Shawn Wheeler

000364855

D209: Data Mining I

Task I

Table of Contents

[A1: Proposal of Question 3](#_Toc166681779)

[A2: Defined Goal 3](#_Toc166681780)

[B1: Explanation of Classification Method 3](#_Toc166681781)

[B2: Summary of Method Assumption 3](#_Toc166681782)

[B3: Packages or Libraries Set 3](#_Toc166681783)

[C1: Data Preprocessing 4](#_Toc166681784)

[C2: Data Set Variables 4](#_Toc166681785)

[C3: Steps for Analysis 5](#_Toc166681786)

[C4: Cleaned Data Set 8](#_Toc166681787)

[D1: Splitting the Data 8](#_Toc166681788)

[D2: Output and Intermediate Calculations 9](#_Toc166681789)

[D3: Code Execution 12](#_Toc166681790)

[E1: Accuracy and AUC 14](#_Toc166681791)

[E2: Results and Implications 14](#_Toc166681792)

[E3: Limitations 15](#_Toc166681793)

[E4: Course of Action 15](#_Toc166681794)

[F: Panopto Recording 15](#_Toc166681795)

[G: Sources for Third-Party Code 15](#_Toc166681796)

[H: Sources 16](#_Toc166681797)

## **Part I: Research Question**

## A1: Proposal of Question

What are the major predictor variables in determining or predicting the readmission rates (ReAdmis) of patients hospitalized from the medical dataset? I will be using k-Nearest Neighbor (KNN) as my method of attempting to answer the aforementioned question.

## A2: Defined Goal

The goal of this analysis is to gain further insight into which features, or variables, are useful in predicting ReAdmis and to what extent. Understanding the reasons that primarily affect ReAdmis would allow for a great deal of useful action for business planning or cost mitigation purposes.

**Part II: Method Justification**

## B1: Explanation of Classification Method

The classification method that will be used in this project is k-Nearest Neighbor, or KNN. According to IBM, “The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.” (IBM, 2023). To explain further, KNN makes predictions based on similarity and a distance metric. First, the value for k must be determined which will be achieved by using SelectKBest. Distance, oftentimes Euclidean distance, is then measured and considered. Using this data, the nearest neighbors are determined. Afterwards, classification of a data point is “determined by a majority vote or average of the K neighbors.” (GeeksforGeeks, 2018). This allows for the classification of data needed to address the research question as to how features may predict ReAdmis.

## B2: Summary of Method Assumption

An important assumption of the KNN algorithm is that distance and similarity are directly correlated. In other words, the closer data points are, the more similar they are, and the further away from one another they are, the less similar they are. If this is not accurate to a data set, the algorithm is much less useful.

## B3: Packages or Libraries Set

*Table of Libraries and Packages:*

|  |  |
| --- | --- |
| **Libraries / Packages** | **Usage** |
| pandas | Data manipulation |
| numpy | Mathematics operations |
| seaborn | Creating visualizations |
| matplotlib.pyplot | Creating visualizations |
| stats.models.stats.outliers\_influence | Calculating variance inflation factor (VIFs) |
| sklearn.feature\_selection | SelectKBest to determine appropriate features for model |
| sklearn.model\_selection |  |
| import train\_test\_split | Splitting data into training and testing data sets |
| import GridSearchCV | Determining best variables to use for k-NN model |
| sklearn.neighbors | KNeighborsClassifier to instantiate k-NN computations |
| sklearn.preprocessing | Assisting with calculations by preprocessing data |
| sklearn.metrics |  |
| import classification\_report | Creating report regarding model performance |
| import confusion\_matrix | Calculating confusion matrix |
| import accuracy\_score | Calculating accuracy of models |
| import roc\_auc\_score | Calculating area under curve (AUC) score |
| import roc\_curve | Visualization of receiver operating characteristic (ROC) curve of model |

**Part III: Data Preparation**

## C1: Data Preprocessing

In order to align our data with what we will need for performing k-NN analysis, I will need to address missing data, outliers, data type mismatches, create dummy variables, and scale the data to be used. Dummy variables are created in order to remove possible collinearity and to ensure categorical data can be utilized for k-NN. Data must also be scaled prior to performing k-NN as k-NN relies on similarity and distance which scaling corrects for variables of different magnitudes of values.

## C2: Data Set Variables

*Variables Chosen to be Tested with SelectKBest:*

|  |  |
| --- | --- |
| **Independent Variables** | **Data Type** |
| Area | Categorical |
| Children | Continuous (Numerical) |
| Age | Continuous (Numerical) |
| Income | Continuous (Numerical) |
| Marital | Categorical |
| Gender | Categorical |
| VitD\_levels | Continuous (Numerical) |
| Doc\_visits | Continuous (Numerical) |
| Full\_meals\_eaten | Continuous (Numerical) |
| vitD\_supp | Continuous (Numerical) |
| Soft\_drink | Categorical |
| Initial\_admin | Categorical |
| HighBlood | Categorical |
| Stroke | Categorical |
| Complication\_risk | Categorical |
| Overweight | Categorical |
| Arthritis | Categorical |
| Diabetes | Categorical |
| Hyperlipidemia | Categorical |
| BackPain | Categorical |
| Anxiety | Categorical |
| Allergic\_rhinitis | Categorical |
| Reflux\_esophagitis | Categorical |
| Asthma | Categorical |
| Services | Categorical |
| Initial\_days | Continuous (Numerical) |

## C3: Steps for Analysis

1. **Load data for exploration and manipulation**

df = pd.read\_csv(“C:/Users/Owner/medical\_clean.csv”)

1. **Data exploration**

df.shape # Check dimensionality of data set

df.head() # View all columns

df.info() # View details of columns including datatypes

1. **Histograms of all quantitative columns**

df.hist(grid=False, figsize(15,10), layout=[5,5])

1. **Checking for duplicates, missing data, outliers, and other issues**

df.duplicated().value\_counts() # Check for duplicates

df.isnull().sum() # Check for missing data

df.describe() # Provide details of quantitative variables

df.value\_counts() # Provide details of categorical variables

# Checking for outliers below

quant\_columns = ['Lat', 'Lng', 'Population', 'Children', 'Age', 'Income', 'VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten', 'VitD\_supp', 'Initial\_days', 'TotalCharge', 'Additional\_charges']

for column in df:

if column in quant\_columns:

plt.figure()

plt.gca().set\_title(column)

df.boxplot([column])

1. **Deal with outliers and rounding values**

# Round Initial\_days

df['Initial\_days'] = df.Initial\_days.round(1)

# Remove outliers (> {Mean + 2\*std})

# Children outliers

df['Children'] = np.where(df['Children'] > 5, np.nan, df['Children'])

df['Children'].fillna(df['Children'].median(), inplace=True)

# Setting Income outliers to NAs

df['Income'] = np.where(df['Income'] > 97532.79, np.nan, df['Income'])

# Setting Income NaN outliers to Median value

df['Income'].fillna(df['Income'].median(), inplace=True)

# Setting values to 2 decimal places to better represent income

df['Income'] = df.Income.round(2)

# VitD\_levels outliers

df['VitD\_levels'] = np.where(df['VitD\_levels'] > 21.99, np.nan, df['VitD\_levels'])

df['VitD\_levels'].fillna(df['VitD\_levels'].median(), inplace=True)

# Round VitD\_levels

df['VitD\_levels'] = df.VitD\_levels.round(2)

# Full\_meals\_eaten outliers

df['Full\_meals\_eaten'] = np.where(df['Full\_meals\_eaten'] > 3, np.nan, df['Full\_meals\_eaten'])

df['Full\_meals\_eaten'].fillna(df['Full\_meals\_eaten'].median(), inplace=True)

1. **Data type conversions**

df['Full\_meals\_eaten'] = df['Full\_meals\_eaten'].astype(int)

df['Children'] = df['Children'].astype(int)

1. **Re-expression of Booleans as numeric values**

df['ReAdmis'] = df['ReAdmis'].map({'Yes': '1', 'No': '0'})

df['Soft\_drink'] = df['Soft\_drink'].map({'Yes': '1', 'No': '0'})

df['HighBlood'] = df['HighBlood'].map({'Yes': '1', 'No': '0'})

df['Stroke'] = df['Stroke'].map({'Yes': '1', 'No': '0'})

df['Overweight'] = df['Overweight'].map({'Yes': '1', 'No': '0'})

df['Arthritis'] = df['Arthritis'].map({'Yes': '1', 'No': '0'})

df['Diabetes'] = df['Diabetes'].map({'Yes': '1', 'No': '0'})

df['Hyperlipidemia'] = df['Hyperlipidemia'].map({'Yes': '1', 'No': '0'})

df['BackPain'] = df['BackPain'].map({'Yes': '1', 'No': '0'})

df['Anxiety'] = df['Anxiety'].map({'Yes': '1', 'No': '0'})

df['Allergic\_rhinitis'] = df['Allergic\_rhinitis'].map({'Yes': '1', 'No': '0'})

df['Reflux\_esophagitis'] = df['Reflux\_esophagitis'].map({'Yes': '1', 'No': '0'})

df['Asthma'] = df['Asthma'].map({'Yes': '1', 'No': '0'})

1. **Convert categorical variables into dummy variables and map dummies to numerical values**

# Area dummies

area\_dummies = pd.get\_dummies(df.Area, columns=['Rural', 'Urban'])

df.insert(1, 'Area\_rural', area\_dummies['Rural'])

df.insert(2, 'Area\_urban', area\_dummies['Urban'])

# Marital dummies

marital\_dummies = pd.get\_dummies(df.Marital, columns=['Divorced', 'Married', 'Widowed'])

df.insert(4, 'Marital\_divorced', marital\_dummies['Divorced'])

df.insert(5, 'Marital\_married', marital\_dummies['Married'])

df.insert(6, 'Marital\_never\_married', marital\_dummies['Never Married'])

# Gender dummies

gender\_dummies = pd.get\_dummies(df.Gender, columns=['Male', 'Female'])

df.insert(7, 'Gender\_male', gender\_dummies['Male'])

df.insert(8, 'Gender\_female', gender\_dummies['Female'])

# Initial\_admin dummies

initial\_admin\_dummies = pd.get\_dummies(df.Initial\_admin, columns=['Elective Admission', 'Emergency Admission'])

df.insert(25, 'Initial\_admin\_elective\_admission', initial\_admin\_dummies['Elective Admission'])

df.insert(25, 'Initial\_admin\_emergency\_admission', initial\_admin\_dummies['Emergency Admission'])

# Complication\_risk dummies

complication\_risk\_dummies = pd.get\_dummies(df.Complication\_risk, columns=['High', 'Low'])

df.insert(28, 'Complication\_risk\_high', complication\_risk\_dummies['High'])

df.insert(29, 'Complication\_risk\_low', complication\_risk\_dummies['Low'])

# Services dummies

services\_dummies = pd.get\_dummies(df.Services, columns=['Blood Work', 'Intravenous', 'CT Scan'])

df.insert(41, 'Services\_blood\_work', services\_dummies['Blood Work'])

df.insert(41, 'Services\_intravenous', services\_dummies['Intravenous'])

df.insert(41, 'Services\_ct\_scan', services\_dummies['CT Scan'])

# Map dummies to numerical values

df['Area\_rural'] = df['Area\_rural'].astype(int)

df['Area\_urban'] = df['Area\_urban'].astype(int)

df['Marital\_divorced'] = df['Marital\_divorced'].astype(int)

df['Marital\_married'] = df['Marital\_married'].astype(int)

df['Marital\_never\_married'] = df['Marital\_never\_married'].astype(int)

df['Gender\_male'] = df['Gender\_male'].astype(int)

df['Gender\_female'] = df['Gender\_female'].astype(int)

df['Initial\_admin\_elective\_admission'] = df['Initial\_admin\_elective\_admission'].astype(int)

df['Initial\_admin\_emergency\_admission'].astype(int)

df['Complication\_risk\_high'] = df['Complication\_risk\_high'].astype(int)

df['Complication\_risk\_low'] = df['Complication\_risk\_low'].astype(int)

df['Services\_blood\_work'] = df['Services\_blood\_work'].astype(int)

df['Services\_intravenous'] = df['Services\_intravenous'].astype(int)

df['Services\_ct\_scan'] = df['Services\_ct\_scan'].astype(int)

1. **Drop columns not to be used for SelectKBest**

df = df.drop(['CaseOrder', 'Customer\_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Marital', 'Gender', 'Initial\_admin', 'Complication\_risk', 'TotalCharge', 'Services', 'Additional\_charges', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'], axis=1)

1. **Select variables (SelectKBest)**

# SelectKBest

skbest = SelectKBest(k='all')

X\_new = skbest.fit\_transform(X,y)

# Determining p-values / listing < .05

p\_values = pd.DataFrame({'Feature': X.columns, 'p\_value': skbest.pvalues\_}).sort\_values('p\_value')

p\_values[p\_values['p\_value'] < 0.05]

# Variables to keep (<.05 p-value)

features\_to\_keep = p\_values['Feature'][p\_values['p\_value'] < 0.05]

print(features\_to\_keep)

## C4: Cleaned Data Set

*See Attached .csv:* cleand209data.csv

**Part IV: Analysis**

## D1: Splitting the Data

The code below was used to split the data set for our model into training data and testing data. There are attached .csvs for each of the four outputs below, as well. Data was set at an 80/20 split and the random\_state was chosen randomly.

*Split Data Code:*

X\_train, X\_test, y\_train, y\_test= train\_test\_split(X, y, train\_size=.8, test\_size=.2, random\_state=8)

*Attached .csvs:* X\_train.csv, X\_test.csv, Y\_train.csv, Y\_test.csv

pd.DataFrame(X\_train).to\_csv('X\_train.csv')

pd.DataFrame(X\_test).to\_csv('X\_test.csv')

pd.DataFrame(y\_train).to\_csv('y\_train.csv')

pd.DataFrame(y\_test).to\_csv('y\_test.csv')

## D2: Output and Intermediate Calculations

For this section, I’ll include screenshots of the outputs below each topic.

To create more useful data, SelectKBest was used to find the highest performing features of the scaled data set. All variables with a p-value <0.05 were then selected for analysis via KNN.

*SelectKBest:*

*A screenshot of a computer

Description automatically generated*

After splitting the data, the next step was to determine the optimum value for k based on the training data set that was created. After the classifier was instantiated, GridSearchCV was then utilized for hyperparameter optimization. In this case, it uses param\_grid and searches over all combinations of the data. These are then cross-validated multiple times, five in this case, as a way of determining model performance (Nashaat, 2023). The optimum value for k was found to be 7 in this case.

*KNN to find k and GridSearchCV:*

**

After determining the optimum value for k(k=7), other values were calculated to determine the performance of this model. The mean score refers to the average performance during GridSearchCV for the model. The mean score of .9835 indicates very high performance as it is on a scale of 0-1 with 1 being the best. Accuracy is shown to be 0.9715 which means that 97.15% of the data in the data set was classified correctly. The AUC was then calculated as 0.99. For AUC, a value of 1 is a perfect model with lower values indicating poorer predictive performance.

*Mean Score, Accuracy, and AUC:*

**

**

**

Next, a classification report was created. Looking at the results, it showed that the model performs well in predicting both the negative and positive classes. Precision and recall for both are above 0.95 with an accuracy of 0.97

*Classification Report:*

A screenshot of a graph

Description automatically generated

Continuing the analysis, a confusion matrix was created and then visualized as a heatmap to examine how the predicted data performed with regards to true and false positives and negatives. This is another way of examining accuracy, for instance. In this example, there were only 57 inaccurate predictions out of the 2,000 tested for a rate of 2.85% thus showing high performance.

*Confusion Matrix and Heatmap:*

*A close up of numbers

Description automatically generated*

*A graph of heat map

Description automatically generated with medium confidence*

Following that, a model complexity curve was generated. A model complexity curve shows to determine accuracy of the testing and training data as complexity of the model increases. Accuracy for the testing data was shown to be 98.15% and 97.2% for the training data.

*Model Complexity Curve:*

*A graph of a number of neighbors

Description automatically generated*

Finally, a ROC curve was produced to visualize accuracy through the lens of false positive and true positive rates of the model. The curve itself shows the AUC along with the 45 degree line for the classifier. Since the AUC was shown to be 0.99, the graph is close to ideal showing an extremely high rate of true positives.

*ROC-AUC Curve:*

*A graph with a line

Description automatically generated*

## D3: Code Execution

*See attached code:* d209task1complete.ipynb

*See code below:*

# Load clean data set

df = pd.read\_csv("C:/Users/Owner/cleand209data.csv")

X = df[['Marital\_divorced', 'Services\_ct\_scan', 'Initial\_days', 'Services\_intravenous', 'Initial\_admin\_emergency\_admission']]

y = df['ReAdmis']

X\_train, X\_test, y\_train, y\_test= train\_test\_split(X, y, train\_size=.8, test\_size=.2, random\_state=8)

# Output to .csvs

pd.DataFrame(X\_train).to\_csv('X\_train.csv')

pd.DataFrame(X\_test).to\_csv('X\_test.csv')

pd.DataFrame(y\_train).to\_csv('y\_train.csv')

pd.DataFrame(y\_test).to\_csv('y\_test.csv')

D2: Perform K-NN and analysis

# Find optimum number of neighbors / cross-validation

param\_grid = {'n\_neighbors': np.arange(1, 25)}

knn = KNeighborsClassifier()

knn\_two = GridSearchCV(knn, param\_grid)

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

knn\_two.best\_params\_

# Mean score

print("Mean Score:", knn\_two.best\_score\_)

# AUC score for kNN model

y\_predicted\_probability = knn.predict\_proba(X\_test)[:,1]

print("AUC:", roc\_auc\_score(y\_test, y\_predicted\_probability))

# Accuracy of kNN model

knn = KNeighborsClassifier(n\_neighbors=7)

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

print("Accuracy:", knn.score(X\_test, y\_test))

# Classification Report

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

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

# Confusion Matrix

print("Confusion Matrix\n", confusion\_matrix(y\_test, y\_predicted))

# Heatmap for Confusion Matrix

seaborn.heatmap(confusion\_matrix(y\_test, y\_predicted), annot=True, fmt='G', cmap='coolwarm' )

plt.xlabel('Predicted')

plt.ylabel('Test')

plt.title('Heatmap for Confusion Matrix')

# (Course Material, n.d.)

# Model Complexity Curve

neighbors = np.arange(1, 25)

train\_accuracy = np.empty(len(neighbors))

test\_accuracy = np.empty(len(neighbors))

# Loop k, classifier fit with training data, find accuracy of both training and test data

for i, k in enumerate(neighbors):

knn = KNeighborsClassifier(n\_neighbors=k)

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

train\_accuracy[i] = knn.score(X\_train, y\_train)

test\_accuracy[i] = knn.score(X\_test, y\_test)

print("Training Accuracy:", train\_accuracy[i], "\nTest Accuracy:", test\_accuracy[i])

# Visualization

plt.title('k-NN for Neighbors < 25')

plt.plot(neighbors, test\_accuracy, label = 'Testing Accuracy')

plt.plot(neighbors, train\_accuracy, label='Training Accuracy')

plt.legend()

plt.xlabel('Number of Neighbors')

plt.ylabel('Accuracy')

plt.show()

# (*How to Plot ROC Curve in Python*, 2024)

auc\_roc = roc\_auc\_score(y\_test, y\_predicted\_probability)

print("AUC-ROC:", auc\_roc)

# Roc Curve

fpr, tpr, thresholds = roc\_curve(y\_test, y\_predicted\_probability)

plt.plot(fpr, tpr)

plt.plot([0,1], [0,1], 'k--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('true Positive Rate')

plt.title('ROC Curve')

plt.show()

**Part V: Data Summary and Implications**

## E1: Accuracy and AUC

Accuracy in this case refers to the correct classifications divided by the total attempted classifications. Therefore, the accuracy score obtained of 0.9715 means that 97.15% of classifications were accurate leaving an error rate of only 2.85%.

AUC, area under the curve, is another way of testing the performance of a model. It has a scale of 0 to 1 and measures “the probability that the model ranks a random positive example more highly than a random negative example.” (Google, 2019). For the values themselves, a 1 represents a perfect model and 0.5 is essentially random as it is essentially a percentage of area. The AUC value calculated for this model was 0.99 which is nearly perfect. AUC is represented as the area under the curve shown on the ROC graph.

## E2: Results and Implications

KNN was chosen as the classification method for this project. In order to perform that, SelectKBest was used to select the most significant features to use for the KNN Model. Using SelectKBest (and after testing for possible multicollinearity issues via VIFs), 5 variables were chosen: Marital\_divorced, Services\_ct\_scan, Initial\_days, Services\_intravenous, Initial\_admin\_emergency\_admission. KNN was then performed to determine that k=7 was optimal for our dataset. This information was used to glean further information.

Accuracy of the KNN model was 97.15%, AUC was 0.99, and the mean score was 98.35%. These values are all very high and correlate with what is shown in the visualizations for the confusion matrix, the model complexity curve, and the ROC curve.

All of these points show that the model is very accurate in its predictions and should be able to provide insight into the research question and consideration for greater research.

## E3: Limitations

One limitation encountered during this data analysis was that the data for ReAdmis was 6331 “Yes” and 3669 “No” values which is a fairly large imbalance. When data has a large deviation from equal distribution, it can lead to issues with the usefulness of the data acquired by the analysis. Ideally it would be 50/50.

## E4: Course of Action

Since the model has such high values for accuracy and AUC, it should be considered for more in-depth research. As there could be issues with the validity of the data, however, due to data imbalance and a small data set, all that can really be said is that in this data set, the prediction rate of readmission based on the five selected features was nearly perfect. I would recommend a business investigating the features that were selected for further analysis especially as there is some overlap with two services of which only one could be selected per sample. If there is correlation, which isn’t always the case, it would be greatly beneficial from a hospital’s perspective. Being able to lessen causes of readmission or predict them allows for better resource allocation and cost savings. I would also recommend gathering more data, both in quantity of samples and types of data harvested (e.g. different categories for Services) as shown to be relevant through further research.

**Part VI: Demonstration**

## F: Panopto Recording

*See Panopto Link:* https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=818e804e-d6f7-4c33-bcaf-b174014f139e

## G: Sources for Third-Party Code

Course Materials

*How to plot ROC curve in Python*. (2024, March 8). GeeksforGeeks. https://www.geeksforgeeks.org/how-to-plot-roc-curve-in-python/

## H: Sources

GeeksforGeeks. (2018, November 13). *K-Nearest Neighbours - GeeksforGeeks*. GeeksforGeeks. https://www.geeksforgeeks.org/k-nearest-neighbours/

Google. (2019). *Classification: ROC Curve and AUC  |  Machine Learning Crash Course*. Google Developers. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

IBM. (2023). *What is the k-nearest neighbors algorithm? | IBM*. Www.ibm.com. https://www.ibm.com/topics/knn

Nashaat, M. (2023, October 22). *Hyperparameter Tuning with GridSearchCV*. Medium. https://medium.com/@mohammednashaat29/hyperparameter-tuning-with-gridsearchcv-8724f215a383