***Classification Data Mining Models***

*WGU*

*Course Number: 604*

*Trayvonious Pendleton*

*011205284*

**B1: Propose a Real-World Question for Gradient Boost:**

This analysis aims to build a model that can accurately predict whether a patient will be readmitted within 30 days based on their medical conditions and personal information.

**B2: Define One Goal of the Data Analysis:**

This data analysis aims to develop a predictive model using gradient boost to identify whether a patient will likely be readmitted to the hospital within 30 days after discharge. This model will help the hospital proactively manage high-risk patients and reduce readmission rates.

**C1: Explain How the Classification Method Analyzes the Dataset:**

Gradient Boost is a powerful ensemble machine learning technique that builds a series of decision trees, where each tree tries to correct the errors made by the previous ones. For this dataset, Gradient Boost will analyze patient demographic information, medical conditions, and hospitalization details to learn patterns associated with readmission.

The model works by minimizing prediction error through boosting, which adjusts weights on incorrectly predicted samples and focuses learning on the most challenging cases. This results in high predictive accuracy and helps the model capture complex relationships between features.

The expected outcome is a binary classification — predicting “yes” or “no” for readmission within 30 days — with strong performance on metrics like accuracy, precision, recall, and AUC-ROC, which are essential in healthcare decision-making.

**C2. List of Python Packages and Justifications**

1. Pandas:

Used for data loading, exploration, and manipulation. It helps with reading the CSV file, handling missing values, encoding categorical variables, and organizing the data into DataFrames for analysis.

1. Numpy:

Supports numerical operations and efficient handling of arrays and mathematical functions, which are often used during preprocessing and model evaluation.

1. Scikit-learn:

Used for splitting the dataset (train/test), preprocessing (e.g., encoding and scaling), evaluating model performance (accuracy, precision, recall, F1 score, AUC-ROC), and hyperparameter tuning with GridSearchCV and cross-validation.

1. Xgboost:

This is the main package used to implement the Gradient Boost algorithm. It provides a highly efficient and optimized implementation for classification tasks.

1. Matplotlib/seaborn:

Used for visualizing data distributions, feature importance, and confusion matrices to better interpret model behavior and results.

**D1: Data Preprocessing Goal:**

Preprocessing aims to clean and prepare the dataset for training the Gradient Boost model by handling missing values, encoding categorical variables, and ensuring that all data types are compatible with the model input.

**D2: Initial Variables and Classification:**

|  |  |
| --- | --- |
| Variable | Type |
| ReAdmis | Categorical |
| Age | Continuous |
| Gender | Categorical |
| Income | Continuous |
| HighBlood | Categorical |
| Diabetes | Categorical |
| Initial\_days | Continuous |
| TotalCharge | Continuous |
| Initial\_admin | Categorical |
| Complication\_risk | Categorical |

**D3. Data Preparation Steps and Code Snippets:**

**Step 1 Load the Dataset:**

import pandas as pd

df = pd.read\_csv(‘medical\_clean.csv’)

**Step 2 Keep Only Selected Variables:**

df = df[['ReAdmis', 'Age', 'Gender', 'Income', 'HighBlood', 'Diabetes',

'Initial\_days', 'TotalCharge', 'Initial\_admin', 'Complication\_risk']]

**Step 3: Encode the target variable:**

df['ReAdmis'] = df['ReAdmis'].map({'Yes': 1, 'No': 0})

**Step 4: Encode categorical variables:**

categorical\_cols = df.select\_dtypes(include='object').columns

df\_encoded = pd.get\_dummies(df, columns=categorical\_cols, drop\_first=False)

**Step 5: Handle missing values**

df\_encoded = df\_encoded.dropna()

**Step 6: Split features and target**

X = df\_encoded.drop('ReAdmis', axis=1)

y = df\_encoded['ReAdmis']

**Step 7 Save Cleaned Dataset**

df\_encoded.to\_csv('cleaned\_medical\_data.csv', index=False)

**E1: Splitting the Data**

To evaluate model performance effectively and ensure that the model generalizes well to new data, the dataset was split into three subsets: training, validation, and testing. I used the train\_test\_split method from scikit-learn to perform this split in two stages:

* First, I split 70% of the data into the training set used to train the model.
* The remaining 30% was split evenly into a validation set (15%) and a test set (15%).

This structure allows for effective model training while reserving separate data for tuning hyperparameters and final evaluation. The validation set was used during model tuning with cross-validation to avoid overfitting, and the test set was only used at the end to evaluate the final model’s performance.

**E2: Initial Model and Metrics**

**A screenshot of a computer program

AI-generated content may be incorrect.**

To establish a baseline for model performance, I trained an initial classification model using the XGBoostClassifier, a robust gradient-boosting algorithm known for its accuracy and efficiency with structured data. The model was trained on the training dataset and evaluated using the test dataset. I measured the model’s performance using five key classification metrics: accuracy, precision, recall, F1 score, and AUC-ROC.

The initial model achieved strong results:

* **Accuracy** of 0.9773,
* **Precision** of 0.9710,
* **Recall** of 0.9675,
* **F1 Score** of 0.9692, and
* **AUC-ROC** of 0.9984.

The confusion matrix showed that the model made few false predictions, demonstrating high sensitivity and specificity. These results indicate that the initial model already had strong predictive power, making it a solid foundation for further optimization.

**E3. Hyperparameter Tuning Using k-Fold Cross Validation**

A screenshot of a computer program

AI-generated content may be incorrect.

Selected Hyperparameters: The following hyperparameters were selected for tuning in the Gradient Boost(XGBoost) model:

* max depth: [3,5,7]
* learning rate:[0.001,0.1,0.2]
* n\_estimators:[100,200]
* subsample:[0.8,1.0]

Justification for Each Hyperparameter:

* max depth: Controls the depth of each decision tree. Shallow trees prevent overfitting, while deeper trees allow more complex relationships. Tuning this helps balance bias and variance.
* learning rate: Determines how quickly the model adapts. Lower values lead to slower but more accurate learning. Finding the right value that allows strong learning without overshooting optimal solutions is crucial.
* n\_estimators: Sets the number of boosting rounds (trees). More trees can improve performance but also risk overfitting. Tuning this helps find the sweet spot between underfitting and overfitting.
* Subsample: Refers to the proportion of the training data used to grow each tree. It helps reduce overfitting and introduces model robustness through randomness.

**E4: Optimized Model**

**A screenshot of a computer program

AI-generated content may be incorrect.**

After completing hyperparameter tuning using GridSearchCV, I evaluated the performance of the optimized gradient boosting model using the test dataset, which the model had not seen during training or validation. This final evaluation provides an unbiased estimate of the model's performance in a real-world deployment.

I calculated key classification metrics:

* **Accuracy**: 0.976
* **Precision**: 0.9887
* **Recall**: 0.9458
* **F1 Score**: 0.9667
* **AUC-ROC**: 0.9967

The optimized model achieved excellent precision, indicating it effectively reduced false positives. While the recall was slightly lower than the initial model, it remained strong, and the F1 score balanced both metrics well. The AUC-ROC score of 0.9967 reflects the model’s excellent ability to distinguish between patients who would and would not be readmitted. These results confirm the success of the hyperparameter tuning process in improving model performance and reliability.

**F1: Performance Comparison: Initial vs Optimized Gradient Boost Model**

To evaluate the performance improvement gained through hyperparameter tuning, the following metrics were compared between the initial and optimized Gradient Boost models:

|  |  |  |
| --- | --- | --- |
| Metric | Initial Model | Optimized Model |
| Accuracy | 0.9773 | 0.9760 |
| Precision | 0.9710 | 0.9887 |
| Recall | 0.9675 | 0.9458 |
| F1 Score | 0.9692 | 0.9667 |
| AUC-ROC | 0.9984 | 0.9967 |

**Confusion Matrices**

**Initial Model:** [[931, 16], [18, 535]]

**Optimized Model:** [[941, 6], [30, 523]]

**Discussion:**

**Accuracy:** Both models achieved high accuracy, but the initial model scored slightly higher. This suggests that the initial model correctly classified a marginally greater number of overall instances. However, because accuracy alone can be misleading, it should be interpreted alongside precision, recall, and AUC-ROC.

**Precision:** The optimized model demonstrated a substantial improvement in precision. This means it was significantly better at avoiding false positives, which is a crucial advantage in healthcare, where incorrectly predicting a patient will be readmitted could lead to unnecessary follow-up procedures or wasted resources.

**Recall**: The initial model had a higher recall, indicating it was better at identifying actual readmission cases. This can be beneficial when the goal is to avoid missing patients who are likely to be readmitted. The optimized model sacrificed some recall to improve precision.

**F1 Score**: Both models performed similarly on the F1 score, which balances precision and recall. The slightly higher F1 score in the initial model reflects its better balance of identifying true positives while minimizing false positives and false negatives. However, the difference is slight, suggesting both models are robust.

**AUC-ROC**: Both models achieved near-perfect AUC-ROC scores, showing excellent ability to distinguish between readmission and non-readmission cases. The slight drop in the optimized model’s AUC-ROC is minor and does not indicate any significant degradation in model performance.

**F2: Discussion of Results and Implications**

The results demonstrate that the initial and optimized gradient boosting models performed exceptionally well, with accuracy, F1 scores, and AUC-ROC values exceeding 0.96. The optimized model slightly improved precision at the cost of a slight drop in recall and accuracy, suggesting it was more conservative in identifying positive cases (readmissions) but more confident when it did.

These findings imply that the model can effectively flag patients at risk of hospital readmission, enabling medical staff to intervene early. The high AUC-ROC score reflects the model’s strong ability to differentiate between patients who will and will not be readmitted. This capability could improve resource allocation, reduce unnecessary hospital stays, and support patient care planning.

**F3: Limitation of the Data Analysis**

One key limitation of this analysis is the limited number of variables used in the final model. While the variables selected (e.g., age, income, initial admission type, etc.) were relevant and based on Part D2, the exclusion of potentially essential features — such as specific diagnoses, medical history, or provider-level data — may limit the model’s ability to capture more complex patterns associated with readmission risk.

Additionally, the dataset did not provide information about the time between discharge and readmission, which could have improved prediction accuracy.

**F4: Recommended Course of Action**

Based on the classification results and implications, the recommended course of action is for the healthcare organization to implement the optimized gradient boosting model into its patient monitoring system. The model can be a decision support tool that flags high-risk patients for additional post-discharge care, follow-up appointments, or home health services.

Given the model’s high precision and solid recall, it would reduce false alarms while still catching most cases of true readmissions. This action aligns with the organization’s goal from Part B1 — reducing preventable readmissions and improving patient outcomes — while supporting the efficient use of staff and resources.