Michael Segaline

D209 Data Mining

Dr. Elleh

1/15/2022

Abstract

Bottom Line Up Front: How accurately can K Nearest Neighbors (KNN) classify hospital readmission, given a set of set of variables?

A major hospital chain has a business problem: reducing patient readmission rates. Outside agencies put financial pressure on hospitals with high readmission rates. The dataset was scrubbed, standardized, and explored via K Nearest Neighbors supervised machine learning algorithm. Several variables will be selected and explored to notice any impact on the target variable “Readmission”. One goal of the data analysis is to hyper tune to the KNN model to find the optimum number “K” given the selected variables, yielding a most accurate model possible with the given dataset. The model will be scored then cross validated with sample strata to simulate how the model will perform on unseen data.

K Nearest Neighbors is easy to implement with few parameters to tune. KNN is a supervised Machine Learning algorithm that will take a dataset with labeled data and classify a categorical discrete variable. KNN can be used to for classification and regression problems. KNN assumes that similar things exist in close proximity (Towardsdatascience.com). KNN captures the idea of similarity by calculating the distance between points on a graph and classifies them by similarity. The variable ‘K’, (number of unlabeled points), will be initialized. The distances of the surrounding points to the unlabeled point will decide what the label should be based off the closer points. The expected outcome if ‘K’ =1, results in the model having the least amount of accuracy. “Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far” (towardsdatascience.com). One of the goals of this analysis is to find the most efficient number of neighbors “K”.

Python will be used because it has all the necessary statistical libraries to conduct the data analysis such as Sklearn. Sklearn has functions that can facilitate all stages of the Extract Load Transform process. Sklearn also contains several Machine Learning functions to include KNN. Sklearn has useful functions such as the StandardScalar(), for scaling the data. The GridSearchCV() function finds the optimum number of ‘K’ . The Pandas package will be used to convert big data into a useable dataset that can be easily manipulated. Lastly, the cross\_val\_score() method gives the Area Under the Curve scores for simulated unseen data. The best\_params\_ method will output the number of neighbors ‘K’ for the optimum parameter tuning. Calling the cross-validation function from Sklearn, will iterate the KNN algorithm several times on different slices or “Folds” of the dataset. Resulting in a ‘K’ that reduces the number of errors and improving the integrity of the model to unseen data. The outcome of the analysis will be a KNN model that is hyper-tunned to optimum classification performance given the medical dataset on an unseen dataset of the same variables.

To achieve this end state, the dataset will first be imported into the Python environment. A data preprocessing goal is to convert the categorical variables into machine readable binary representation. The variables to be analyzed are: Population (Continous), Children (Continous), Marital\_Married (Categorical), ReAdmis\_Yes (Categorical), HighBlood\_Yes (Categorical), Stroke\_Yes (Categorical) Overweight\_Yes (Categorical), Diabetes\_Yes(Categorical), Anxiety\_Yes (Categorical) Asthma\_Yes (Categorical). Second, all missing values will be dropped so they do not impact the model or the summary statistics; This is accomplished via the dropna() function. Third, unnecessary features will be dropped from the data set via the drop() function. Fourth, categorical variables will be converted into binary representation via the get\_dummies() function; get\_dummies() creates two columns for every category and drops the first column. Fifth, to remove outliers, the StandardScalar() function is called and the dataset is passed as an argument. Before creating the model, the cleaned dataset will be separated into two data frames, one for testing ‘X’ and the other for training ‘y’. This accomplished by calling the test\_train\_split() function which creates a tuple of four variables when passing ‘X’ and ‘y’ as arguments. This tuple creates the variables: ‘X\_train’, ‘X\_test’, ‘y\_train’, ‘y\_test’; In which 30% of the data was allocated to testing and 70% to training.

**#-- Importing all nessesary libraries --#**

**import statistics**

**import numpy as np**

**import seaborn as sns**

**from scipy import stats**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**from sklearn.neighbors import KNeighborsClassifier**

**from sklearn.model\_selection import train\_test\_split**

**import numpy as np**

**from sklearn.metrics import roc\_curve, roc\_auc\_score**

**import matplotlib.pyplot as plt**

**from sklearn.model\_selection import cross\_val\_score**

**from sklearn.metrics import confusion\_matrix**

**from sklearn.metrics import classification\_report**

**from sklearn.metrics import roc\_curve**

**from sklearn.model\_selection import GridSearchCV**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.preprocessing import StandardScaler**

**from sklearn.metrics import mean\_squared\_error**

**from sklearn import metrics**

**#-- Importing the dataset --#**

**med\_df\_raw = pd.read\_csv("/Users/lindasegalini/Desktop/WGU/New Program/D208 Predictive Modeling/medical\_clean.csv")**

**#-- Dropping missing values --#**

**med\_df\_raw.dropna()**

**#-- Dropping unessesary features --#**

**med\_df\_raw = med\_df\_raw.drop(columns = ['Gender','Complication\_risk','Initial\_admin','Additional\_charges','TotalCharge','Initial\_days','VitD\_levels','Age','CaseOrder','Allergic\_rhinitis','Reflux\_esophagitis','Doc\_visits','Full\_meals\_eaten','vitD\_supp','Soft\_drink','Customer\_id','Interaction','UID','City','State','County','Zip','Lat','Lng', 'Area','TimeZone','Job','Income','Services','Item1','Item2','Item3','Item4','Item5','Item6','Item7','Item8', 'Arthritis','Hyperlipidemia','BackPain'])**

**#-- Changing categorical to binary with get\_dummies() and dropping the first column.**

**med\_df = pd.get\_dummies(med\_df\_raw, drop\_first =True)**

**#-- Dropping the previously combined columns to keep the number of columns down and create a tidier dataset --#**

**med\_clean = med\_df.drop(columns = ['Marital\_Never Married','Marital\_Separated','Marital\_Widowed'])**

**#-- Filtering the data frame to remove the values exceeding 3 standard deviations --#**

**med\_remove\_df = med\_clean[(np.abs(stats.zscore(med\_df)) < 3).all(axis=1)]**

**#-- Displaying what rows were removed --#**

**med\_clean.index.difference(med\_remove\_df.index)**

**#-- Saving a copy of the cleaned dataset --#**

**med\_clean.to\_csv('/Users/lindasegalini/Desktop/WGU/New Program/D209 Data Mining/Submissions/KNN.csv')**

**#-- Split the data into CrossValidation and holdout sets --#**

**#-- Split the data into X & y --#**

**X = med\_clean.drop('ReAdmis\_Yes', axis = 1).values**

**y = med\_clean['ReAdmis\_Yes']**

**y = y.astype(int)**

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

**#-- Initailizing KNN Classifier --#**

**knn = KNeighborsClassifier(n\_neighbors = 48)**

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

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

**y\_pred**

**#-- The initial score of the model --#**

**#-- 'n\_neighbors': is the hyperparameter name --#**

**#-- The parameters the grid will cover --#**

**knn = KNeighborsClassifier()**

**param\_grid = {'n\_neighbors': np.arange(1, 50)} #-- if we specify more parameters, all will b tried--#**

**#-- grid variable--#**

**knn\_cv = GridSearchCV(knn, param\_grid, cv=10)**

**knn\_cv.fit(X,y)**

**#-- Generating the confusion matrix --#**

**from sklearn.metrics import confusion\_matrix**

**y\_true = y\_test # True values**

**cf\_matrix = confusion\_matrix(y\_true, y\_pred)**

**print("\n KNN \nTest confusion\_matrix")**

**sns.heatmap(cf\_matrix, annot=True, cmap='Blues')**

**plt.xlabel('Predicted', fontsize=12)**

**plt.ylabel('True', fontsize=12)**

**#-- Calculating False Positives (FP), False Negatives (FN), True Positives (TP) & True Negatives (TN)**

**FP = cf\_matrix.sum(axis=0) - np.diag(cf\_matrix)**

**FN = cf\_matrix.sum(axis=1) - np.diag(cf\_matrix)**

**TP = np.diag(cf\_matrix)**

**TN = cf\_matrix.sum() - (FP + FN + TP)**

**print('KNN Scores: ')**

**#-- Sensitivity, hit rate, recall, or true positive rate --#**

**TPR = TP / (TP + FN)**

**print("The True Positive Rate is:", TPR)**

**#-- Precision or positive predictive value --#**

**PPV = TP / (TP + FP)**

**print("The Precision is:", PPV)**

**#-- False positive rate or False alarm rate --#**

**FPR = FP / (FP + TN)**

**print("The False positive rate is:", FPR)**

**#-- False negative rate or Miss Rate --#**

**FNR = FN / (FN + TP)**

**print("The False Negative Rate is: ", FNR)**

**#-- Total averages --#**

**print("")**

**print("The average TPR is:", TPR.sum()/2)**

**print("The average Precision is:", PPV.sum()/2)**

**print("The average False positive rate is:", FPR.sum()/2)**

**print("The average False Negative Rate is:", FNR.sum()/2)**

**#-- Finds all the parameters --#**

**#--to retrieve the hyperparameters that perform the best along with the mean cross-validation**

**#-- score over that fold--#**

**knn\_cv.best\_params\_**

**#-- Compute and print metrics --#**

**print("Accuracy: {}".format(knn\_cv.score(X\_test, y\_test)))**

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

**print("Tuned Model Parameters: {}".format(knn\_cv.best\_params\_))**

**knn.score(X\_test,y\_test)**

**#-- Generating the ROC /AUC --#**

**fpr, tpr, \_ = metrics.roc\_curve(y\_true, y\_pred)**

**auc = metrics.roc\_auc\_score(y\_test, y\_pred)**

**#-- Plotting the ROC curve --#**

**plt.plot(fpr,tpr,label="AUC="+str(auc))**

**plt.ylabel('True Positive Rate')**

**plt.xlabel('False Positive Rate')**

**plt.legend(loc=4)**

**plt.show()**

Table

Description automatically generated

Chart

Description automatically generated

Chart, line chart

Description automatically generated

The final accuracy of the model is 63.7%. However, the Area Under the Curve scores are not very good considering the six different Cross Validation sets on “unseen” data scored around 50%; No better than flipping a coin. “When AUC is 0.5, it means the model has no class separation capacity whatsoever” (Towardsdatascience.com). Additionally, the Receiver Operator Characteristic ROC is probability curve that compares two bell-shaped distributions plotted on against the True Positive rate (‘x’ axis) and the False Positive Rate (y-axis). “When two curves don’t overlap at all means model has an ideal measure of separability. It is perfectly able to distinguish between positive class and negative class” (Towardsdatascience.com). If the distributions overlap at all, then type 1 and 2 errors can occur. In the case of this model, both distributions overlap completely, making one distribution. Indicated by the straight 45-degree line traveling from the lower left to upper right of the above graph.

In final analysis, the variables chosen don’t have much impact on hospital readmission. Furthermore, the optimum number of neighbors is 48, which seems like a rather large amount. Regardless of the number of neighbors, the model’s accuracy improved very slightly between 1 to 48 neighbors. Additionally, the ratio of testing to training data was tested and the highest accuracy came from 30% testing 70% training. The silver-lining is the elimination of the variables selected from further analysis since they do not contribute to an efficient KNN model. Overall, this is not an efficient model for predicting hospital readmission. A reasonable course of action is to explore different variables in relation to readmission via KNN.

Work cited

Harrison, O. (2018, September 10). *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. towardsdatascience.com. Retrieved from https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

Narkhede, S. (2021, June 15). *Understanding AUC - roc curve*. Medium. Retrieved January 16, 2022, from https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5