as it is called in sklearn, is approximately 0.5068 for our model. A score of 0.5 implies that the model has no ability to differentiate between positive and negative classes while a score of zero implies the model is always predicting the opposite of the true class and a score of one implies perfect predictive ability (Narkhede, 2021). According to these interpretations, a score of 0.5 is the worst score possible as the model is performing about as well as you would expect if the model was randomly guessing the labels for each observation. The score for our model is very close to 0.5, implying that the model cannot group and differentiate between the positive and negative class.

1. The poor performance of our model does not mean nothing is learned. On the contrary, this model’s performance implies that patient readmittance cannot be reliably predicted based on the information available at time of admittance. From this, we can deduce that the characteristics of a patient’s stay are more influential on a patient’s readmittance than any pre-existing conditions or lifestyle characteristics.
2. One factor that limits our analysis is the selected variables. Because the KNN analysis attempts to group observations based on the values of their predictor variables, it is possible that there are identifiable parameters for grouping in another set of predictor variables. By providing additional information about patients at the time of admission, for example heart rate or blood pressure values, it is possible that a KNN analysis could prove fruitful; however, with the data available in this data set, KNN analysis is shown to be ineffective as a predictor of patient readmission.
3. Based on the lack of confidence we have in this model’s predictions, as discussed above, it is our recommendation that hospitals focus less on the patient’s history and lifestyle and more on providing quality care in a timely manner. The recommendations provided in the tasks for the previous course were shown to have greater effects on patient readmittance than the information available at time of admittance used in this analysis and the recommendations found in those analyses are more likely to reduce patient readmittance chance than any prioritization that could be provided by this analysis.

**Part VI: Demonstration**

1. <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=4362dce2-524e-4ae8-aea1-adb701593cec>
2. No outside sources were used to generate code for this project and all code was written personally by the student.
3. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

Narkhede, S. (2021, June 15). *Understanding AUC - roc curve*. Medium. Retrieved October 1, 2021, from https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.