CS 178: Final Project

CS 178 Final Project

Names	**Calvin Nguyen, Alex Meng, Will Huang**
Dataset	**Diabetes 130-US Hospitals**
Classification Methods	**kNN, Logistic Regression, Neural Network, Decision Tree**

June 10, 2024

Collaborators: Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore (“Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records”

Summary

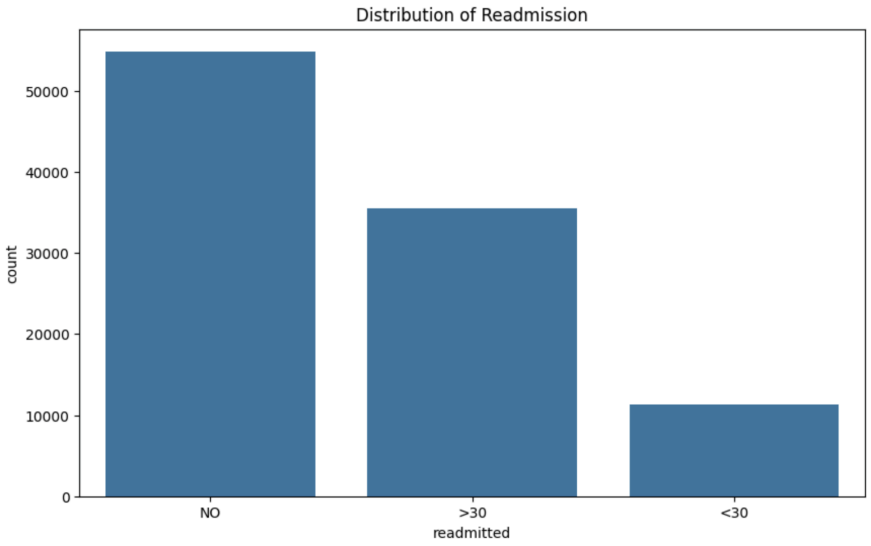
Our group investigated the classification problem for the Diabetes 130-US Hospitals dataset using K-nearest neighbors, logistic regression, neural network with one hidden layer, and decision trees. Through our experiments with each classifier, we were able to discover that even with tuned, optimal hyperparameters, each classifier had roughly the same accuracy score of 56%. Future work could involve including more data into the dataset, utilizing different classifiers, and fitting models on specific feature chunks.

Data Descriptions

The Diabetes 130-US Hospitals dataset consists of 47 integer and categorical features and a target column that classifies three labels. These labels include whether the patient was readmitted within 30 days of discharge, readmitted after 30 days of discharge, or not readmitted at all.

Our first step was to explore which columns had missing values, and so we generated summary statistics using Pandas and found that 9 of the 47 features contained missing values of which include race, weight, payer_code, medical_specialty, diag_1, diag_2, diag_3, A1Cresult, and max_glu_serum. Generating the columns that contained missing values allowed us to identify which features to evaluate for preprocessing and possible removal or modification.

We also wanted to explore possible causations for decreases in model performance and results by generating statistics and figures of the label distributions in order to see if overfitting may be a possible issue when testing our model against the testing sets.



Classifiers

The following models are the selected classifiers we chose for this dataset and problem:

- K-Nearest Neighbors
    - A classifier that operates on the principle that similar data points are likely to belong to the same class. It takes a new data point and classifies it based on the labels of the K closest data points.
    - Hyperparameters: 51 n-neighbors, weights = distance, algorithm = ball_tree, p = 1
- Logistic Regression
    - Usually a binary classification model, however, due to the dataset it was best to use a multinomial solver as there were three labels not two. The logistic regression model predicts the probability of being a specific class.
    - Hyperparameters: newton-cg solver, l2 penalty, C=1, fit_intercept=False, and a maximum iteration of 100.
- Neural Network
    - A model that closely mimics the human brain, using input, hidden, and output layers with nodes that pass data between the layers. For this classification experiment, we used one hidden layer.
    - Hyperparameters: 1 hidden layer with 64 nodes, learning rate of 0.001, batch size of 128, alpha value of 0.05
- Decision Trees
    - A classification model that takes the form of a binary tree. It splits the data into subsets based on the value of input features.
    - Hyperparameters: entropy, best, max_depth = 10, min_samples_split = 750, min_samples_leaf = 5, random_state = seed

Experimental Setup

The dataset consisted of missing values in several of the columns which would not allow us to train our model. To handle missing values, our team evaluated the columns that had missing values and measured its impact using the provided variables table in the dataset repository and additional data exploration. We concluded that most of the columns, such as payer_code and

medical_specialty, would not impact the results of our model and removed them from the dataset. Others such ass max_glu_serum had mostly empty or unavailable data; we used a threshold that if less than 5% of patients had data for some feature and that the feature held little to no relation to the output, we would conclude that it held little significance for our classification problem and removed it from the dataset. Only one feature was excluded from this threshold: we found that according to the research article "Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records", A1Cresult held high significance to which "readmission rates for patients with diabetes appeared to be associated with the decision to test for HbA1c, rather than the values of the HbA1c result." This analysis provided us with a causation to modify the data by the following: if a patient did not test for HbA1c, we would impute 0, else we would change the value to 1 regardless of the HbA1c result.

To partition the dataset, we first randomly selected 20% of the data to serve as our testing set. With the remaining 80%, we partitioned it further by performing a 75/25 train-test split into training and validation sets. Overall, we used 60% for training, 20% for validation, and 20% for testing. For consistency, we used the same testing set for all of our models so that we could directly compare their performances against each other.

In k-neighbors classification, our team first kept the all default hyperparameters except for n-neighbors. We then iterated through different values of n-neighbors ranging from 1 to 500 and plotted the error rates. This gave a general idea of how n-neighbors would affect classification accuracy. To see if we could obtain better results, we then opted to modify the hyperparameters of weights, algorithm, and p and ran a grid_search to find the best combination.

For the logistic regression classifier, our team opted to modify the hyper parameters of solver, penalty, C, max_iter, and fit_intercept. An exhaustive grid search was performed on these parameters to find the best combination of parameters to maximize accuracy. Our team also decided to focus on specific penalties, solvers, and combinations of the two as the documentation stated that certain solvers like liblinear and newton-cholesky were unable to work on multiclass classification. Working with multiclass classification solvers was extremely important as the diabetes dataset is not a typical binary target, and instead a multiclass target.

Since training a neural network model can be time consuming, the computation costs to perform a grid search and cross-fold validation would be significant. As a result, our team

decided to perform a randomized search to sample a fixed number of hyperparameter combinations rather than exhaustively searching and training models for every combination.

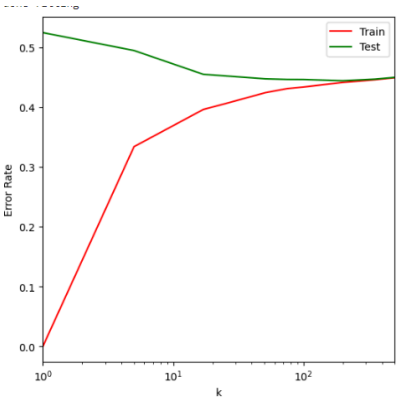
In our decision tree model, we performed a grid search on the following hyperparameters. Criterion, splitter, max_depth, min_samples_split, and min_samples_leaf. We chose these hyperparameters because we believed that they heavily influenced the calculation of a split and the structure of the tree. In turn, this would affect our model's ability to accurately classify data.

After training our models, we began testing and measuring the classification errors by measuring the difference between the true and predicted outputs using the testing sets.

Experimental Results

The following table shows the results of the k-neighbors classification. As seen, the validation error decreases by 5% when hyperparameters are tuned compared to the default. This is not surprising given that the default value for n-neighbors is 5. For a large dataset with over 100k rows and 40 features, we would expect better accuracy with a greater n-neighbors value.

K-Nearest Neighbors	Training Error	Validation Error
Default	33.4%	49.47%
Tuned Hyperparameters (Grid Search)	0.01%	44.02%



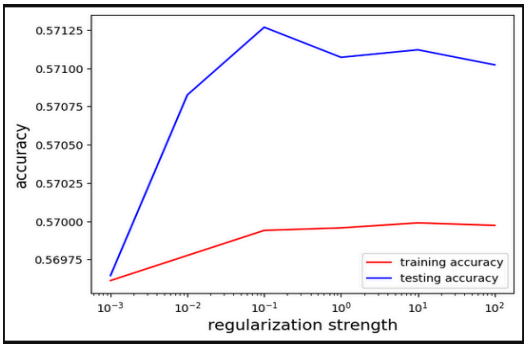
As previously mentioned, we first only adjusted the n_neighbors parameter and plotted the resulting error rates. Our minimum testing error was 44.42%. After using a grid search to see if adjusting other parameters would affect the accuracy, we discovered that the other hyperparameters only made the validation accuracy slightly better by 0.45%. However, the training error did decrease significantly. This leads us to conclude that n-neighbors is the only significant factor in using the kNN classifier in our dataset.

Logistic Regression	Training Error	Validate Error
Default	43.00%	42.94%

Tuned Hyperparameters (Grid Search)	42.96%	43.16%

Our untuned logistic regression classifier was trained on the 75% split of the original 80% split of the dataset. The classifier was used to predict on both the training set and validation set. The measured error rates between the two are practically identical.

Upon running a grid search, our hyper parameters were optimized. We were optimistic that our accuracy score would go up by at least 5-10%. However, the results from our optimized and tuned classifier shocked us. The classifier which was also fitted with a similar dataset as the untuned, and predicted with the 20% test set and same training set, had exactly the same results.



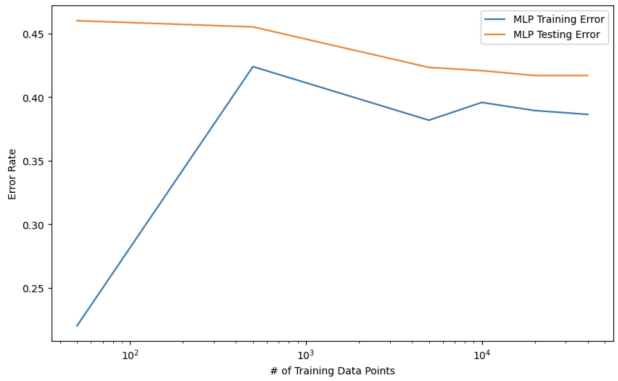
Based on the figure, past the regularization strength of 10^-1 or 10^0, the training accuracy levels off and sees little improvement. The testing accuracy has a somewhat similar response, but instead of leveling off the accuracy starts to decrease. Overall, the classifier results indicate that logistic regression may not be the best choice for prediction of patient readmission.

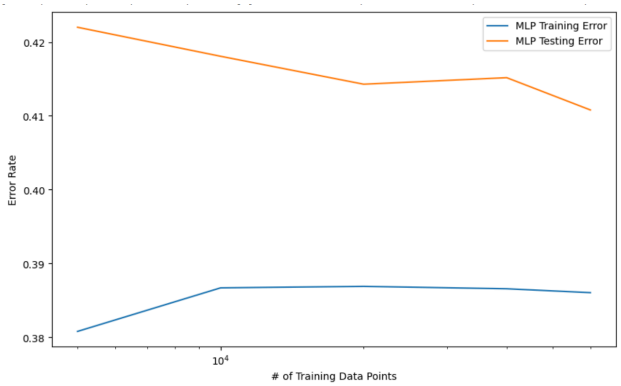
The following table shows our results using a neural network with one hidden layer:

Neural Network	Training Error	Validation Error
Default	0.01%	52.39%
Tuned Hyperparameters (Random Search)	39.12%	41.62%

We found that after tuning the hyperparameters, our validation error decreased by an estimated 10.77% compared to a default model. We tested this particular model with the final testing set, which resulted in a testing error of 42.76%.

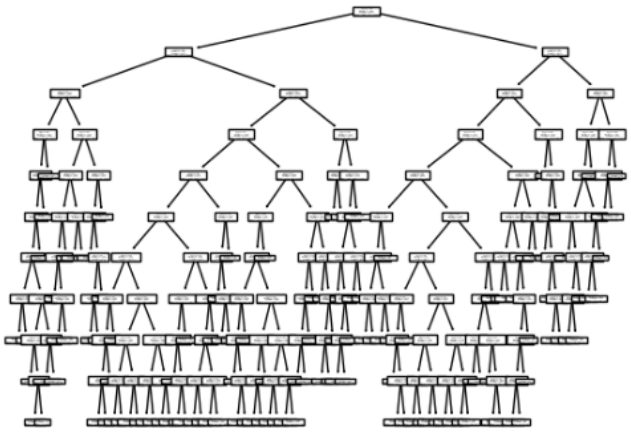
We also experimented with how our neural network's error rate would vary across different amounts of training data. The following graph shows that the error rate of the classifier decreases as the number of data points increases:





Decision Tree	Training Error	Validation Error
Default	0.01%	52.90%
Tuned Hyperparameters (Grid Search)	41.73%	41.71%

Our results in the table show that tuning hyperparameters for the decision tree classifier lowered validation error by over 10%. This demonstrates that the hyperparameters we chose to adjust indeed had a large influence on our dataset. Furthermore, we can see that the training error and validation error are almost identical, suggesting that the changes we made also reduced potential overfitting or underfitting. The image on the right is the tree obtained using the plot_tree method on our tuned decision tree.



Insights

Using different classification models and techniques allowed us to view how different applications of machine learning could

have an effect on the results. For example, since the dataset contained missing values, we had to investigate further in order to decide between imputing values or removing the feature altogether. Experimenting with different classifiers also taught us the value of model comparisons, or how different models performed against each other. We learned that this problem is most likely not suited for logistic regressions; after optimizing the hyperparameters using grid search, we found no improvement. Similarly, because the dimensionality of our dataset was high due to the numerous features, k-nearest neighbors may not be the best classifier to use for this problem. On the contrary, tuning the neural network and decision tree resulted in an error rate reduction of over 10%, which may suggest it is fit for this classification problem.

Contributions

Calvin worked on exploring the data and preprocessing the dataset by removing non-impactful features, imputing missing values, and performing One-Hot-Encoding to convert categorical data points into numerical values. He also trained and tested the neural network model and gathered classification error rates between a default and tuned neural network.

Will worked on the logistic regression classifier and performed a grid search to find the most optimal hyperparameters to maximize the classifier's accuracy.

Alex trained and tested the k-nearest neighbors classifier along with the decision trees classifier. For both models, he experimented with different hyperparameters and collected the error rates.

As a group, we collectively put together this report showcasing our experimental setup, results, and insights.