Performance Assessment D209 – Data Mining I  
Task I

Doug Haunsperger

College of IT, Western Governors University

December 5, 2023

# Part I. Rese**arch Questi**on

## A1. Question Proposal

I propose to research the question, “Can patient readmission to the hospital be predicted by demographic or medical data using a k-nearest neighbor (KNN) classification model?”

## A2. Goals

According to the scenario given in the provided data dictionary, hospitals have a need to reduce readmissions to avoid CMS fines and would find this data analysis highly valuable.

The answer to this question could lead to improvements in patient treatment. For example, if it is shown that there are controllable factors that contribute to readmission, the hospital can try to ameliorate those factors while still in the hospital. Alternatively, if it is shown that there are non-controllable factors (i.e. income, gender, etc.) the hospital can focus resources on those patients and double-check their health status prior to release to attempt to reduce the chance of readmission.

# Part II. Method Justification

## B1. Explanation of Classification Method

A KNN classification model is built using a training set of data. The independent variables, or features, of this data are used to build an n-dimensional feature space. The rows of training data are located within this feature space along with their target variable’s class. To predict the classification of new data, the model algorithm uses “a voting system, where the majority class label determines the class label of a new data point among its nearest ‘k’ (where k is an integer) neighbors in the feature space.” (Shafl, 2023).

After training and hyperparameter tuning, if successful, I expect to have a model that outputs a correct classification for ‘ReAdmis’ a high percentage of the time on the test data. Then when new patient data is entered, the hospital can assume that an output of ‘ReAdmis =1’ can be interpreted as a patient with a high likelihood of hospital readmission.

## B2. Method Assumptions

“The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.” (Harrison, 2018). In this case, “near” means a minimal distance between points in the feature space using a particular distance metric (Manhattan, Euclidean, Minkowski, etc.).

## B3. Package / Library List

Figure 1  
*Package and library import statements*A screen shot of a computer program

Description automatically generated

Figure 1 shows my import code. Each package or library imported is useful in the full scope of the project. Pandas is used to work with DataFrames. Numpy is used for various mathematical functions (finding means, covariance matrices, eigenvalues, etc.). Pyplot and seaborn are used for graphical display.

The remaining lines import certain functions or classes from the scikit-learn package. From the API reference (2023), we see that KNeighborsClassifier is used to perform the KNN analysis and classification. Train\_test\_split automates the splitting of a dataset into training and testing data. Cross\_validate is a function used to perform cross validation, that is, ensuring that when we score our model while tuning the hyperparameter *k*, we’re not overfitting on training data. Accuracy\_score, precision\_score, recall\_score, f1\_score, and roc\_auc\_score are various functions that calculate scoring metrics for the model: namely, accuracy, precision, recall, F1 score, and area under the ROC curve. Roc\_curve is a function to compute the receive operating characteristic (ROC) curve. Confusion\_matrix computes a “confusion matrix to evaluate the accuracy of a classification” (scikit-learn, 2023). Classification\_report creates a test report with important classification metrics from the model. I used StandardScaler to scale the quantitative variables to unit variance with zero mean. I used the PCA class for principal component analysis to reduce the dimensionality of the dataset because the performance of KNN “gets worse as the number of features increases” (Tokuç, 2022). Finally, the ColumnTransformer function allowed me to easily scale & transform the quantitative variables while leaving the categorical columns untouched.

# Part III. Data Preparation

## C1. Data Preprocessing

As mentioned in the previous section, I performed PCA to reduce the dimensionality of the dataset in order to have the best chance of a well-performing KNN model. After examining the eigenvalues and explained variance given by each principal component, I chose to retain 7 PCs, which together explained about 62.5% of the variance in the original 26-column dataset.

## C2. Data Set Variables

Figure 2 shows a snippet of my code where I have chosen my explanatory variables and split them into quantitative and categorical groups.

Figure 2  
*Variable list*A close-up of a computer screen

Description automatically generated

## C3. Preparation Steps for Analysis

See attached Jupyter notebook, ‘KNNClass.ipynb’ under the heading ‘Data Preparation’. I have provided headers to each code block indicating the preparation steps.

## C4. Cleaned Data Set

The cleaned data set is attached as ‘clean\_medical\_data.csv’.

# Part IV. Analysis

## D1. Split Data

The split test and training data sets used for the KNN analysis are attached as ‘test\_xform\_medical\_data.csv’ and ‘train\_xform\_medical\_data.csv’. Since these datasets have undergone PCA transformation, I have also attached a CSV with the loadings for each explanatory variable per PC, as ‘PC\_loadings.csv’.

## D2. Analysis Technique

See attached Jupyter notebook, under the heading ‘Data Analysis’ for code and output. I largely followed the methodology given by Shafl, 2023.

To summarize, these are the steps I performed:

1. Instantiate a KNeighborsClassifier object with k=3 neighbors as an initial value
2. Fit to the scaled, PCA-transformed training set X\_train\_pca and calculate initial accuracy (54%).
3. Tune the hyperparameter *k,* looking at accuracy, F1 score (combining precision and recall), and area under the ROC curve (AUC). I only checked odd values of *k* since Harrison states that in classification problems, it is usual to “make [*k*] an odd number to have a tiebreaker”. (2018).
   1. Looking at each potential odd value of *k* less than 50, I performed a 5-fold cross validation and calculated the mean accuracy, F1, and AUC scores for each of the 5 folds.
   2. I plotted the results on a graph, showing that accuracy generally increases as *k* increases, but F1 plummets. I chose to base my selection on AUC.
   3. AUC was maximized at *k =* 3.
4. Plot the ROC curve. “[T]o compute the ROC we do not merely want the predictions on the test set, but we want the probability” output by the model. “To do this we apply the method predict\_proba to the model and pass it the test data.” (Bowne-Anderson, n.d.)
5. Calculate and plot a confusion matrix for the model showing true and false positives, and true and false negatives.

## D3. Code

See attached Jupyter notebook.

# Part V. Data Summary & Implications

## E1. Accuracy & AUC

Accuracy is defined as the number of correct predictions divided by the total number of predictions made (Zvornicanin, 2023). The accuracy of my model was 54.4%. A somewhat higher accuracy (>60%) could have been achieved with a higher *k* (*k* > 20), but at the cost of much worse precision and recall, as shown by the F1 score line in Figure 3.

Figure 3  
*Accuracy vs. F1 score vs. AUC for varying values of k*

A graph of different colored lines

Description automatically generated

The Receiver Operating Characteristics “curve shows the relationship between false-positive rate and true positive rate for different probability thresholds of model predictions.” (Zvornicanin, 2023). AUC is the area under the ROC curve. The AUC of my model was 0.493. Unfortunately, this is worse than a no-skill, random assignment model (the dashed line in Figure 4), which has an AUC of 0.5.

Figure 4  
*Model ROC Curve*

A graph with a line

Description automatically generated

## E2. Results & Implications

An accuracy of just over 50%, and an AUC just under 0.5 means that this model is only approximately as good as random chance in determining whether a patient will be readmitted based on the selected input variables. It would not be helpful to the hospital to utilize this model as-is. I showed in section E1 that accuracy could be improved to about 63% by choosing a high k-value (*k =* 49), at the cost of very poor precision and recall (See Figure 5). AUC is little different than a random-chance model regardless of the value of k with this set of variables. Additional feature selection to attempt to find a better set of independent variables to analyze with KNN would be needed to find a model with a better AUC score.

Figure 5  
*Accuracy of high-k KNN model*

*A screenshot of a computer code

Description automatically generated*

## E3. Limitations

It is likely that this data set violates the KNN assumption of ‘like is near like’. If patients with similar demographic and medical history profiles do not have similar odds of readmission, then this KNN method will be ineffective. There may be other variables than those in the data set that would be more receptive to this analysis.

## E4. Course of Action

I would recommend that the hospital not use this model to make patient decisions. Further work, as discussed in section E2, is needed to produce a more accurate classifier model.

# Part VI. Demonstration & Supporting Documentation

## F. Demonstration Video

## A video describing my methods and code can be found at: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=17a2418b-a784-44e2-9111-b0d500e39140>

## G. Third-party Code Sources

Chuang, L. (2020, March 7). *Principal Component Analysis (PCA) using python (Scikit-learn).* Medium. <https://medium.com/luca-chuangs-bapm-notes/principal-component-analysis-pca-using-python-scikit-learn-48c4c13e49af>

Müller, A. (n.d.). Lesson 3: Fine-tuning your model: Plotting an ROC curve. *Machine Learning with scikit-learn* [MOOC]*.* Datacamp. <https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/fine-tuning-your-model?ex=5>

“Piman” [StackOverflow username] (2017). Answer to: *Evaluate multiple scores on sklearn cross\_val\_score*. StackOverflow. <https://stackoverflow.com/a/35886445>

## H. References

Bowne-Anderson, H. (n.d.). Lesson 3: Fine-tuning your model: Logistic Regression and the ROC curve. *Machine Learning with scikit-learn* [MOOC]*.* Datacamp. <https://campus.datacamp.com/courses/machine-learning-with-scikit-learn/fine-tuning-your-model?ex=3>

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

Shafl, A. (2023, February). *K-Nearest Neighbors (KNN) Classification with scikit-learn.* DataCamp. <https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn>

Middleton, K. (n.d.). *Getting Started with D206 | Principal Component Analysis.* Western Governors University. <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=3bcc452f-fa35-43be-b69f-b05901356f95>

Middleton, K. (November 2022). *Getting Started with D208, Part I.* Western Governors University. <https://westerngovernorsuniversity.sharepoint.com/:b:/r/sites/DataScienceTeam/Shared%20Documents/Graduate%20Team/D208/Student%20Facing%20Resources/Dr.%20Middleton%20Getting%20Started%20with%20D208(Part%20I)COIT.pdf?csf=1&web=1&e=CLRFMI>

*scikit-learn 1.3.2 API reference documentation* (2023). <https://scikit-learn.org/stable/modules/classes.html#>

Tokuç, A. (November 9, 2022). *k-Nearest Neighbors and High Dimensional Data*. Baeldung. <https://www.baeldung.com/cs/k-nearest-neighbors>

Zvornicanin, E. (May 31, 2023). *Accuracy vs AUC in Machine Learning.* Baeldung. <https://www.baeldung.com/cs/ml-accuracy-vs-auc>