Gaby Masak

D603 – Machine Learning

Task 1: Classification Data Mining Models

11/17/2024

Explanation: Code for Data Production Pipeline

**Requirement A: Gitlab Subgroup and Project**

GitLab URL: <https://gitlab.com/wgu-gitlab-environment/student-repos/gmasak/d603-machine-learning/-/tree/working_branch?ref_type=heads>

Screenshot of Repository Branch History:

**A screenshot of a computer

Description automatically generated**

Figure 1: Screenshot of Repository Branch History

**Requirement B: Purpose of Data Mining Report**

Hospitals are frequently penalized and fined by the Centers for Medicare and Medicaid Services (CMS) for excessive readmission. By collecting and leveraging larges sets of data on patient demographics, health, and treatment, it is possible to apply gradient boosting classification methods to answer questions surrounding predicting readmission before it occurs. With this analysis, hospitals can formulate better practices and potential solutions with the ultimate goal of reducing future hospital readmissions, avoiding fines from CMS, and improving patient outcomes. For example, if the hospital uses the model and predicts that a patient will be readmitted, the hospital can introduce an at-home or telehealth service to the patient, more effectively targeting at-risk patients and lowering the chances of readmission.

**Requirement C: Reasons for Classification Method**

Gradient boosting is an ensemble learning method employed to make predictions about the classifications of data from a number of different models. Models, comprised of decision trees, are trained sequentially, each learning from the pseudo-residuals of the previous model, improving the models until a limit is reached or no improvements can be made (Dash, 2020). Expected outcomes include improved prediction model accuracy compared to logistic classification methods, with less bias and variance in spite of complicated data sets.

Python packages:

1. **pandas**:

* **Purpose**: Data manipulation and analysis.
  + pandas provides data structures like dataframes for data handling and processing.

1. **scikit-learn**:

* **Purpose**: Machine learning and data mining.
  + scikit-learn offers a various tools, including GradientBoostingClassifier, for model building, evaluation, and hyperparameter tuning.

1. **matplotlib**:

* **Purpose**: Data visualization.
  + matplotlib is used to create visualizations such as confusion matrices, among other plots and graphs.

1. **seaborn**:

* **Purpose**: Statistical data visualization.
  + seaborn builds on matplotlib and provides an interface for drawing statistical graphics, such as heatmaps for confusion matrices.

1. **Numpy**:

* **Purpose**: Numerical computing.
  + numpy provides mathematical functions to operate on arrays.

**Requirement D: Data Preparation**

As discussed above, GradientBoostingClassifier in scikit-learn was essential in the development and implementation of gradient boosting models on the health data set. The scikit-learn class requires numeric input to perform splits and make predictions. This means that the categorical values in Table 1 below needed to be transformed by either one-hot encoding or label encoding. Due to the lack of ordinal relationships in the majority of categorical variables, I used hot-one encoding to transform the variables into numerous binary columns, with the originally binary categorical variables remaining as a single column to avoid redundancy.

Table 1: Variable Classification

|  |  |
| --- | --- |
| **Variable** | **Classification (Continuous/Categorical)** |
| Lat | Continuous |
| Lng | Continuous |
| Population | Continuous |
| Area | Categorical |
| Job | Categorical |
| Children | Continuous |
| Age | Continuous |
| Income | Continuous |
| Marital | Categorical |
| Gender | Categorical |
| ReAdmis | Categorical |
| VitD\_levels | Continuous |
| Doc\_visits | Continuous |
| Full\_meals\_eatan | Continuous |
| VitD\_supp | Categorical |
| Soft\_drink | Continuous |
| Initial\_admin | Categorical |
| HighBlood | Categorical |
| Stroke | Categorical |
| Complication\_risk | Categorical |
| Overweight | Categorical |
| Arthritis | Categorical |
| Diabetes | Categorical |
| Hyperlipidemia | Categorical |
| BackPain | Categorical |
| Anxiety | Categorical |
| Allergic\_rhinitis | Categorical |
| Reflux\_esophagitis | Categorical |
| Asthma | Categorical |
| Services | Categorical |
| Initial\_days | Continuous |
| TotalCharges | Continuous |
| Additional\_charges | Continuous |
| Item1 | Continuous |
| Item2 | Continuous |
| Item3 | Continuous |
| Item4 | Continuous |
| Item5 | Continuous |
| Item6 | Continuous |
| Item7 | Continuous |
| Item8 | Continuous |

Several steps were taken to prepare the data for analysis. First, any duplicate rows are removed, identified by the code segment in Figure 2 below.



Figure 2: Screenshot of Code Removing Duplicate Rows

Next, variable selection reduced the 49 independent variables into 39 by removing redundant variables, such as geographical or identification variables in line 26. Of the geographical variables, only longitude and latitude remained as their calculation is objective and free from political bias and historical subjectivity.



Figure 3: Screenshot of Code Removing Redundant Variables

Subsequently, any missing values are filled. For missing values in categorical columns, the mode is used to replace the null entries. For missing values in numerical columns, the mean of the column is used, as seen in Figure 4 below.

A screen shot of a computer code

Description automatically generated

Figure 4: Screenshot of Code Replacing Missing Values

Next, the data was prepared for one-hot encoding. This required for categorical columns to be identified and split into binary and polyadic groups, with polyadic groups splitting into several columns after undergoing one-hot encoding, as seen in Figure 5. After one-hot encoding, the data was transformed back into a dataset. A copy of the cleaned dataset can be found in the attached cleaned\_data.csv file.

A computer screen with many colorful text

Description automatically generated

Figure 5: Screenshot of One-Hot Encoding Code and Preparation

**Requirement E: Data Analysis**

After the data is preprocessed, it is split into training, validation, and test datasets. These datasets are exported to the eponymous CSV files:

X\_train.csv, X\_val.csv, X\_test.csv, y\_train.csv, y\_val.csv, y\_test.csv

The training set is used to create an initial model. Metrics regarding accuracy, precision, recall, F-1 score, and AUC-ROC are calculated and displayed in Figure 6. A confusion matrix is displayed in Figure 7.

A screenshot of a computer screen

Description automatically generated

Figure 6: Screenshot if Initial Model Metrics

A blue squares with numbers and a graph

Description automatically generated

Figure 7: Confusion Matrix from Initial Model

Hyperparameter tuning was performed on the validation dataset using k-fold cross validation to find the optimized model. Four hyper parameters were selected for tuning: the number of boosting stages to be run (n\_estimators), the maximum depth of individual decision trees (max\_depth), the step size at each iteration (learning\_rate), and the fraction of samples used for fitting individual base learners (subsample).

A computer screen shot of numbers and symbols

Description automatically generated

Figure 8: Screenshot of Hyperparameters

These hyperparameters were selected for various reasons. The value of n\_estimators can improve the model if increased, but doing so risks overfitting. Similarly, the values of max\_depth and subsample can respectively capture complex patterns and improve generalization, but values too high can also lead to overfitting. With learning\_rate, the value determines the contribution in reduction of error rate of each tree, meaning that too low of a value will require the generation of more trees, linking its optimal value to that of n\_estimators. The optimal values are displayed in Figure 9.



Figure 9: Screenshot of Best Hyperparameters

The metrics produced by the optimized model o make predictions using the test dataset are displayed in Figure 10 below with the confusion matrix in Figure 11.

A screen shot of a computer

Description automatically generated

Figure 10: Screenshot of Metrics from Optimized Model

A blue squares with white text

Description automatically generated

Figure 11: Confusion Matrix of Optimized Model

**Requirement F: Summary**

Analyzing the metrics of accuracy, precision, recall, F1 score, and AUC-ROC from the use of the optimized model on the test dataset (Figure 10) and the initial model on the training dataset (Figure 6) to evaluate the performance of the optimized model demonstrates that the optimized model performs exceptionally well across all metrics. The high accuracy, precision, recall, F1 score, and AUC-ROC indicate that the model is highly effective in predicting hospital readmissions. The metrics of both models were extremely high, only slightly lower than the maximum value of 1 across all measured metrics. The slight improvements in AUC-ROC for the optimized model suggest that it is better at correctly identifying true positives and distinguishing between classes.

As a result, the hospital can rely on the optimized model to accurately predict readmissions with confidence, allowing for targeted interventions before readmission occurs. In turn, this can help the hospital focus on high-risk patients and more strategically implement preventive measures. This approach will save time and effort, as hospital staff can more efficiently focus their talents on more impactful avenues. This efficiency is also cost effective due to the reduction of fines, material costs, and labor. A beneficial recommendation would be to use the data to employ at-home educational, medical, and support programs for the identified high-readmission risk patients, thereby effectively reducing the chances of readmission and associated fines.

While the benefits are intriguing, the limitations of the data analysis should not be ignored. One limitation of the data analysis is the potential for overfitting, especially given the high performance metrics. While k-fold cross validation helps mitigate this risk, the model may still perform differently on entirely new datasets. Additionally, the analysis relies on the quality and completeness of the data. Any biases or inaccuracies in the data can affect the model's performance and the validity of the results.

**Requirement G: Panopto Video Link**

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=ad128b27-7c05-4554-82b4-b232014500e5>

**Sources Cited**

**Code:**

* + 1. GeeksforGeeks. (2020, August 25). *Gradient Boosting in ML*. GeeksforGeeks.

https://www.geeksforgeeks.org/ml-gradient-boosting/#

* + 1. *Gradient Boosting Classification Example with Scikit-learn*. (2019, February 27).

Datatechnotes.com. https://www.datatechnotes.com/2019/02/gradient-boosting-

classification.html

**Report:**

* + 1. Dash, S. (2020, October 21). *Gradient Boosting - A Concise Introduction from*

*Scratch*. Machine Learning Plus. https://www.machinelearningplus.com/machine-

learning/gradient-boosting/