CSCE421-501 Final Project

Lam Nguyen, Joseph Quismorio
Department of Computer Science
Texas A&M University
College Station, TX 77840
`lam65nguyen@tamu.edu`
`josephquismorio@tamu.edu`

Abstract

Predicting mortality for patients is an important area of healthcare where machine learning can be utilized to help healthcare providers make more informed decisions about patient care. This project aims to build a machine learning model that can accurately predict the likelihood of mortality for patients based on various features such as age, gender, admission weight, etc. The model will be trained on a large dataset of patient records and evaluated using several performance metrics such as ROC-AUC score. The findings of this study have the potential to help healthcare providers identify patients who are at high risk of mortality, allowing them to provide more targeted care and interventions to improve patient outcomes.

1 Introduction

The expansion of machine learning has revolutionized many industries, including healthcare. By utilizing predictive modeling, machine learning is now able to help medicial professionals more easily identify patients who are at risk for certain diseases or conditions. One of the most important areas of healthcare where this technology can be applied is predicting mortality. Mortality prediction models can be used to identify patients who are at high risk of dying, which can help healthcare providers make more informed decisions about their care. This project aims to use a dataset provided by Phillips eICU program to understand how to work with large unorganized datasets to make inferences about them and clean them up, and then utilize that cleaned data to build and tune an appropriate machine learning model that can accurately predict mortality for patients based on a range of features defined by us.

2 Your Method

2.1 Data preprocessing

Our dataset given is composed of input data with a mix of categorical and continuous features such as admissionheight, gender, patientunitstayid, etc. and label data that will contain information about the hospital status (1 being dead and 0 and alive) of a given patientunitstayid. Before we can actually use any machine learning model, we needed to preprocess the data and select features as the input data is about 700k rows but there is only about 2k unique patients provided; so we essentially need to figure out a way to group all the data in the input together for a patientunitstayid down into one row instead of many scattered all across the dataset for a particular patientunitstayid.

The first step we chose to do is iterative add features by starting with the simple categorical and continuous features such as age, admissionweight, gender (encoded to be 1 for female and 0 for male). From those selected features, we use the groupby command to group rows by patientunitstayid so we can collapse 700k rows down to about 2k, and we found the sum of the numerical values; typically this is dangerous as we may not want to take the sum of all the values of a column for a particular patient we are grouping by but the features we selected only appear once since for example age is only recorded once in our dataset for example, so it is not a big deal for now. From there we were able to come up with a dataset with only about 2k rows, and used train_test_split with 20% going to testing to get data to train and validate our model on.

After we evaluated the model using ROC-AUC score for our simple preprocessed data, we looked at the remaining features and started analyzing everything that has to do with the nursing values such nursingchartcelltypevalname, nursingchartvalue, offset to see what we can extract. We noticed that there could be many types of nursingchartcelltypevalname, so what we decided to do to extract those values and still be comprehensive is find the min, max, first, and last values for a particular item such as Heart Rate, and create new columns in our input dataset to include those values for a particular patientunitstayid, and when we are done, just drop nursingchartcelltypevalname, nursingchartvalue, offset since all the useful information is extracted. We found that after doing this, our ROC-AUC score jumped significantly so we are definitely on the right track, and applied the same logic of extraction for columns such as labname and labresult.

In the end, we had 33 features used as seen below:

age, admissionweight, gender
Min, Max, First, Last Heart Rate
Min, Max, First, Last Respiratory Rate
Min, Max, First, Last Non-Invasive BP Systolic
Min, Max, First, Last Non-Invasive BP Mean
Min, Max, First, Last Non-Invasive BP Diastolic
Min, Max, First, Last Oxygen Saturation
Min, Max, First, Last Glucose
First, Last pH

Figure 1: Features Extracted

2.2 Model Design

To be able to successfully and accurately predict mortality rates among patients with our preprocessed data, we need to choose an appropriate classification model. That could be logisitic regression, decisions trees, XGBoost, random forest, or neural networks. All choices are fine technically but we wanted to focus on neural networks and XGBoost.

The reason why we chose to focus on looking at both and then deciding is because they both can achieve high accuracy and handle large datasets and missing values, which for the context of this project of medical records is pretty important as in the real world there will be millions of rows when dealing with medical records and many of those rows may have missing values.

The neural network will initially have two hidden layers with 100 and 50 neurons respectively, and use a ReLU activation function to be able to *wake* up neurons. In addition, the solver will use Adam to get an adaptive learning rate. The XGBoost will initially have 100 estimators, learning rate of 0.01, and max depth of around 3, so a simple starting model.

After running both untuned and viewing the ROC-AUC score, we believe that we should be using XGBoost instead of neural networks. The reason is not because neural networks yielded a low score, quite the opposite actually, but rather it is quite inconsistent in scoring doing multiple runs using the same model versus XGBoost yielded the same scores for multiple runs.

Furthermore, we also did not really do anything about missing values in preprocessing so we will be using SimpleImputer to impute missing values by replacing missing values (signified as NaN) with the mean of the available values. Then we used StandardScale to scale the values to be able to normalize our data since we have things like age with values 0-90 and offset with values 0-20000, and we want to have them in the same scale. Finally we can apply XGBoost via XGBClassifier. All of this will be put into a pipeline to streamline the process of developing the model.

2.3 Model Training

For training the model, we trained it based on the 33 features selected that have been scaled and free of missing values.

2.4 Hyperparameter tuning

Initially we did not plan to tune our hyperparameters since just using our simple one defined in 2.2 yielded us a ROC-AUC score that is above the professor's value, so we did not feel the need to tune the model. But for the sake of completeness, we decided to use GridSearchCV to do the tuning where we will try hyperparameters such as:

1. max_depth: [2, 4, 6, 8, 10]
2. learning_rate: [0.01, 0.1, 0.5, 1]
3. n_estimators: [50, 100, 200]
4. min_child_weight: [1, 5, 10]
5. gamma: [0, 0.1, 0.5]

The result was using XGBoost with a max depth of 8, n estimators to be 200, gamma of 0, minimum child weight of 5, and a learning rate of 0.5. We also tuned a neural network model that has 100 and 50 neurons for each of the two hidden layers, an adaptive learning rate to prevent it from being too low or high, go through 100 epochs during training, and alpha of 0.05.

3 Results

Results were surprisingly interesting as we incrementally added features. As stated before we initially started off with extracting age, admission weight, and gender and received 0.66706 for our ROC-AUC score when testing on the given training data and using a tuned model. Then when adding the nurse values, the ROC-AUC score jumped to 0.86565. Then finally a ROC-AUC score of 0.88121 when adding the lab results. When we submitted it to Kaggle, we received a ROC-AUC score of 0.87816 when untuned and after tuning, a score of 0.89723. In addition, we also submitted a tuned neural network and yielded inconsistent results hovering between 0.885 and 0.899, but XGBoost still remained at 0.89723. Quite a good improvement overall when tuned and the ROC curve can be seen below:

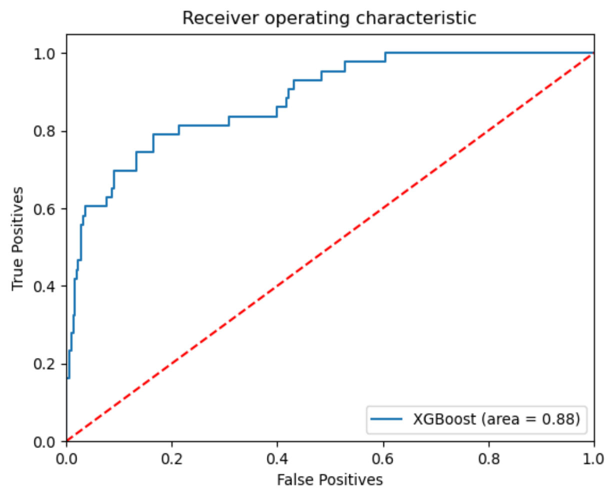

Figure 2: Plot of ROC curve

4 Conclusion

In conclusion, this project provided great insight not only into machine learning and getting a deeper understanding of the models discussed in class, but also how to perform data science to make inferences about raw data and making it organized with those inferences. The results analyzed for the ROC-AUC score are relatively high, and the model and hyperparameters selected were done well in a way that is able to accurately predict chance of mortality given the 33 features extracted for each unique patient. But more importantly, our feature selection was done relatively well as it also played a huge factor in our score since choosing maximum, minimum, first, and last values for a subfeature would be the most representative and less sensitive to outliers that could result from using something like mean.