D209 Performance Assessment

Task 1

Adrienne Stafford

06/07/24

**Part I: Research Question**

A1. Proposed Question

How can we classify patients as "High Risk" or "Low Risk" for hospital readmission (ReAdmis)? And which variables are most significant to Readmission?

A2. The Goal

The primary goal of the data analysis is to develop a predictive model that classifies patients into "High Risk" or "Low Risk" categories for hospital readmission within 30 days. By identifying individuals who are more likely to require a readmission based on their medical, demographic, and health characteristics, this classification helps healthcare practitioners to tailor treatments to lower readmission rates.

**Part II: Data Preparation**

B1. Classification Method

The K-Nearest Neighbor (KNN) method will analyze the dataset by normalizing numerical variables such as Age, Income, and VitD\_Level, and encoding categorical ones like ViD\_supp and Complication\_risk. The K nearest neighbors will then be determined using KNN using the Euclidean distance calculation between a new patient and current patients. Depending on the majority class among these neighbors, the new patient will be categorized as either "Low Risk" or "High Risk" for readmission. By comparing fresh patient data with previously identified cases, this method enables real-time classification with the goal of achieving high accuracy and offering insights into significant variables like the likelihood of complications and medical visits. Healthcare practitioners will be able to forecast readmission risk, manage resources, and enhance patient outcomes with the help of this model, which will be assessed using metrics like accuracy and precision and confirmed by cross-validation more accurately (Nelson, 2020).

B2. Assumption

The K-Nearest Neighbor (KNN) classification approach makes the important assumption that similar occurrences are located near to one another in the feature space. This suggests that patients with comparable profiles—that is, with comparable health, medical, and demographic traits—will also have comparable distances and, as a result, similar readmission risk classifications. In order to ensure that patients in the dataset who are closer are in fact more alike and have the same categorization, KNN requires that the features used to evaluate similarity are valid and suitably scaled. The model's capacity to correctly categorize new patients by comparing them to their closest neighbors in the dataset is predicated on this premise.

B3. Packages

The packages I have chosen for the analysis are as follows:

• Matplot.lib.pylot- An object-oriented charting library called Matplotlib has a procedural interface called PyLab. The complete package is called matplotlib.Matplotlib has a module called pyplot, and PyLab is a module that is installed in addition to Matplotlib. PyLab is a handy package that imports matplotlib in bulk. NumPy (for math and dealing with arrays) and pyplot (for graphing) in one name space (M, 2024).

• Pandas- Pandas works mostly with data in 1-D and 2-D arrays, much like Numpy, although it does so in a different way. 1-D arrays are called series in pandas. The pd.Series constructor, which takes numerous optional arguments, is used to generate a series. Data, which lists the components of the series, is the most often used argument (What is pandas python?).

• NumPy- The core Python library for scientific computing is called NumPy. The multidimensional array object, different derived objects (like masked arrays and matrices), and a variety of routines for quick array operations—like sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation, and much more—are all provided by this Python library (What is numpy?, 2008).

• Scikit-learn- The most reliable and practical Python machine learning library is called Scikit-learn, or Sklearn. Through a Python consistency interface, it offers a range of effective tools for statistical modeling and machine learning, including as regression, clustering, classification, and dimensionality reduction. This library is based on NumPy, SciPy, and Matplotlib and is mostly developed in Python. As of 2021, TutorialsPoints.com Datasets, KNeighborsClassifer, Train\_test\_split, cross\_val\_score, GridSearchCV, metrics, accuracy\_score, classification\_report, confusion\_matrix, preprocessing, StandardScaler, and Pipeline are the tools that will be used in the sklearn analysis (M, 2024).

• Seaborn- A Python package called Seaborn is used to create statistical visualizations. It strongly integrates with pandas data structures and builds upon the matplotlib framework. Seaborn facilitates data exploration and comprehension. Its charting functions work with data frames and arrays that hold entire datasets, and they internally carry out the statistical aggregation and semantic mapping required to create visually appealing graphs. You may concentrate on the meaning of the various plot parts rather than the specifics of how to design them thanks to its declarative, dataset-oriented API (Melanie, 2023).

Part IV: Analysis

C1. Processing Goal

Normalizing numerical variables is crucial for preprocessing data when using the K-Nearest Neighbor (KNN) classification method. Since KNN relies on distance calculations to identify nearest neighbors, the magnitude of the variables can significantly impact the results. Variables with larger ranges, such as Age, Income, VitD\_Level, and Initial\_days, can skew the distance metric if they are not on a similar scale. Normalizing these variables ensures that each one equally influences the distance calculations, enhancing the accuracy and reliability of the classification. By scaling numerical variables to a comparable range, often between 0 and 1, normalization preserves the integrity of distance measurements, allowing KNN to more accurately determine the nearest neighbors based on true similarities in patient profiles

C2. Dataset Variables

|  |
| --- |
| Continuous Variables |
| Age |
| Income |
| VitD\_levels |
| Doc\_visits |
| Children |
| Initials\_days |
| TotalCharge |
| Additional\_charges |
| VitD\_supp |
|  |
| Categorical Variables |
| Area |
| Marital |
| Gender |
| ReAdmis |
| Soft\_drink |
| Initial\_admin |
| Complication\_risk |
| HighBlood |
| Stroke |
| Overweight |
| Arthritis |
| Diabetes |
| Hyperlipidemia |
| BackPain |
| Anxiety |
| Allergic\_rhinitis |
| Reflux\_esophagitis |
| Asthma |
| Services |
| Item1 |
| Item2 |
| Item3 |
| Item4 |
| Item5 |
| Item6 |
| Item7 |
| Item8 |

C3. Data Preparations

C4. Cleaned Data Set

See attached CSV file

**Part IV: Analysis**

D1. Training and Test Data Sets

# Re-read prepared dataset

med\_df = pd.read\_csv('Med\_prepared\_D209t1.csv')

# Set predictor features & target variable

X = med\_df.drop('ReAdmis\_num', axis=1).values

y = med\_df['ReAdmis\_num'].values

# Create training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20, random\_state =1)

# Export X\_train dataset

X\_train\_df = pd.DataFrame(X\_train)

X\_train\_df.to\_csv('X\_traint1.csv')

# Export X\_test dataset

X\_test\_df = pd.DataFrame(X\_test)

X\_test\_df.to\_csv('X\_testt1.csv')

# Export y\_train dataset

y\_train\_df = pd.DataFrame(y\_train)

y\_train\_df.to\_csv('Y\_traint1.csv')

# Export y\_test dataset

y\_test\_df = pd.DataFrame(X\_test)

y\_test\_df.to\_csv('Y\_testt1.csv')

See attached CVS files

D2: Technique

My training and test data sets are then formed, and after that, I fit the data sets into the model to build a new array that I name y\_pred. My method for properly analyzing the data was to calculate the K nearest neighbors model's accuracy score. The next step is to investigate if scaling the data sets can result in a more accurate model once the accuracy score has been established. This is to make sure we have thoroughly examined the model.

A close-up of a sign

Description automatically generated



A screenshot of a number

Description automatically generated



A number of numbers in a row

Description automatically generated with medium confidence

A number with black text

Description automatically generated with medium confidence

A blue squares with white text

Description automatically generated

D3: Classification Code

# Initialize KNN model

knn = KNeighborsClassifier(n\_neighbors = 7)

# Fit data to KNN model

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

# Predict outcomes from test set

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

# Export y\_pred dataset

y\_pred\_df = pd.DataFrame(y\_pred)

y\_pred\_df.to\_csv('Y\_predt1.csv')

# Print initial accuracy score of KNN model

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

# Compute classification metrics

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

# Compute classification metrics

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

# Initiate pipeline

pipeline = Pipeline(steps)

# Split dataframe

X\_train\_scaled, X\_test\_scaled, y\_train\_scaled, y\_test\_scaled = train\_test\_split(X, y, test\_size = 0.2, random\_state = 1)

# Scale dateframe with pipeline object

knn\_scaled = pipeline.fit(X\_train\_scaled, y\_train\_scaled)

# Predict from scaled dataframe

y\_pred\_scaled = pipeline.predict(X\_test\_scaled)

# Print 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)))

# Compute classification metrics after scaling

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

#Confusion\_matrix & generate results

cf\_matrix = confusion\_matrix(y\_test, y\_pred)

print(cf\_matrix)

# Visual confusion 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='Blues')

**Part V: Data Summary and Implications**

E1: Explanation of AUC and Accuracy

The evaluation metrics indicate that the categorization model performs well. With an 88% accuracy rate, 88% of its predictions agree with the actual class designations. By calculating the percentage of accurate forecasts among all predictions, accuracy offers a clear evaluation of the model's overall performance.

Furthermore, the model has a high degree of class distinction, as indicated by an area under the curve (AUC) of 0.9699. In particular, the Receiver Operating Characteristic (ROC) curve's AUC shows how well the model can distinguish between positive and negative classes. An AUC near 1 (e.g., 0.9699) indicates that the model performs exceptionally well in prioritizing positive events over negative ones.

E2: Results and Implications

With an accuracy of 88%, the classification analysis's results show that the model operates robustly and that 88% of the predictions match the actual class labels. To properly understand the model's success, however, this high accuracy needs to be analyzed in conjunction with other performance criteria. Class 0 has a precision, recall, and F1-score of 0.94, 0.87, and 0.90, but class 1 has these values of 0.80, 0.91, and 0.85. These measures indicate that although the model has a somewhat lower precision, it is more accurate at class 0 prediction and also captures most positive events in class 1. The model's capacity to rank positive occurrences higher than negative ones with a 96.99% probability is demonstrated by its good discrimination across classes, as indicated by its area under the curve (AUC) of 0.9699.

These findings have notable implications. The model is useful for applications where differentiation between positive and negative classes is crucial, such fraud detection or medical diagnosis, because of its high AUC, which highlights its strong capacity to do so. The reduced precision for class 1 indicates, however, that false positives must be decreased. This might be done by adjusting the decision threshold or by utilizing cost-sensitive learning strategies. In situations when both false positives and false negatives have substantial effects, maintaining this equilibrium is essential. The model's overall performance indicates that it may be trusted to make decisions in important applications; however, more validation on other datasets is advised to guarantee the model's consistency and generalizability. In summary, the model has great overall performance; nevertheless, its practical usability and reliability will be improved if precision for class 1 is improved.

E3: One Implication

The imbalance between the classes, which is represented in the performance measures, is one obvious shortcoming of the data analysis. In particular, the support numbers show that there is a difference in the number of class 0 instances (1261 instances) compared to class 1 instances (739 instances). The model's performance and how its measurements are interpreted may be affected by this mismatch. The findings may be skewed even though the data analysis shows good overall performance metrics due to the imbalance between classes. By addressing this imbalance, different approaches can guarantee the model's robustness in a variety of real-world circumstances and result in a more accurate and equitable evaluation of the model's capabilities.

E4: Recommendation

The organization can improve the model's capacity to correctly categorize patients' readmission risk by correcting class imbalance, adjusting the threshold, and concentrating on important variables. Reducing hospital readmissions and enhancing patient care can be achieved by incorporating the model into clinical practice and using it to direct preventative actions. The model's continued effectiveness and responsiveness to actual circumstances will be ensured by ongoing validation and improvement.

F: Video

See Attached Link

G: Web Sources

M, R. (2024a, January 23). *How to classify data in python using Scikit-Learn*. ActiveState. https://www.activestate.com/resources/quick-reads/how-to-classify-data-in-python/

H: In-Text Sources

M, R. (2024a, January 23). *What is pyplot in Matplotlib*. ActiveState. https://www.activestate.com/resources/quick-reads/what-is-pyplot-in-matplotlib/

M, R. (2024b, January 23). *What is scikit-learn in python?*. ActiveState. https://www.activestate.com/resources/quick-reads/what-is-scikit-learn-in-python/

Melanie. (2023, October 9). *Seaborn: Everything you need to know about the python data visualization tool*. DataScientest. https://datascientest.com/en/seaborn-everything-you-need-to-know-about-the-python-data-visualization-tool

Nelson, D. (2020, August 23). *What is a KNN (K-nearest neighbors)?*. Unite.AI. https://www.unite.ai/what-is-k-nearest-neighbors/

*What is numpy?#*. What is NumPy? - NumPy v2.0 Manual. (2008). https://numpy.org/doc/stable/user/whatisnumpy.html

*What is pandas python?*. NVIDIA Data Science Glossary. (n.d.). https://www.nvidia.com/en-us/glossary/pandas-python/