**D209 Task 1: Classification Analysis**

**Joseph Duszynski**

**ID #011237236**

**A.1. Proposal of question**

Using the medical dataset (WGU, 2024 [1]), this project seeks to answer the question “Can we predict if a patient will need to be readmitted within a month of release (‘ReAdmis’ variable) from existing patient data?”. Specifically, can we use a KNN classification model to achieve accurate predictions for patients’ readmission status?

**A.2. Defined goal**

This project aims to find the most relevant variables affecting patients’ readmission status in a KNN model. This can be achieved through scikit-learn’s “SelectKBest()” with an ANOVA F-test, which will then be used for constructing the KNN model.

These insights can assist hospital administrators and analysts understand what factors drive patients needing to be readmitted shortly after leaving the hospital to make the necessary changes for greater efficiency and better patient health outcomes. That is, current and future patients who fit known profiles of past patients who needed to be readmitted can be identified as at risk of readmission.

**B.1. Explanation of classification method**

KNN aims to use independent variables as input to predict a label of a categorical target variable. The independent variables are numeric, but categorical variables can be one-hot or ordinal encoded. Training data with known target labels is used to construct the model. For a given observation, its *k* nearest neighbors as determined by a specified metric (Euclidean in this case) are located from the training data. The target variable class with a plurality among the *k* nearest neighbors becomes the predicted class (label) for the desired observation. The resulting model takes an input observation and produces a predicted label for the value of its target variable. In this project, values of a set of independent variables (to be discussed in future sections) for a patient produce a ‘Yes’ or ‘No’ prediction for the patient’s readmission status (‘ReAdmis’).

**B.2. Summary of method assumption**

KNN assumes the data resides in feature space where data points can be compared to one another with distance metrics such as Euclidean and Manhattan (Vishalmendekarhere, 2021 [1]). That is, the independent variables must be numeric. The *k* nearest neighbors (with an assigned label for the target variable) of a point determine its predicted label in the model. For the model to provide accurate results, this requires that nearby neighboring points must be similar to one another in that they have similar class labels. If nearby points are not similar, then the model’s predictions will be distorted and likely useless.

**B.3. Packages or libraries list**

I chose Python 3.9 for its ease of use, widespread adoption, personal familiarity, speed, multitude of mathematical and statistical packages, and extensive documentation and examples online for debugging and understanding. I prefer the consistent syntax of Python and its packages compared to libraries in R. Additionally, I wrote data cleaning functions in Python for the D206 course, making it convenient to repurpose that work here.

Numpy and pandas were used for handling numerical computations on arrays and dataframes, respectively. They’re effectively mandatory for any of the project to work. Matplotlib was needed to produce histograms and scatterplots. Scipy.stats was required for calculating z-scores to be used in creating z-score histograms for outlier searching. Statsmodels was used for variance inflation factor calculations when inspecting multicollinearity. From scikit-learn, various packages were used in selecting the best variables for a KNN model (SelectKBest() and f\_classif for F-tests); the needed calculations for the confusion matrix, classification report, accuracy score, cross validation, and area under the ROC curve; scalers for normalizing the data; KNeighborsClassifier() for creating the KNN model; and train\_test\_split was used to split the independent and dependent variables into training and test data.

A screen shot of a computer

Description automatically generated

**C.1. Data preprocessing**

In addition to the usual steps of data cleaning such as handling null values, addressing outliers, and verifying there are no duplicates, the KNN classifier requires one-hot encoding to utilize categorical variables as input, as it uses numerical data when computing metric distances between data points.

This is accomplished through the function one\_hot\_encoder which creates dummy variables (through pandas.get\_dummies(df[column])), converts the Boolean values to 32-bit integers (1s and 0s), and creates a new column of the form ‘{column\_name}\_{column\_label}’. For instance, a binary categorical variable like ‘Stroke’ introduces the column ‘Stroke\_Yes’. Binary categorical variables only require one encoded column to completely specify the presence or absence of a condition (i.e. the drop\_first=True condition is used in get\_dummies). For categorical variables withclasses, encoded columns are introduced for each class, as a KNN model cannot identify the presence of the *k*th dropped class through columns all recording 0.

See below:

A screen shot of a computer program

Description automatically generated

**C.2. Data set variables**

Before further processing in future sections (which will identify the final variables used in section D.2.), the independent variables to initially be used in the KNN model are the following:

Lat: numeric

Lng: numeric

Population: numeric

Area: categorical

TimeZone: categorical

Children: numeric

Age: numeric

Income: numeric

Marital: categorical

Gender: categorical

ReAdmis: categorical

VitD\_levels: numeric

Doc\_visits: numeric

Full\_meals\_eaten: numeric

vitD\_supp: numeric

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: numeric

TotalCharge: numeric

Additional\_charges: numeric

Item1: numeric

Item2: numeric

Item3: numeric

Item4: numeric

Item5: numeric

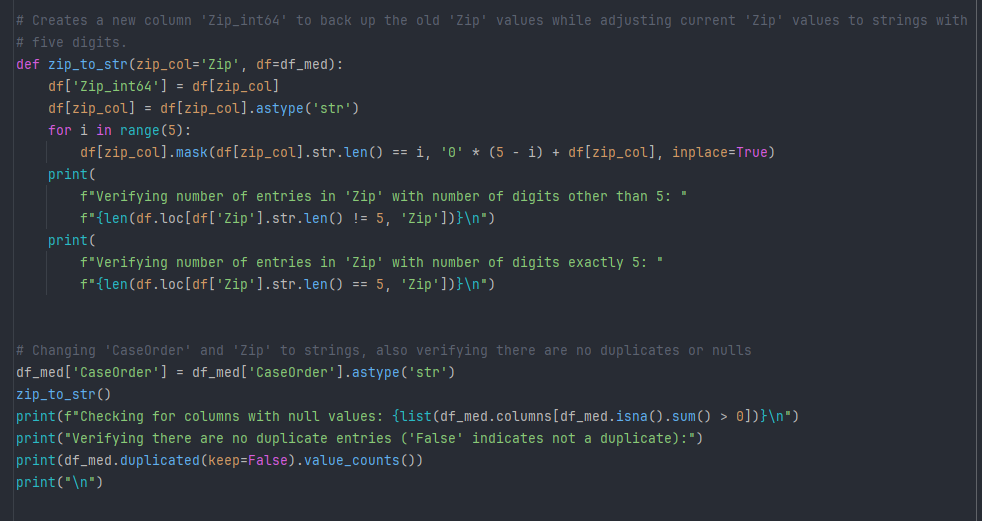
Item6: numeric

Item7: numeric

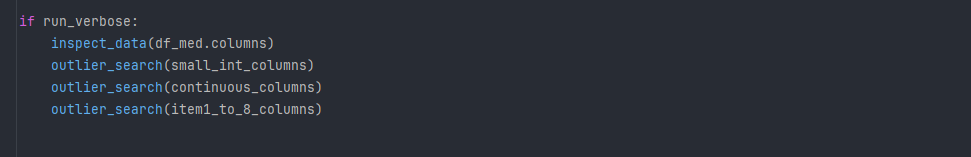
Item8: numeric

**C.3. Steps for analysis**

First, the dataset backs up integer ‘Zip’ values to a new column ‘Zip\_int64’ then replaces the values in the ‘Zip’ column with categorical strings of five digits (i.e. leading 0s are restored). ‘CaseOrder’ is also converted to a string as it and ‘Zip’ are more accurately regarded as categorical data. A few simple commands verify there are no null values or duplicates.



The functions inspect\_data and outlier\_search are used to verify no outliers distort the analysis and that column constraints are obeyed.



The function one\_hot\_encoder uses pandas.get\_dummies to create binary numeric columns for the presence or absence of a given class for each categorical feature in columns\_to\_encode. As mentioned in section C.1., drop\_first=True is only used on the binary categorical columns (that record ‘Yes’/’No’ values). This is a necessary transformation step for (raw) categorical variables to be valid input in a KNN model, as it requires numerical data to calculate distances between data points.

A screen shot of a computer program

Description automatically generated

As an initial step in inspecting correlations between the target variable and other columns, the function corr\_search displays correlation matrix values in descending order.

A screen shot of a computer program

Description automatically generated

**C.4. Cleaned data set**

For a dataset with outliers, duplicates, and null values addressed, as well as encoded columns, see “medical\_cleaned\_classification.csv”. For a subset of that file including only the final independent and dependent variables, see “medical\_transformed\_classification.csv”.

**D.1. Splitting the data**

The data is split into training and test data using the following:

A screenshot of a computer program

Description automatically generated

See “X\_train.csv” and “X\_test.csv” for the training and test data, respectively, for the independent variables. See “y\_train.csv” and “y\_test.csv” for the training and test data, respectively, for the dependent variable.

**D.2. Output and intermediate calculations**

Having preprocessed the data with the cleaning and transformation steps discussed in part C., scaled the independent variables using MinMaxScaler(), then split the independent and dependent variables into training and test data, we can now begin creating a KNN model. Using the initial “kitchen sink” approach discussed in section C.2., we define y\_0 and X\_0, with the dependent variable being ‘ReAdmis\_Yes’ (X\_1 will be used later):

A screen shot of a computer code

Description automatically generated

The function knn\_num\_search determined the highest accuracy occurs with a value of nearest neighbors:

A screen shot of a computer program

Description automatically generated

It has output:

knn\_num\_search(40, y\_0, X\_0, 'minmax', True, True)

A screenshot of a computer code

Description automatically generated

The KNN model is constructed with:

A screen shot of a computer program

Description automatically generated

With n\_neighbors = 29, it produces the following:

classifier\_knn(y\_0, X\_0, 'minmax', 29)

A screenshot of a computer

Description automatically generated

To improve upon this and find the best possible model, the helper function vif\_print(X) produces variance inflation factors on the independent variables contained in X. The feature\_selection function uses SelectKBest (from scikit-learn) and an F-test to find the best features for a KNN model on this dataset. It outputs a list of relevant features in descending order of importance (i.e. p-values in ascending order) and associated variance inflation factors:

A screen shot of a computer code

Description automatically generated

Output:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Noting the high variance inflation factors well above 5.0 in ‘TotalCharge’ and ‘Initial\_days’, and using the output of corr\_search mentioned in section C.3. finding a correlation coefficient of 0.988 between ‘TotalCharge’ and ‘Initial\_days’, we conclude that including both of these variables amounts to duplication. As ‘Initial\_days’ has a higher F-score, when we remove ‘TotalCharge’ and recalculate variance inflation factors:

A screenshot of a computer

Description automatically generated

The hyperparametric tuning function hyper\_search iteratively adds features on the basis of ascending p-values (as output by feature\_selection) and runs knn\_num\_search for a desired value of max\_n (i.e. evaluating KNN models with n\_neighbors = 1 through max\_n):

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

For max\_features=10, it has output:

A computer screen shot of numbers

Description automatically generated

A graph with blue dots

Description automatically generated

A graph with blue dots

Description automatically generated

A graph with blue dots

Description automatically generated

A graph with blue dots

Description automatically generated

In relative terms, a significant drop in accuracy and AUC occurs after five features. In the interest of model robustness for future data, having only one or two independent variables is inadequate and unstable. For a small decrease in accuracy and AUC, we select a KNN model with the following five independent variables:

* ‘Initial\_days’ (numeric)
* ‘Services\_CT Scan’ (one-hot encoded categorical)
* ‘Children’ (numeric)
* ‘Marital\_Divorced’ (one-hot encoded categorical)
* ‘Services\_Intravenous’ (one-hot encoded categorical)

Recalling top\_features and X\_1 as defined at the beginning of this section:

knn\_num\_search(40, y\_0, X\_1, 'minmax', True, True)

has output:

A screenshot of a computer code

Description automatically generated

A graph with blue dots

Description automatically generated

A graph with blue dots

Description automatically generated

Using with the final selection of independent variables in X\_1:

classifier\_knn(y\_0, X\_1, 'minmax', 8)

A screenshot of a computer

Description automatically generated

**D.3. Code execution**

The entire code can be found in the included file “classification.py”.

For the classification model portion using results found in section D.2., see below:

A screen shot of a computer program

Description automatically generated

**E.1. Accuracy and AUC**

The model accuracy is defined as the sum of the true positives and true negatives (correct predictions) divided by the total number of data points. The area under the curve (AUC) is the area under the receiver operator characteristic (ROC) curve, formed by plotting the true positive rate vs the false positive rate at threshold values as displayed in the plot below. The AUC represents the model’s ability to distinguish between classes, with 0.5 representing pure randomness and 1.0 corresponding to perfect predictions (Elleh, 2023 [2]).

The final KNN model from the above section has an accuracy of 0.9737 and an AUC of 0.9935. In contrast, the initial “kitchen sink” model had an accuracy of 0.8483 and AUC of 0.9488. For the selected five independent variables in the final model, the optimal number of nearest neighbors is to produce the highest accuracy (with a negligible decrease in AUC compared to ).

A graph with blue dots

Description automatically generated

**E.2. Results and implications**

The resulting KNN model from this project looking to predict patient readmission status has an accuracy of 0.9737, AUC of 0.9935, and uses the predictor variables ‘Initial\_days’, ‘Services\_CT Scan’, ‘Children’, ‘Marital\_Divorced’, and ‘Services\_Intravenous’.

The accuracy and AUC are sufficiently high to make it a very good predictor of patient readmissions with this dataset, but its long-term utility with additional data remains to be tested. As discussed in section D.2., the correlation between ‘Initial\_days’ and the target ‘ReAdmis\_Yes’ is very high at 0.8509, which reduces the relevance of the other four predictor variables. However, to only use one or two predictor variables risks future instability should the predictive power of ‘Initial\_days’ decrease or encounter outliers. For a small penalty in accuracy and AUC, the five predictor variables mentioned were used to provide some measure of resilience. The off-diagonal elements of the confusion matrix are quite similar, so the model’s inaccurate predictions aren’t heavily skewed towards false positives nor false negatives. The accuracy on the training data is 0.982, comparable to the accuracy on the test data of 0.9737. As the readmission rate is ~36.7%, the minority class isn’t completely overwhelmed by the majority, so the comparable accuracies indicate reliability (i.e. the model isn’t simply memorizing the training data labels). Stratification in the data split likely helped in balancing the errors.

**E.3. Limitation**

The extraordinarily high F-score of 26222 for ‘Initial\_days’ is followed by 5.95 for ‘Services\_CT Scan’ (‘TotalCharge’ is ignored here as it was found to effectively be a duplication of ‘Initial\_days’ and has a similarly high F-score). This extreme imbalance in feature scores, despite both having very low p-values, indicates the model is overly reliant on a single feature. While the accuracy, AUC, and performance of the model with the current data is remarkable, should there be outliers or significant changes in relationships between variables in the future, the model’s utility and accuracy would plummet. A more robust model would benefit from additional data collection, such as the reason a patient is readmitted, why they were admitted for the initial stay (beyond the type of admission such as “Emergency”), how long their second stay is, among other factors. As it currently is, this model works primarily due to the correlation between ‘Initial\_days’ and ‘ReAdmis’. Causative factors are not known at this time.

**E.4. Course of action**

In seeking an answer to the research question can we use existing patient data in a KNN model to accurately predict if a patient will need to be readmitted within a month of release, a statistically significant KNN model of five predictor variables (‘Initial\_days’, ‘Services\_CT Scan’, ‘Children’, ‘Marital\_Divorced’, and ‘Services\_Intravenous’) was found to have a very high accuracy of 0.9737 and AUC of 0.9935.

Hospital staff should use the predictions from this model as an early warning that a patient will likely be readmitted within 30 days. While their readmission may be inevitable in some cases, there are other times where the warning may be critically important for a physician to take extra precautions in ensuring the patient won’t need to return for the same condition within a few weeks. However, the predictive ability shouldn’t be mistaken for causation or absolute, and it will likely deteriorate over time. In particular, variables such as ‘Children’ and ‘Marital\_Divorced’ are almost certainly not causative but contain unknown relationships with underlying factors that weren’t collected in this dataset.

The model could be made more resilient to future data that will inevitably diminish its accuracy by increasing the dataset’s columns and level of detail. For instance, numerical blood pressure readings, patient diseases, and expected prognosis day-to-day would be very helpful in producing better approximations that are relatively future proof (and less dependent on the length of initial stay). Binary representations of conditions, such as blood pressure, may mask potential relationships to patient readmission status that could in turn create more accurate models.

**F. Panopto recording**

See the attached link: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=b22de431-5fc0-4561-b61f-b165013061c9

**G. Sources for third-party code**

**1.** WGU. 2024. D209 Data Mining I “Data Sets and Associated Data Dictionaries”. Medical Data and Dictionary Files. Retrieved April 29, 2024, from <https://access.wgu.edu/ASP3/aap/content/g9rke9s0rlc9ejd92md0.html>.

**H. Sources**

**1.** Vishalmendekarhere. 2021. “It’s all about Assumptions, Pros & Cons”. Retrieved April 29, 2024 from <https://medium.com/swlh/its-all-about-assumptions-pros-cons-497783cfed2d>.

**2.** Elleh, Festus. 2023. WGU “D209 Data Mining 1 Task 1 Cohort.pptx”. Retrieved May 3, 2024 from https://westerngovernorsuniversity.sharepoint.com/:p:/r/sites/DataScienceTeam/\_layouts/15/Doc.aspx?sourcedoc=%7B945F58A7-B99E-4D7A-BEC0-9BB216B4D2BD%7D&file=D209%20Data%20Mining%201%20Task%201%20Cohort.pptx&action=edit&mobileredirect=true.