CISC 451 – Assignment 2

Using Supervised Learning to Predict Diabetes Patient Readmission

Due: October 28th, 2020

Gavin McClelland – 10211444
Marshall Cunningham - 20249991

1.0 Introduction

The provided dataset was representative of ten years of clinical care from years 1999-2008 at 130 hospitals in the United States. These data were presented as a collection of 50 features describing patients and corresponding hospital outcomes. The objective at hand is to develop a predictive model(s) that can predict if a patient will be readmitted to the hospital again or not. An attempt was made to extend this approach to support the additionally specified classification problem concerning whether a patient will be readmitted within 30 days of discharge (multi-class classification). This paper details the methodology applied to construct different models to address this classification problem, the accuracy achieved using different approaches against a test dataset, and the software required to replicate the results obtained.

2.0 Software Packages

All code written for this assignment was written in the Python programming language using Jupyter notebooks. The main software packages used in this assignment were, pandas, scikit-learn, and numpy. A tree structure of the submitted directory is shown below.

```
|    CISC 451 – Assignment 2 Report.pdf
|    requirements.txt
+---code
|    patient_readmission.ipynb
|    patient_readmission_30days.ipynb
|    helper functions: plot_roc_curve.py, validate_model.py, validate_model_multiclass.py
\---data
     C2T1_Train.csv, C2T1_Test.csv, C2T1_Test_Labeled.csv, C2T1_Test_Labeled_30days.csv
```

3.0 Instructions

Instructions to replicate all work completed as submitted are listed below:

1. Install all dependencies by running the command “pip install -r requirements.txt” in the root directory of the submitted folder.
2. Before running either notebook in the code folder, add your working directory at the top where indicated (i.e. %cd “<your directory here>”). Then, run all cells.

4.0 Methodology and Analytics Process

The analytics and modeling methodology consisted of an EDA including sufficient data processing, feature selection and feature engineering, basic modeling using several ML models, hyperparameter tuning and model testing.

4.1 Preprocessing and Data Preparation

The training dataset consists of 90766 patient encounters, each containing 50 features representing patient and hospital outcomes.

4.1.2 Feature Selection

Of these 50 features, four features were dropped for the following reasons:

- Encounter ID: This is a unique identifier of each encounter and thus carries no predictive value
- Patient Number: This is a unique identifier of each patient, may carry slight predictive value but will most likely lead to overfitting.
- Payer Code: Many unique values and many missing values.

- Secondary and Tertiary Diagnoses: The primary diagnosis (diag_1) was the only diagnosis considered due to the large number of unique values in all three features.

The remaining numerical features were kept with no further processing while the categorical features were engineered for better predictive results.

4.1.2 Feature Engineering

The following categorical features were binned to reduce the number of unique values and combine analogous values: Age, Weight, Admission Type, Admission source, Discharge Disposition, Medical specialty, and Primary diagnosis. After these features were binned, all categorical features were converting into numerical features using one-hot encoding.

4.1.3 Down Sampling

To achieve the best results from the predictive model, the training data was down sampled to obtain an equal number of patients who were readmitted to those who were not. After all the preprocessing, the final training dataset consisted of 82,810 rows each with 123 features.

4.2 Modeling

Several ML models were trained and validated using 10-fold cross-validation on the processed training dataset. The gradient boosting classifier resulted in the highest prediction accuracy and therefore it was chosen for hyperparameter tuning. Tuning methodology consisted of selecting baseline parameter values, then using GridSearchCV on each parameter to find the optimum parameter value. After the hyperparameters were found, the learning rate was decreased as the number of estimators was increased, this resulted in a slightly more accurate model. The confusion matrices and the ROC plots for the untuned, and tuned models are shown below in Figure 1.

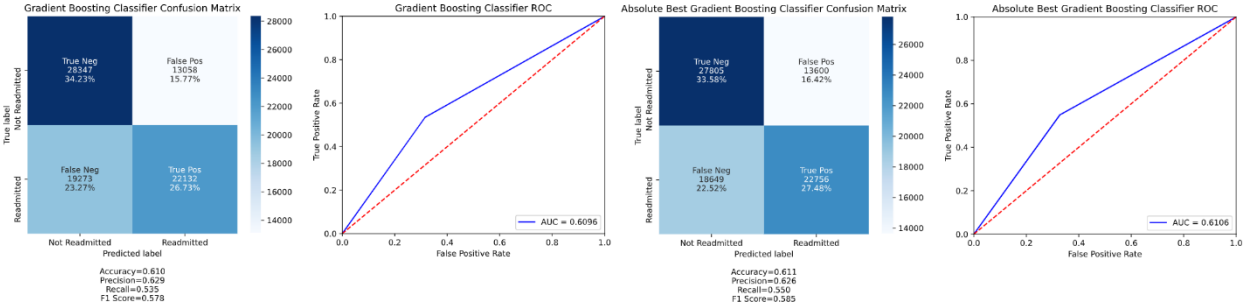

Figure 1 – Results expressed as confusion matrices and ROC curves

4.3 Other Approaches Explored

The following classification models (and their respective accuracies) were trained using the default parameter values in Scikit-learn

- Logistic Regression: Accuracy – 0.601, ROC AUC – 0.6008
- K Nearest Neighbors: Accuracy – 0.541, ROC AUC – 0.5413
- Decision Tree: Accuracy – 0.543, ROC AUC – 0.5427
- Random Forest: Accuracy – 0.593, ROC AUC – 0.5926

5.0 Model Evaluation

Models were evaluated based on their ROC AUC to measure how the models distinguish between classes. The gradient boosting classifier was used to make predictions on the test dataset, shown in the file ‘C2T1_Test_Labeled.csv’ included with the submission.