**D209 Performance Assessment Task 1**

**CLASSIFICATION ANALYSIS FOR MEDICAL DATA**

Fahim A. Akbar Student ID 001434895 Masters Data Analytics (January 1, 2021) Program Mentor: Lea Yoakem (877) 435-7948x6422 [fakbar3@wgu.edu](mailto:fakbar3@wgu.edu)

**Part I: Research Question**

**A. Purpose of the Data Mining Report**

**1. Question**

Classification analysis could be used to help answer the following: “Which patients are at a high risk of being readmitted?”

**2. Goal**

The goal of the data analysis is to accurately showcase which patients are at risk of readmission. An accuracy measure will be provided to ensure that the hospital leadership can be confident in the data. Weights for variables that the hospital should keep track of will be used to help the hospital leadership reduce readmissions.

**Part II: Method Justification**

**B. Reason for using K-Nearest Neighbor**

**1. Classification Method: K-Nearest Neighbor (KNN)**

For this analysis, the K-Nearest Neighbor method will be used. The K-Nearest Neighbor (KNN) method algorithm operates by assuming that similar things exist in close proximity. KNN finds the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label, or averages the labels. The “k” value is an arbitrary numeric value that is determined by the user, and it will drive the algorithm to look at that set number of data points around the data point in question to determine its label. The algorithm carries out by setting decision boundaries in the data points, and these decision boundaries are determined by continuous variables. The data point of interest is labeled according to where it falls within those decision boundaries. For this analysis, the algorithm will be conditioned via a training set of continuous variables and match that to the outcome of either readmitted or not. Once the algorithm can determine which continuous variables classify a patient as readmitted, the model performance will then be determined on the test data set. From this we can derive a model accuracy score. The goal of this analysis is to create an algorithm that will have a model accuracy of at least 95% without over or under fitting the model. Additional expected outcomes include test data being classified according to their closest neighbors (Harrison, 2019).

**2. Assumptions of KNN**The main assumption of the K-Nearest Neighbor method is that similar things will exist in close proximity to one another. This means that an unlabeled data point will exist within close proximity to a similar labeled data point, and can be classified based on the similarity in features. (Harrison, 2019).

**3. Python packages and libraries used**

Listed below are the Python packages and libraries that will be used, and how each item supports KNN classification analysis:

* Pandas – this standard import provides methods to read and visualize data. It also offers statistical tools to parse and score data.
* Numpy – this standard import provides methods to read and visualize data. It also offers statistical tools to parse and score data.
* Matplotlib – this package is used for data visualization and will provide more robust tools to visualize reports and data points
* Seaborn – this package will provide us descriptive and visually intuitive graphs, plots, and matrices
* Scikit-learn – this package will provides method and arguments for splitting, training, testing, and fitting data. This package also has arguments for predicting and classifying data as well as applying metrics for models

The code used to download these packages is provided in the “Python Package Code” document attached to this task submission.

**Part III: Data Preparation**

**C. Data preparation**

**1. Relevant data preprocessing goal**

The first step in preparing the data is to make sure that there are no missing data entries in any of the columns. Next, we will ensure that there are no duplicated columns or rows to further prevent dealing with repeated entries. For the predictive analysis, several columns in the dataset were deemed irrelevant and were subsequently dropped from the dataset (i.e latitude, longitude). The “yes/no” entries for the categorical variables will need to be converted to 1 and 0, respectively.

**2. Initial data set variables used to perform the classification analysis**

The purpose of this analysis is to determine which variables indicate a patient’s chance of readmission. Therefore, the categorical target variable will be ReAdmis, with responses of “Yes/No”. The following categorical variables that will be used in this analysis initially: Soft\_drink, HighBlood, Stroke, Overweight, Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic\_rhinitis, Reflux\_esophagitis, and Asthma. The following continued variables will be used in this analysis initially: Children, Age, Income, VitD\_Levels, Doc\_Visits, Full\_meals\_eaten, vitD\_supp, Initial\_days, TotalCharge, and Additional\_charges. The survey questions performed by the hospital may impact a patient’s decision to readmit themselves to the hospital if necessary. These are classified as discrete ordinal variables and are currently listed as Item1-Item8. They will be renamed to the following, respectively: Timely\_admis, Timely\_treat, Timely\_visits, Reliability, Options, Hrs\_treat, Courteous, and Active\_listen.

**3. Data Preparation Steps**

The first step in preparing the data is to make sure that there are no missing data entries in any of the columns. Next, we will ensure that none of the data in the columns is duplicated. We will make sure that none of the columns or rows are duplicated, to further prevent dealing with repeated entries. For the predictive analysis, several columns in the dataset were deemed irrelevant and were subsequently dropped from the dataset (i.e latitude, longitude). The “yes/no” entries for the categorical variables will need to be converted to 1 and 0, respectively. Additionally, the survey variables will be renamed so they can properly used in the analysis.

**4. Code used for cleaning the data sheet**

A copy of the cleaned dataset is provided with the task submission as a pdf. The code for cleaning and preparing the data set is provided below. The full code for the project is provided at the end of this document and is also attached as txt document titled “Full code used for D209 Task 1 Submission.” In addition, a PDF file of the Jupyter notebook created through Python to run the models is provided with the task submissions to showcase the data in its created environment.

**Code:**

**import** numpy **as** np

**import** pandas **as** pd

**from** sklearn **import** linear\_model

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**%matplotlib** inline

**import** sklearn

**from** sklearn **import** datasets

**from** sklearn **import** preprocessing

**from** sklearn.neighbors **import** KNeighborsClassifier

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

**from** sklearn.metrics **import** accuracy\_score

**from** sklearn.model\_selection **import** cross\_val\_score, train\_test\_split

**from** sklearn **import** metrics

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

pd**.**set\_option('display.max\_columns', **None**)

df **=** pd**.**read\_csv (r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_clean.csv')

df**.**head()

df**.**info()

*#check for missing data entries*

df**.**isna()**.**any()

*#check for any duplicate data entries in columns*

df[df**.**duplicated()]

*#check if any columns are duplicated - looking for False*

df**.**columns**.**duplicated()**.**any()

*#check if any rows are duplicated - looking for False*

df**.**duplicated()**.**any()

*# drop demographic data*

df **=** df**.**drop(['CaseOrder','Customer\_id','Interaction','UID','City','State','County','Zip','Lat','Lng','Population','Area','TimeZone','Job'], axis**=**1)

*# confirm that columns were dropped*

df**.**head()

*#overview of descriptive statistics*

df**.**describe()

*#rename survey columns for easier identification*

df**.**rename(columns**=**{'Item1':'Timely\_admis','Item2':'Timely\_treat','Item3':'Timely\_visits','Item4':'Reliability','Item5':'Options','Item6':'Hrs\_treat','Item7':'Courteous','Item8':'Active\_listen'},inplace**=True**)

*#verify that the survey columns were renamed correctly*

df**.**head()

*#change yes/no to 1/0*

df **=** df**.**replace(to\_replace **=** ['Yes','No'],value **=** [1,0])

*#verify that the values were changed*

df**.**head()

*# drop non-numeric variables that are less relevant*

df **=** df**.**drop(['Marital', 'Gender','Initial\_admin','Complication\_risk','Services'], axis**=**1)

*#verify that the non-numeric variables were dropped*

df**.**head()

df**.**to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_D209TASK1PREPARED.csv', index**=False**)

**Part IV: Analysis**

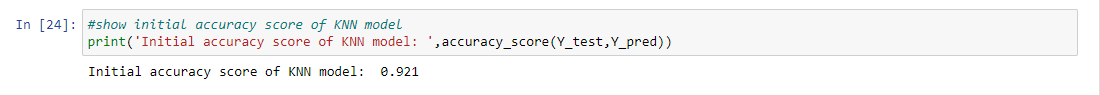
**D. Data Analysis**

**1. Training and Testing Sets**

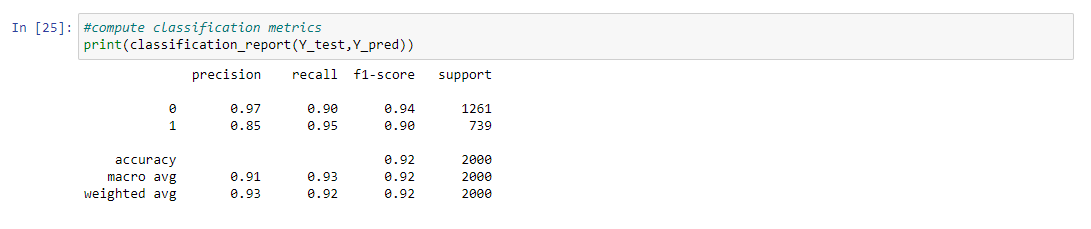
20% of the data was used for a testing set, and the remaining 80% of the data was used for training. For this KNN model, n-nearest neighbor value was set to 7; K = 7. This was done to tell the model to look at the seven nearest values to make the classification. The data was then fit to the KNN model and outcomes were predicted using the above information. Code for splitting the data is provided in the end of the report as well as the full code txt document. A copy of the training and testing sets are provided in the task submission as pdf files.

**2. Analysis Technique**

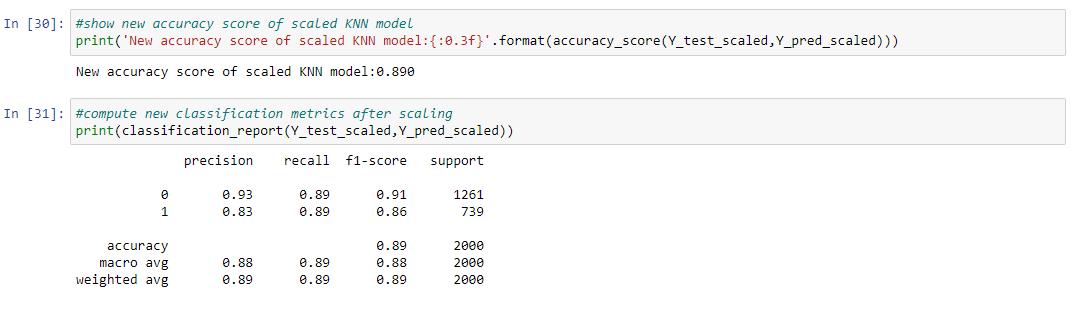
The initial accuracy score of our KNN model is 92.1%.



Shown below are the metrics for our KNN model:



Scaling the data was done to see if the accuracy holds steady, increases, or decreases. The data issplit into train and test again and fit to the KNN model again. From this, a new model accuracy score of 89% was obtained.



Next, a confusion matrix was generated for the scaled data to show the true negatives vs. the false negatives and the false positives vs. the true positives.



Scaling the model ended up decreasing model performance. Accuracy moved from 0.921 to 0.89 while precision moved from 0.97 to 0.93.

**3. Code used to perform the classification analysis**

**Code:**

*#set predictor variables and target variable*

x**=**df**.**drop('ReAdmis',axis**=**1)**.**values

y**=**df['ReAdmis']**.**values

**from** sklearn.metrics **import** accuracy\_score

**from** sklearn.model\_selection **import** cross\_val\_score, train\_test\_split

*#set seed in order to reproduce*

SEED**=**1

*#create training and test datasets*

X\_train,X\_test,Y\_train,Y\_test **=** train\_test\_split(x,y,test\_size**=**0.20,random\_state**=**SEED)

*#export test and training data*

X\_train**.**tofile(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_Xtrain.csv',sep**=**',')

X\_test**.**tofile(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_Xtest.csv',sep**=**',')

Y\_train**.**tofile(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_Ytrain.csv',sep**=**',')

Y\_test**.**tofile(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_Ytest.csv',sep**=**',')

*#begin the KNN model*

knn**=**KNeighborsClassifier(n\_neighbors**=**7)

*#fit data to KNN model*

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

*#predict outcomes from test data*

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

*#show initial accuracy score of KNN model*

print('Initial accuracy score of KNN model: ',accuracy\_score(Y\_test,Y\_pred))

Initial accuracy score of KNN model: 0.921

*#compute classification metrics*

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

*#scale data*

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.pipeline **import** Pipeline

**from** sklearn.metrics **import** accuracy\_score

steps**=**[('scaler',StandardScaler()),('KNN',KNeighborsClassifier())]

pipeline**=**Pipeline(steps)

*#split data*

X\_train\_scaled,X\_test\_scaled,Y\_train\_scaled,Y\_test\_scaled**=**train\_test\_split(x,y,test\_size**=**0.20,random\_state**=**SEED)

*#scale data with pipeline*

KNN\_scaled**=**pipeline**.**fit(X\_train\_scaled,Y\_train\_scaled)

*#predict from scaled data*

Y\_pred\_scaled**=**pipeline**.**predict(X\_test\_scaled)

*#show new accuracy score of scaled KNN model*

print('New accuracy score of scaled KNN model:{:0.3f}'**.**format(accuracy\_score(Y\_test\_scaled,Y\_pred\_scaled)))

New accuracy score of scaled KNN model:0.890

*#compute new classification metrics after scaling*

print(classification\_report(Y\_test\_scaled,Y\_pred\_scaled))

*#import confustin matrix from sklearn*

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

cf\_matrix**=**confusion\_matrix(Y\_test,Y\_pred)

print(cf\_matrix)

*#visualize confustion matrix*

group\_names**=**['True Neg','False Pos','False Neg','True Pos']

group\_counts**=**["{0:0.0f}"**.**format(value) **for** value **in** cf\_matrix**.**flatten()]

group\_percentages**=**["{0:.2%}"**.**format(value) **for** value **in** cf\_matrix**.**flatten()**/**np**.**sum(cf\_matrix)]

labels**=**[f"{v1}\n{v2}\n{v3}" **for** v1,v2,v3 **in**

zip(group\_names,group\_counts,group\_percentages)]

labels**=**np**.**asarray(labels)**.**reshape(2,2)

sns**.**heatmap(cf\_matrix,annot**=**labels,fmt**=**'',cmap**=**'Greens')

<AxesSubplot:>

*#import GridSearch for cross validation*

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

*#set parameters*

param\_grid**=**{'n\_neighbors':np**.**arange(1,50)}

*#recall KNN for cross validation*

KNN**=**KNeighborsClassifier()

*#begin GridSearch cross validation*

knn\_cv**=**GridSearchCV(KNN,param\_grid,cv**=**5)

*#fit model*

knn\_cv**.**fit(X\_train,Y\_train)

*#show best parameters*

print('Best parameters for the KNN model: {}'**.**format(knn\_cv**.**best\_params\_))

Best parameters for the KNN model: {'n\_neighbors': 3}

*#show model best score*

print('Best score for the KNN model: {:.3f}'**.**format(knn\_cv**.**best\_score\_))

Best score for the KNN model: 0.918

**Part V: Data Summary and Implications**

**E. Data Analysis Summary:**

**1. Accuracy and the area under the curve (AUC)**

For this classification model, cross validation helped us find the best parameters for the KNN model by analyzing the training data. In this case, the best parameters ended up being “n-neighbors:3”. The best score for the KNN model was 0.918. The best score for the KNN model was 0.918. The Area Under Curve (AUC) was calculated against the validation data and resulted in a score of 0.9765. The following scores were the result of the 5 cross validation method: 0.94054771; 0.9373297; 0.94170315; 0.92813728; 0.6558256. Based on these results we can conclude that the KNN model we created results in a true positive result 97.65% of the time.

**Code used to find the area of the curve:**

*#import ROC AUC to explain area under curve*

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

*#fit to data*

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

*#determine predicted probabilities*

Y\_pred\_prob**=**knn\_cv**.**predict\_proba(X\_test)[:,1]

*#determine and show AUC score*

print("The Area Under Curve (AUC) on validation data is:{:.4f}"**.**format(roc\_auc\_score(Y\_test,Y\_pred\_prob)))

The Area Under Curve (AUC) on validation data is:0.9765

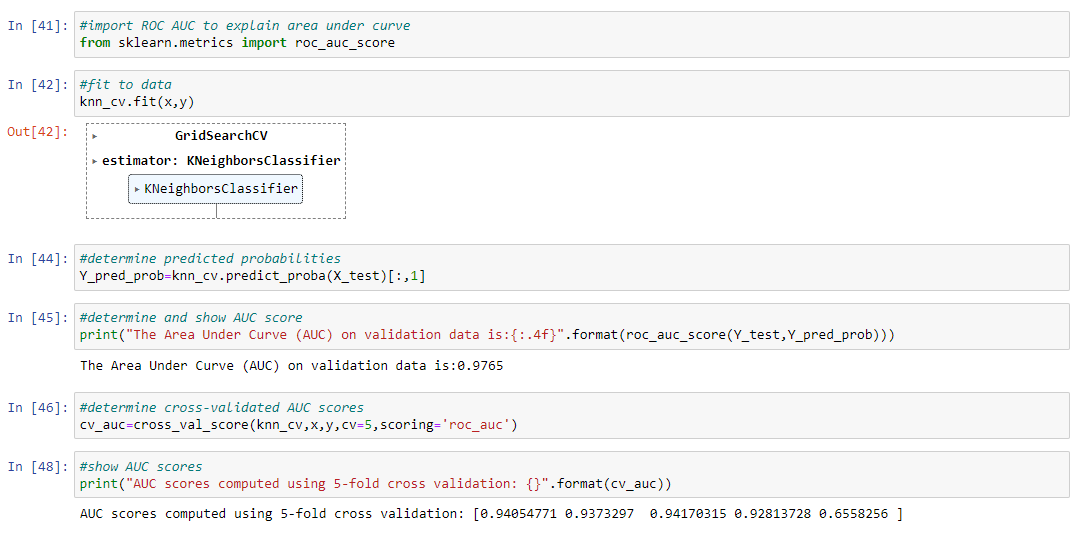
*#determine cross-validated AUC scores*

cv\_auc**=**cross\_val\_score(knn\_cv,x,y,cv**=**5,scoring**=**'roc\_auc')

*#show AUC scores*

print("AUC scores computed using 5-fold cross validation: {}"**.**format(cv\_auc))

AUC scores computed using 5-fold cross validation: [0.94054771 0.9373297 0.94170315 0.92813728 0.6558256 ]



**2. Results and implications of the classification analysis**

Currently, the model has an accuracy score of 0.89, a precision score of 0.93, a recall score of 0.89, an F1-score of 0.91, and an AUC score of 0.9765. The model is able to correctly predict that a patient is not readmitted 93% of the time. It was also able to correctly predict the patients’ status 89% of the time. We can therefore conclude that this model is a strong classifier of data and is able to produce true positive results with a high degree of confidence. Moving forward this model will be a good precursor to making accurate and precise predictions on whether a patient is admitted, using the variables identified.

**3. Limitations of the analysis**

One limitation of using the K-Nearest Neighbor method is that the arbitrary determination of K can affect how accurately the data is classified. While a higher K value may result in a more accurate model, it can also result in over-fitting the model. On the contrary, a K value that is too low can result in underfitting. Both cases can end up leading to a model that does not accurately or precisely classify data. For this model, the K value was set to K = 7. Through model scaling and tuning, it was discovered that K = 3 would have been a better fit. If the value was set to K = 3, then speed and memory usage for running the model would have been reduced.

**4. Recommended course of action based on analysis results**

The recommended course of action going forward is for the hospital to further evaluate the predictor variables so they can better determine which ones are the most impactful in relation to a patient’s chance of readmission. Determining the correct predictor variables with high accuracy and precision would help the hospital staff narrow down their focus and reduce the likelihood of readmitting patients. The accuracy and precision score of the current model are high at 0.89, giving a fair amount of confidence in the model. Nonetheless, additional analysis could prove beneficial to determine if this was an over or under fit of the model, which in turn would help sequester any potential doubt going forward. This could be done by running the analysis again at a lower K value, such as K=3, to possibly provide a more accurate model and reduce time to classify patients and predict readmission.

**Part VI: Demonstration and Supporting Documents**

**Link to the Panopto Video recording:**

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=166704c8-9790-45d6-8629-aebf01136dfc>

**Sources for third party code used to run KNN:**

Saji, Basil. “A Quick Introduction to K – Nearest Neighbor (KNN) Classification Using Python.” *Analytics Vidhya*, 22 Jan. 2021, [www.analyticsvidhya.com/blog/2021/01/a-quick-introduction-to-k-nearest-neighbor-knn-classification-using-python](http://www.analyticsvidhya.com/blog/2021/01/a-quick-introduction-to-k-nearest-neighbor-knn-classification-using-python).

**References**

Harrison, Onel. “Machine Learning Basics with the K-Nearest Neighbors Algorithm.” *Medium*, Towards Data Science, 14 July 2019, [www.towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761](http://www.towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761).

**Full Code Used for this Project**

**import** numpy **as** np

**import** pandas **as** pd

**from** sklearn **import** linear\_model

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**%matplotlib** inline

**import** sklearn

**from** sklearn **import** datasets

**from** sklearn **import** preprocessing

**from** sklearn.neighbors **import** KNeighborsClassifier

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

**from** sklearn.metrics **import** accuracy\_score

**from** sklearn.model\_selection **import** cross\_val\_score, train\_test\_split

**from** sklearn **import** metrics

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

pd**.**set\_option('display.max\_columns', **None**)

df **=** pd**.**read\_csv (r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_clean.csv')

df**.**head()

df**.**info()

*#check for missing data entries*

df**.**isna()**.**any()

*#check for any duplicate data entries in columns*

df[df**.**duplicated()]

*#check if any columns are duplicated - looking for False*

df**.**columns**.**duplicated()**.**any()

*#check if any rows are duplicated - looking for False*

df**.**duplicated()**.**any()

*# drop demographic data*

df **=** df**.**drop(['CaseOrder','Customer\_id','Interaction','UID','City','State','County','Zip','Lat','Lng','Population','Area','TimeZone','Job'], axis**=**1)

*# confirm that columns were dropped*

df**.**head()

*#overview of descriptive statistics*

df**.**describe()

*#rename survey columns for easier identification*

df**.**rename(columns**=**{'Item1':'Timely\_admis','Item2':'Timely\_treat','Item3':'Timely\_visits','Item4':'Reliability','Item5':'Options','Item6':'Hrs\_treat','Item7':'Courteous','Item8':'Active\_listen'},inplace**=True**)

*#verify that the survey columns were renamed correctly*

df**.**head()

*#change yes/no to 1/0*

df **=** df**.**replace(to\_replace **=** ['Yes','No'],value **=** [1,0])

*#verify that the values were changed*

df**.**head()

*# drop non-numeric variables that are less relevant*

df **=** df**.**drop(['Marital', 'Gender','Initial\_admin','Complication\_risk','Services'], axis**=**1)

*#verify that the non-numeric variables were dropped*

df**.**head()

df**.**to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_D209TASK1PREPARED.csv', index**=False**)

*#set predictor variables and target variable*

x**=**df**.**drop('ReAdmis',axis**=**1)**.**values

y**=**df['ReAdmis']**.**values

**from** sklearn.metrics **import** accuracy\_score

**from** sklearn.model\_selection **import** cross\_val\_score, train\_test\_split

*#set seed in order to reproduce*

SEED**=**1

*#create training and test datasets*

X\_train,X\_test,Y\_train,Y\_test **=** train\_test\_split(x,y,test\_size**=**0.20,random\_state**=**SEED)

*#export test and training data*

X\_train**.**tofile(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_Xtrain.csv',sep**=**',')

X\_test**.**tofile(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_Xtest.csv',sep**=**',')

Y\_train**.**tofile(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_Ytrain.csv',sep**=**',')

Y\_test**.**tofile(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_Ytest.csv',sep**=**',')

*#begin the KNN model*

knn**=**KNeighborsClassifier(n\_neighbors**=**7)

*#fit data to KNN model*

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

*#predict outcomes from test data*

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

*#show initial accuracy score of KNN model*

print('Initial accuracy score of KNN model: ',accuracy\_score(Y\_test,Y\_pred))

Initial accuracy score of KNN model: 0.921

*#compute classification metrics*

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

*#scale data*

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.pipeline **import** Pipeline

**from** sklearn.metrics **import** accuracy\_score

steps**=**[('scaler',StandardScaler()),('KNN',KNeighborsClassifier())]

pipeline**=**Pipeline(steps)

*#split data*

X\_train\_scaled,X\_test\_scaled,Y\_train\_scaled,Y\_test\_scaled**=**train\_test\_split(x,y,test\_size**=**0.20,random\_state**=**SEED)

*#scale data with pipeline*

KNN\_scaled**=**pipeline**.**fit(X\_train\_scaled,Y\_train\_scaled)

*#predict from scaled data*

Y\_pred\_scaled**=**pipeline**.**predict(X\_test\_scaled)

*#show new accuracy score of scaled KNN model*

print('New accuracy score of scaled KNN model:{:0.3f}'**.**format(accuracy\_score(Y\_test\_scaled,Y\_pred\_scaled)))

New accuracy score of scaled KNN model:0.890

*#compute new classification metrics after scaling*

print(classification\_report(Y\_test\_scaled,Y\_pred\_scaled))

*#import confustin matrix from sklearn*

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

cf\_matrix**=**confusion\_matrix(Y\_test,Y\_pred)

print(cf\_matrix)

*#visualize confustion matrix*

group\_names**=**['True Neg','False Pos','False Neg','True Pos']

group\_counts**=**["{0:0.0f}"**.**format(value) **for** value **in** cf\_matrix**.**flatten()]

group\_percentages**=**["{0:.2%}"**.**format(value) **for** value **in** cf\_matrix**.**flatten()**/**np**.**sum(cf\_matrix)]

labels**=**[f"{v1}\n{v2}\n{v3}" **for** v1,v2,v3 **in**

zip(group\_names,group\_counts,group\_percentages)]

labels**=**np**.**asarray(labels)**.**reshape(2,2)

sns**.**heatmap(cf\_matrix,annot**=**labels,fmt**=**'',cmap**=**'Greens')

<AxesSubplot:>

*#import GridSearch for cross validation*

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

*#set parameters*

param\_grid**=**{'n\_neighbors':np**.**arange(1,50)}

*#recall KNN for cross validation*

KNN**=**KNeighborsClassifier()

*#begin GridSearch cross validation*

knn\_cv**=**GridSearchCV(KNN,param\_grid,cv**=**5)

*#fit model*

knn\_cv**.**fit(X\_train,Y\_train)

*#show best parameters*

print('Best parameters for the KNN model: {}'**.**format(knn\_cv**.**best\_params\_))

Best parameters for the KNN model: {'n\_neighbors': 3}

*#show model best score*

print('Best score for the KNN model: {:.3f}'**.**format(knn\_cv**.**best\_score\_))

Best score for the KNN model: 0.918

*#import ROC AUC to explain area under curve*

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

*#fit to data*

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

*#determine predicted probabilities*

Y\_pred\_prob**=**knn\_cv**.**predict\_proba(X\_test)[:,1]

*#determine and show AUC score*

print("The Area Under Curve (AUC) on validation data is:{:.4f}"**.**format(roc\_auc\_score(Y\_test,Y\_pred\_prob)))

The Area Under Curve (AUC) on validation data is:0.9765

*#determine cross-validated AUC scores*

cv\_auc**=**cross\_val\_score(knn\_cv,x,y,cv**=**5,scoring**=**'roc\_auc')

*#show AUC scores*

print("AUC scores computed using 5-fold cross validation: {}"**.**format(cv\_auc))

AUC scores computed using 5-fold cross validation: [0.94054771 0.9373297 0.94170315 0.92813728 0.6558256 ]