D209 Performance Assessment

Task 2

Adrienne Stafford

06/17/24

Part I: Research Question

A1: Question

Can we predict the readmission status of patients based on their medical and demographic information using a random forest classifier?

A2: Goal

The main goal of this study is to create a prediction model that can effectively identify patients who are at a high risk of returning to the hospital soon after being discharged. By enabling healthcare practitioners to perform customized post-discharge interventions—like close monitoring, follow-up appointments, and personalized support—that directly address individual health concerns that could otherwise result in readmission, this strategy aims to improve patient care. Hospitals may also more effectively and strategically manage resources by predicting which patients are most likely to return, which lowers the number of avoidable readmissions and the associated expenses. By lowering medical costs, this strategy not only helps healthcare systems financially, but it also benefits patients and payers. Accurate forecasts also maximize the use of available resources by enabling hospitals to focus their administrative and medical attention on patients who pose a greater risk. This improves patient care and frees up space for more urgent cases and new admissions.

**Part II: Method Justification**

B1: Method

As part of its ensemble learning process, Random Forest builds several decision trees during training and outputs the class mode for classification. It offers a decent balance between variation and bias and works well when dealing with a big number of input variables. In order to classify cases as either "readmitted" or "not readmitted," the model will yield probabilities of readmission (2021).

B2. Assumption

The basic premise of random forests as a machine learning technique is that the training data is representative of the general population. This means that the diversity and variability of the full dataset or real-world scenario from which the model is trained should be appropriately reflected in the sample of data used to train the model. The random forest model should work well and generalize to new, unobserved data if the training data accurately depicts the patterns and relationships found in the population. The model's performance, however, may suffer if the training data is skewed, constrained, or does not fully capture the diversity and complexity of the population. If these biases or errors are learned and replicated, it could produce inaccurate predictions and choices when used in larger contexts. For the random forest model to be reliable and accurate, the training data must be complete and objective since this directly affects the model's capacity to forecast outcomes and make defensible conclusions based on patterns seen in real-world data. This presumption emphasizes how crucial thorough data collection and preparation are to creating reliable and successful predictive models (2021).

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).

Part III: Data Preparation

C1: Preparation Goal

Encoding categorical variables into a numerical format is a crucial data preparation objective related to creating the random forest model for predicting hospital readmissions. Because random forest models split data according to number thresholds in order to produce decision trees, they need numerical inputs. Categorical variables including gender, Initial\_admin, services, and Complication\_risk are present in many healthcare datasets, including the one used in this investigation, however the model cannot use these variables directly in their raw format.

Categorical variables need to be converted into a format that the model can understand in order to solve this. This entails employing methods like one-hot encoding or label encoding to translate categories into numerical representations. By generating binary columns for every category level by one-hot encoding, the model can handle each category independently of the others without assuming an ordinal relationship between them. While label encoding might be useful, it may also inadvertently suggest an ordinal relationship between categories because it gives each category level a distinct integer.

C2: 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: Preparation

The dataset has to be prepared for analysis in a few crucial ways in order for the random forest model to work with it. Pandas was initially used to load the data from the CSV file into a DataFrame, giving the data a structured format for additional processing. In order to preserve data integrity, missing values were handled in the following phase by either removing rows with missing entries or filling them with appropriate estimations, like the median. After that, one-hot encoding was used to convert categorical information including gender, ReAdmis, Soft\_drink, Initial\_admin, and Complication\_risk into a numerical format. In order to facilitate their use in the model, this technique constructed binary columns for each category level, meaning no ordinal linkages between them.

Then, using the StandardScaler from the Scikit-learn module, numerical variables including age, the number of vitamin D taken, the length of stay and the total charge. By guaranteeing consistent input ranges, this normalization step improved the model's performance by standardizing the range of these variables. Following these preprocessing stages, the predictors were isolated from the desired outcome by splitting the data into feature variables (X) and the goal variable (Readmission). Ultimately, the dataset was split into training and test sets, with 30% of the data set aside for assessing the model's performance and 70% of the data used for training. This division made it possible to evaluate the model's capacity to generalize to fresh, untested data.

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.preprocessing import LabelEncoder, StandardScaler

# Load data

data = pd.read\_csv('medical\_clean.csv')

# Handle missing values

data = data.dropna()

# Examining fist five records of dataset

data.head()

# Viewing DataFrame descriptive information

data.info

# Getting an overview of descriptive stats

data.describe()

# Getting data types of features

data.dtypes

# Renaming the last 8 Survey Columns for better description of variables

data.rename(columns = {'Item1' : 'TimelyResponse',

'Item2' : 'Fixes' ,

'Item3' : 'Replacements' ,

'Item4' : 'Reliability' ,

'Item5' : 'Options' ,

'Item6' : 'Respectfulness' ,

'Item7' : 'Courteous' ,

'Item8' : 'Listening'},

inplace=True)

# Converting ordinal categorical data into numeric variables

data['Area\_num'] = data.Area.map({'Rural' : 0, 'Urban' : 1, 'Suburban' : 2})

data['Marital\_num'] =data.Marital.map({'Never Married' : 0, 'Married' : 1, 'Separated' : 2, 'Divorced' : 3, 'Widowed' : 4})

data['Gender\_num'] = data.Gender.map({'Nonbinary' : 0, 'Male' : 1, 'Female' : 2})

data['Initial\_admin\_num'] = data.Initial\_admin.map({'Elective Admission' : 0, 'Observation Admission' : 1, 'Emergency Admission' : 2})

data['Complication\_risk\_num'] = data.Complication\_risk.map({'Low' : 0, 'Medium' : 1, 'High' : 2})

data['Services\_num'] = data.Services.map({'Blood Work' : 0, 'Intravenous' : 1, 'CT Scan' : 2, 'MRI' : 3})

# Converting binary categorical variables to numeric variables

data['ReAdmis\_num'] = [1 if v == 'Yes' else 0 for v in data['ReAdmis']]

data['Soft\_drink\_num'] = [1 if v == 'Yes' else 0 for v in data['Soft\_drink']]

data['HighBlood\_num'] = [1 if v == 'Yes' else 0 for v in data['HighBlood']]

data['Stroke\_num'] = [1 if v == 'Yes' else 0 for v in data['Stroke']]

data['Overweight\_num'] = [1 if v == 'Yes' else 0 for v in data['Overweight']]

data['Arthritis\_num'] = [1 if v == 'Yes' else 0 for v in data['Arthritis']]

data['Diabetes\_num'] = [1 if v == 'Yes' else 0 for v in data['Diabetes']]

data['Hyperlipidemia\_num'] = [1 if v == 'Yes' else 0 for v in data['Hyperlipidemia']]

data['BackPain\_num'] = [1 if v == 'Yes' else 0 for v in data['BackPain']]

data['Anxiety\_num'] = [1 if v == 'Yes' else 0 for v in data['Anxiety']]

data['Allergic\_rhinitis\_num'] = [1 if v == 'Yes' else 0 for v in data['Allergic\_rhinitis']]

data['Reflux\_esophagitis\_num'] = [1 if v == 'Yes' else 0 for v in data['Reflux\_esophagitis']]

data['Asthma\_num'] = [1 if v == 'Yes' else 0 for v in data['Asthma']]

#Drop original categorical features from dataframe for further analysis

data = data.drop(columns=['ReAdmis', 'Area', 'Marital', 'Gender', 'Initial\_admin',

'Complication\_risk' , 'Services', 'Soft\_drink', 'HighBlood',

'Stroke', 'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia',

'BackPain', 'Anxiety', 'Allergic\_rhinitis', 'Reflux\_esophagitis', 'Asthma'])

#Remove the other less meaningful categorical variables from dataset to provide fully numerical dataframe for further analysis

data = data.drop(columns=['Customer\_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'TimeZone', 'Job'])

# Save the cleaned data

data.to\_csv('medical\_clean\_prepared.csv', index=False)

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

# Define target and features

X = data.drop('ReAdmis\_num', axis=1)

y = data['ReAdmis\_num']

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Save the training and testing data

X\_train.to\_csv('D209\_T2\_X\_train.csv', index=False)

X\_test.to\_csv('D209\_T2\_X\_test.csv', index=False)

y\_train.to\_csv('D209\_T2\_y\_train.csv', index=False)

y\_test.to\_csv('D209\_T2\_y\_test.csv', index=False)

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, mean\_squared\_error, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Initialize and train the model

rf = RandomForestClassifier(n\_estimators=100, random\_state=42)

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

# Make predictions

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

# Calculate accuracy and MSE

accuracy = accuracy\_score(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

# Confusion matrix

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

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

# Print results

print(f'Accuracy: {accuracy}')

print(f'Mean Squared Error: {mse}')

C4: Prepared Data

See attached

**Part IV: Analysis**

D1: Training and Test Split

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

# Define target and features

X = data.drop('ReAdmis\_num', axis=1)

y = data['ReAdmis\_num']

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Save the training and testing data

X\_train.to\_csv('D209\_T2\_X\_train.csv', index=False)

X\_test.to\_csv('D209\_T2\_X\_test.csv', index=False)

y\_train.to\_csv('D209\_T2\_y\_train.csv', index=False)

y\_test.to\_csv('D209\_T2\_y\_test.csv', index=False)

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, mean\_squared\_error, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Initialize and train the model

rf = RandomForestClassifier(n\_estimators=100, random\_state=42)

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

# Make predictions

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

# Calculate accuracy and MSE

accuracy = accuracy\_score(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

# Confusion matrix

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

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

# Print results

print(f'Accuracy: {accuracy}')

print(f'Mean Squared Error: {mse}')

See attached files

D2: Analysis Technique

The prepared data was processed by the Random Forest technique in an efficient manner. It learned from the attributes to predict hospital readmissions with a robustness and accuracy balance. Insights into the model's performance were obtained by intermediate computations, which included model training, prediction, and evaluation. This helped to guarantee that the model can accurately identify high-risk patients and assist focused actions. These procedures and measurements show how effectively the model generalizes to fresh data, which makes it an effective tool for enhancing patient care outcomes and predicting readmissions.

A group of blue and white graphs

Description automatically generated

A graph with numbers and lines

Description automatically generated

A green and white rectangular chart

Description automatically generated

A green rectangle with numbers

Description automatically generated

A graph of different colored columns

Description automatically generated with medium confidence

A graph with different colored bars

Description automatically generated

A graph with different colored squares

Description automatically generated with medium confidence

A graph with red dots

Description automatically generated

A graph with red dots

Description automatically generated

A graph with red dots

Description automatically generated

A screenshot of a graph

Description automatically generated

D3: Analysis Code

# Save the cleaned data

data.to\_csv('medical\_clean\_prepared.csv', index=False)

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

# Define target and features

X = data.drop('ReAdmis\_num', axis=1)

y = data['ReAdmis\_num']

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Save the training and testing data

X\_train.to\_csv('D209\_T2\_X\_train.csv', index=False)

X\_test.to\_csv('D209\_T2\_X\_test.csv', index=False)

y\_train.to\_csv('D209\_T2\_y\_train.csv', index=False)

y\_test.to\_csv('D209\_T2\_y\_test.csv', index=False)

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, mean\_squared\_error, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Initialize and train the model

rf = RandomForestClassifier(n\_estimators=100, random\_state=42)

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

# Make predictions

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

# Calculate accuracy and MSE

accuracy = accuracy\_score(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

# Confusion matrix

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

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

# Print results

print(f'Accuracy: {accuracy}')

print(f'Mean Squared Error: {mse}')

**Part V: Data Summary and Implications**

E1: Accuracy and Mean squared error (MSE)

The Random Forest model, which was created to forecast patient readmissions, performs exceptionally well, as seen by its 0.98 accuracy and 0.02 mean squared error (MSE). The accuracy of the model, which quantifies the percentage of accurate predictions, indicates that it anticipated patients' readmission status 98% of the time. This suggests that the model performs a very good job of identifying patients who will be readmitted versus those who won't. The accuracy metric, which emphasizes the model's capacity to reduce misclassification mistakes, is computed as the ratio of true positive and true negative predictions to the total number of predictions. Specifically, the confusion matrix reveals that out of all predictions, 1909 were true negatives (correctly identified as not readmitted), and only 25 were false positives (incorrectly identified as readmitted).

By calculating the average squared difference between the anticipated values and the actual outcomes, the MSE not only measures accuracy but also offers insight into prediction errors. The model's little mistakes are reflected in the low MSE value of 0.02, which shows that the predictions and actual readmission outcomes are fairly similar. The model's accuracy and dependability in anticipating patient readmissions are highlighted by its low mean square error (MSE), which implies that the model makes relatively few errors and that the size of those errors is minimal. When taken as a whole, these measures demonstrate that the Random Forest model is a reliable method for precisely identifying patients who are at risk of readmission, allowing medical professionals to carry out focused treatments that enhance patient care and lower readmission rates.

E2: Results and Implications

With a mean squared error (MSE) of 0.02 and an accuracy of 0.98, the Random Forest model demonstrated its potent predictive ability for hospital readmissions. The confusion matrix demonstrates that, with just 25 false positives, the model successfully detected all genuine readmitted cases and classified 1909 individuals as not readmitted. It may be inferred from the model's high accuracy and low MSE that it can accurately identify patients who will and won't be readmitted. The approach can be utilized by healthcare providers to proactively identify individuals who pose a risk, conduct focused interventions, and enhance patient outcomes. These are substantial implications.

Additionally, by avoiding needless readmissions and maximizing resource allocation, the approach can assist in lowering healthcare expenditures. Precise forecasts aid medical facilities in fulfilling regulatory requirements and informing data-driven choices that improve patient care and administrative procedures. Even if the model works well, its dependence on the training data's representativeness exposes a generalizability flaw and calls for further improvements using a wider range of data sources. All things considered, the Random Forest model is a useful tool for enhancing patient outcomes and healthcare productivity.

E3: Limitation

The predictability of our data analysis is severely limited by the prediction model's lack of generalizability. The training data must be representative of the larger patient population in order for the Random Forest model to be accurate. The model could not function well on different or unseen data if the training set does not reflect a range of patient demographics or varied hospital situations. This restriction implies that although the model performs well on the given dataset, its predictions may not hold true in other patient populations or hospital settings. Future work should incorporate more representative and diverse data to help offset this, and the model should be updated frequently to take into account fresh trends and patterns in the data.

E4: Recommendation

For high-risk patients that the Random Forest model identified, the healthcare organization should apply a tailored intervention approach based on the model's accuracy in predicting hospital readmissions. This tactic entails creating customized post-discharge care plans with improved follow-up plans, home health services, and patient education for efficient condition management at home. High-risk patients should be given priority for extra nursing care and quick access to outpatient services in order for the organization to allocate resources as efficiently as possible. Enhancing care coordination is critical, as continuous communication between healthcare professionals is ensured by integrated care teams and electronic health record (EHR) systems. It will be possible for the organization to track readmission rates and modify strategies in response to results with regular monitoring and evaluation of the intervention's efficacy. Furthermore, the accuracy and usefulness of the model will be improved by continuously updating it and broadening the scope of data collection to cover a more diverse patient group. This proactive, data-driven strategy seeks to guarantee effective use of healthcare resources, lower readmission rates, and enhance patient outcomes.

**Part VI: Demonstration**

F: Video

See Attached

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/

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/

*What is Random Forest?*. IBM. (2021, October 20). https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems.