**Task 1: Classification Data Mining Models**

Joseph Emmanuel Cayetano

Western Governors University

D603: Machine Learning Task 1

January 23, 2025

1. **Create your subgroup and project in GitLab**

**GitLab Repository URL**: <https://gitlab.com/wgu-gitlab-environment/student-repos/jcayet5/d603-machine-learning/-/tree/master?ref_type=heads>

**Repository Branch History Screenshot**

A screenshot of a computer

Description automatically generated

**B1. Propose one question relevant to a real-world organizational situation that you will answer using one of the following classification methods**

One question that is relevant to a real-world organizational situation that I will answer using the Random Forest classification method is: What factors are the most predictive of patient readmission within a month of release?

**B2. Define one goal of the data analysis**

The goal of this data analysis is to identify the most significant factors that predict patient readmission within a month of release, using demographic, medical, and hospitalization data. This will help the hospital chain to prioritize targeted interventions and resource allocation to reduce readmission rates and avoid penalties from organizations like the Centers for Medicare and Medicaid Services (CMS).

This goal is reasonable as the dataset includes relevant variables such as patient demographics, medical conditions, and hospitalization details, which are related to the problem of patient readmissions.

**C1. Explain how the classification method you chose analyzes the selected dataset. Include expected outcomes**

The Random Forest classification method analyzes the medical dataset in multiple steps. The first step involves preprocessing the dataset. The dataset is prepared by handling missing values and encoding categorical variables. Scaling is not required for Random Forest. Relevant features are selected from the dataset, such as patient demographics, medical conditions, and hospitalization details. The target variable, ReAdmis (indicating if a patient was readmitted within a month), is separated from the predictor variables. The second step involves model training. The Random Forest algorithm creates multiple decision trees. Each decision tree in the Random Forest is “built from a different subset of data and features” (Sruthi, 2024, par. 46). This is to reduce overfitting and improve generalization. The target variable, ReAdmis, is used to train the model, with the goal of predicting whether a patient will be readmitted. The third step involves majority voting. Each decision tree in the forest makes a prediction. The final prediction is based on majority voting across all trees to determine the most likely class for each patient. The fourth step involves feature importance. The Random Forest algorithm calculates the importance of each feature by measuring how much it reduces impurity across all decision trees in the forest. This helps identify which factors are most predictive of patient readmission.

Using the Random Forest classification method on the medical dataset yields multiple outcomes. The first outcome involves assessing the model’s ability to predict readmissions using metrics like accuracy, precision, and recall. The second outcome involves identifying the most important factors influencing patient readmission. This will help hospitals prioritize intervention strategies. The third outcome involves uncovering patterns such as patients with certain conditions or demographic characteristics being at higher risk of readmission. This can help hospitals design programs, such as improved follow-up care for high-risk patients.

**C2. List the packages or libraries you have chosen for Python and justify how each item on the list supports the analysis**

There are three Python libraries that I used for the Random Forest classification analysis. First is the pandas library and it was used to load, manipulate, and preprocess the medical dataset. The to\_csv() function from the pandas library was also used to export the training, validation, and test datasets as CSV files. Second is the scikit-learn library and it provides the RandomForestClassifier() function, which I used to build and train the Random Forest model. Scikit-learn provides functions like accuracy\_score(), precision\_score(), and recall\_score(), which I used to evaluate the model’s performance. Scikit-learn also includes tools for hyperparameter tuning, such as GridSearchCV() and StratifiedKFold(), to optimize the Random Forest model. Third is the NumPy library and it was used implicitly by scikit-learn for computations during the Random Forest model training and evaluation.

**D1. Describe one data preprocessing goal relevant to the classification method from part B1**

One data preprocessing goal for using the Random Forest classification method is to handle categorical variables by encoding them into numerical formats. The Random Forest algorithm cannot process categorical data directly, as the algorithm needs numerical inputs to compute splits and make predictions. The medical dataset contains several categorical variables like ReAdmis (Yes/No), Gender (Male/Female), Area (Urban/Suburban/Rural), Services (Blood Work or CT Scan), etc. These variables must be converted into numerical format for use in the Random Forest algorithm, typically through label encoding. Label encoding assigns each category a unique integer value.

**D2. Identify the initial dataset variables that you will use to perform the analysis for the classification question from part B1, and classify each variable as continuous or categorical**

The variables that I will use to perform the analysis for the Random Forest classification question from part B1 are: ReAdmis(categorical), Age(continuous), Gender(categorical), Income(continuous), Marital(categorical), Area(categorical), HighBlood(categorical), Stroke(categorical), Diabetes(categorical), Overweight(categorical), Arthritis(categorical), Complication\_risk(categorical), Initial\_admin(categorical), Initial\_days(continuous), Doc\_visits(continuous), VitD\_levels(continuous), Full\_meals\_eaten(continuous), Soft\_drink(categorical), vitD\_supp(continuous), TotalCharge(continuous), and Additional\_charges(continuous).

**D3. Explain each of the steps used to prepare the data for the analysis. Identify the code segment for each step**

A screenshot of a computer

Description automatically generated

I used the read\_csv() function from the pandas library to load the medical dataset into a DataFrame for analysis. The output from the head() function verifies that the medical dataset was loaded successfully.

A screenshot of a computer program

Description automatically generated

These are the variables that I selected for analysis: ReAdmis, Age, Gender, Income, Marital, Area, HighBlood, Stroke, Diabetes, Overweight, Arthritis, Complication\_risk, Initial\_admin, Initial\_days, Doc\_visits, VitD\_levels, Full\_meals\_eaten, Soft\_drink, vitD\_supp, TotalCharge, and Additional\_charges. The dataset is filtered to keep only the specified variables, and all other variables are removed. The output from the info() confirms the selected variables exist and shows their data types and non-null value counts.

A screenshot of a computer

Description automatically generated

I used the isnull() function with the sum() function to count missing values in each column of the medical dataset. Since there are no missing values, we can skip handling them and move straight to encoding categorical variables.

A screenshot of a computer

Description automatically generated

I imported the LabelEncoder() function from the sklearn library to convert categorical variables into numerical values for Random Forest, which only works with numerical inputs. I used a for loop to iterate over each categorical variable and apply label encoding. I used the head() function to display the first 5 rows of the updated medical dataset after label encoding.

**D4. Provide a copy of the cleaned dataset**

Cleaned\_dataset.csv is the medical dataset that has been properly prepared for analysis using the Random Forest classification method. I included this file in my submission.

**E1. Split the data into training, validation, and test datasets and provide the file(s)**

A screenshot of a computer program

Description automatically generated

Before splitting the data, I separated the predictors and the target variable, ReAdmis. I assigned the predictors to variable X and the target variable to y. Next, I used the train\_test\_split() function from the sklearn library to split the data into training and test sets. Then I split the training set into training and validation sets. The training set is 56%, the validation set is 24%, and the test set is 20% of the original data.

A screenshot of a computer program

Description automatically generated

I combined features and target for the training, validation, and test sets, and exported all three as CSV files. The training set is saved as “training\_data.csv.” The validation set is saved as “validation\_data.csv.” The test set is saved as “test\_data.csv.”

**E2. Create an initial model using the training dataset and provide a screenshot of the following metrics**

A screenshot of a computer program

Description automatically generated

In this code, I started by importing the RandomForestClassifier() function from the sklearn library, as this function is used to create and train the Random Forest model. I also imported functions like accuracy\_score(), precision\_score(), recall\_score(), etc., as these functions are used to evaluate the model’s performance.

I initialized the Random Forest model by using the RandomForestClassifier() function. Next, I fit the model to the training set. Then, I used the predict() function for the model to predict the class labels for the validation set. I also used the predict\_proba() function for the model to predict probabilities for each class. This is used to calculate the AUC-ROC metric.

I calculated and displayed the accuracy, precision, recall, F1 score, AUC-ROC, and confusion matrix for the Random Forest model. The model has an accuracy score of 0.98, precision score of 0.97, recall score of 0.97, F1 score of 0.97, and an AUC-ROC of 1.00. The model’s confusion matrix shows 1435 true negatives, 26 false positives, 26 false negatives, and 913 true positives.

**E3. Perform hyperparameter tuning on the validation dataset using k-fold cross validation to find the optimized model**

A screenshot of a computer program

Description automatically generated

The hyperparameters that I selected for tuning are ‘n\_estimators’, ‘max\_depth’, ‘min\_samples\_split’, and’ min\_samples\_leaf’. The hyperparameter ‘n\_estimators’ represents the number of trees in the forest. The hyperparameter ‘max\_depth’ controls the maximum depth of each tree. The hyperparameter ‘min\_samples\_split’ is the minimum number of samples required to split a node. The hyperparameter ‘min\_samples\_leaf’ is the minimum number of samples required in a leaf node.

Here, I will discuss the justification of the selected hyperparameters. I selected ‘n\_estimators’ because it controls the number of trees in the forest, with more trees typically improving performance but increasing training time. I selected ‘max\_depth’ because it limits tree depth, helping to prevent overfitting by avoiding overly complex trees. I selected ‘min\_samples\_split’ because it reduces overfitting by requiring more samples to split a node. I selected ‘min\_samples\_leaf’ because it reduces overfitting by ensuring leaf nodes have sufficient samples.

A screenshot of a computer code

Description automatically generated

The optimized Random Forest model’s best hyperparameters are max\_depth=10, min\_samples\_leaf=4, min\_samples\_split=10, and n\_estimators=50. This combination of hyperparameters achieved an average accuracy score of 97.96% during cross-validation.

A screenshot of a computer program

Description automatically generated

I evaluated the optimized Random Forest model on the validation dataset. The optimized model has an accuracy score of 0.98, precision score of 0.97, recall score of 0.97, F1 score of 0.97, and an AUC-ROC of 1.00. The model’s confusion matrix shows 1437 true negatives, 24 false positives, 25 false negatives, and 914 true positives.

**E4. Use the optimized model identified in part E3 to make predictions using the test dataset and provide a screenshot of the following metrics**

A screenshot of a computer program

Description automatically generated

I evaluated the optimized Random Forest model on the test dataset. The optimized model has an accuracy score of 0.98, precision score of 0.98, recall score of 0.98, F1 score of 0.98, and an AUC-ROC of 1.00. The model’s confusion matrix shows 1250 true negatives, 15 false positives, 16 false negatives, and 719 true positives.

A screenshot of a computer program

Description automatically generated

I extracted and sorted the feature importance values from the optimized Random Forest model, ranking features by their contribution to predicting patient readmissions within a month. The most significant predictor of readmission is ‘Initial\_days’ with an importance value of 0.5439. This suggests that longer stays may indicate severe health issues, increasing the likelihood of readmission. The second most significant predictor is ‘TotalCharge’ with an importance value of 0.4190. This suggests that higher charges may indicate intensive treatments or complex conditions, which correlate with higher readmission risks. Medical conditions like arthritis, high blood pressure, stroke, and diabetes have very low importance values, indicating they contribute minimally to the model’s predictions.

**F1. Compare and discuss the metrics of accuracy, precision, recall, F1 score, and AUC-ROC from the use of the optimized model on the test dataset and the initial model on the training dataset to evaluate the performance of the optimized model**

Here we will compare and discuss the metrics between the initial Random Forest model that was evaluated on the validation dataset and the optimized Random Forest model that was evaluated on the test dataset. Both initial and optimized models achieved an accuracy of 98%, which means both models are very effective at correctly classifying patient readmissions. The optimized model’s performance on the test dataset confirms that the model generalizes well to new data. Precision improved slightly in the optimized model, increasing from 97% to 98%. This indicates that the optimized model is slightly better at reducing false positives, making it more accurate in identifying patients likely to be readmitted. Recall also improved slightly in the optimized model, increasing from 97% to 98%. This indicates that the optimized model is slightly better at identifying true positives, making it more effective at detecting patients who will actually be readmitted. The F1 score improved from 0.97 in the initial model to 0.98 in the optimized model. This improvement shows a better balance between precision and recall in the optimized model. Both models achieved an AUC-ROC of 1.00, indicating that they are excellent at distinguishing between patients who will and will not be readmitted. This consistency shows the optimized model maintained the excellent class separation of the initial model.

Overall, the optimized Random Forest model showed slight improvements in precision, recall, and F1 score compared to the initial model. These improvements demonstrate more accurate identification of true positives with fewer false positives. These improvements also suggest that hyperparameter tuning and cross-validation successfully refined the Random Forest model, making it more reliable for predicting patient readmissions on unseen data.

**F2. Discuss the results and implications of your classification analysis**

The Random Forest classification analysis on the medical dataset produced strong results, as highlighted by the key metrics. The optimized model achieved an accuracy score of 98%, indicating excellent performance in correctly classifying patient readmissions. The model achieved a precision score of 98%, showing excellent performance in reducing false positives and reliably identifying patients likely to be readmitted. The model achieved a recall score of 98%, indicating that the model is very effective at identifying patients who will actually be readmitted. The model achieved an F1 score of 98%, showing a near-perfect balance between precision and recall, making it reliable for predicting readmissions. The model achieved an AUC-ROC of 1.00, demonstrating excellent ability in distinguishing between patients who will and will not be readmitted.

Regarding the results of performing feature importance analysis in the optimized Random Forest model, there are two features that significantly influence patient readmission. The most important feature is ‘Initial\_days’ with an importance value of 0.5439. This suggests that longer stays could reflect serious health problems, raising the chances of readmission. The second most important feature is ‘TotalCharge’ with an importance value of 0.4190. This suggests that higher charges might reflect intensive treatments or complex conditions, which are associated with increased readmission risks. Medical condition features like arthritis, high blood pressure, stroke, and diabetes have very low importance values, showing they have little impact on the model’s predictions.

The analysis has several important implications. The first implication is that the optimized model can accurately identify patients at risk of readmission, since it has high precision and recall. This allows hospitals to focus resources on high-risk patients, potentially reducing readmission rates and improving patient outcomes. The second implication is that the optimized model reduces false positives, minimizing unnecessary interventions for low-risk patients. The third implication is that hospitals can use this model to predict readmissions proactively, helping reduce penalties from organizations like CMS. The fourth implication is hospitals can prioritize interventions for patients with long hospital stays and high treatment costs, as these factors often indicate higher readmission risks. The fifth implication is hospitals can simplify their predictive models by focusing on key features like ‘Initial\_days’ and ‘TotalCharge’, reducing complexity while maintaining accuracy.

**F3. Discuss one limitation of your data analysis**

One limitation of the Random Forest data analysis is it lacks straightforward interpretability compared to simpler models like logistic regression. While feature importance provides some interpretability, it does not detail the relationships between predictors and the target variable, such as how ‘Initial\_days’ specifically influences readmission risk. This makes it harder for healthcare professionals to understand or trust the model’s predictions, which is critical in healthcare decision-making.

**F4. Recommend a course of action for the real-world organizational situation from part B1 based on your results and implications discussed in part F2**

One recommended course of action to address the question “What factors are the most predictive of patient readmission within a month of release?” is to focus on high-risk predictors such as ‘Initial\_days’ and ‘TotalCharge.’ Hospitals should improve discharge planning for patients with longer hospital stays, as they face a higher risk of readmission. Hospitals should provide additional follow-up care, such as home visits, telehealth consultations or regular check-ins for these patients. Patients with higher hospital charges often have more complex medical conditions. Hospitals should create personalized post-discharge care plans for these patients to address potential complications early. Overall, hospitals should prioritize resources for patients with the highest risk scores and reduce unnecessary interventions for low-risk patients to save time and costs.

**G. Panopto Video Recording**

Here is the link to my Panopto video recording: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=00a8caac-64cf-4544-9387-b26e001a4503>

**H. Record the web sources used to acquire data or segments of third-party code to support the analysis**

Firdose, T. (2023, August 24). Fine-tuning your random forest classifier: A guide to hyperparameter tuning. Medium. <https://tahera-firdose.medium.com/fine-tuning-your-random-forest-classifier-a-guide-to-hyperparameter-tuning-d5ceab0c4852>

**I. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized**

Sruthi. (2024, December 11). Understanding random forest algorithm with examples. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/