CS 273A Project Report

By: Chase Overcash, Jonathan Dedinata, Josiah Lopez-Wild

Team Name: JJC ML_Diabetes 130-US hospitals

I. Introduction

For this group project, we chose the Diabetes 130-US hospitals for years 1999-2008 Data Set. The data describes the factors related to readmission as well as other outcomes pertaining to patients with diabetes. With this data set, we aim to train a machine learning model to accurately predict if a patient is more likely to be readmitted to the hospital, given a set of features. In this report, we will discuss what features are used to train our model, what machine learning model we chose and how we chose it, what hyper-parameters we used to train our model and how the model performed overall in accuracy.

II. Data Exploration and Preprocessing

In the data set, there are 101766 samples and 50 features. Before processing the data, we obtained a few tables to see the data each feature contained and what type of data it is. From the tables, we can see that there are a few features that have a high amount of "unknown" values. These features are "weight", "payer_code", and "medical_specialty".

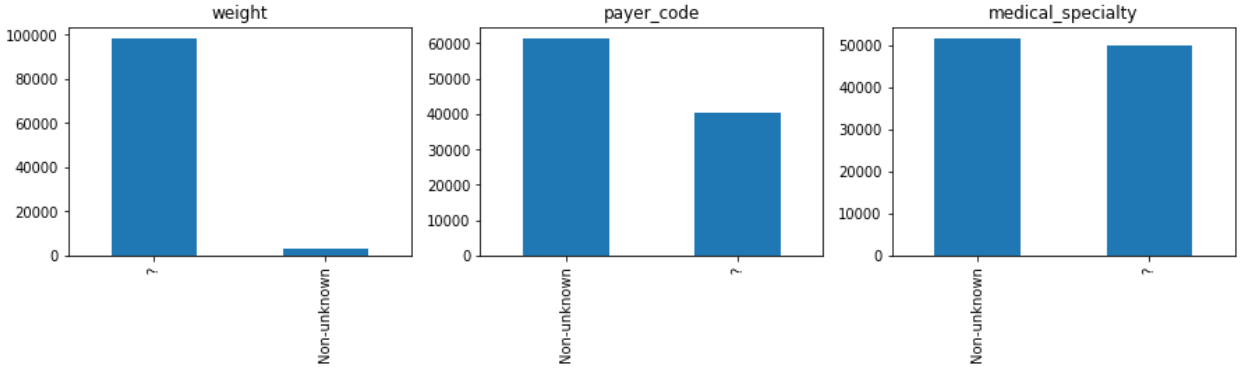

Fig 1. Features with high amounts of unknown values

To reduce inaccuracies, we removed these features from our training data set. Additionally, we also removed features that have a low amount of variance and features that are totally unique from one another, such as the "encounter_id". There were 20 features that had a low percentage of variance. This leaves us with 26 different features to train our models on. We also found duplicates of patient data, based on the data from the "patient_nbr" column. Removing the duplicates leaves us with 71518 samples.

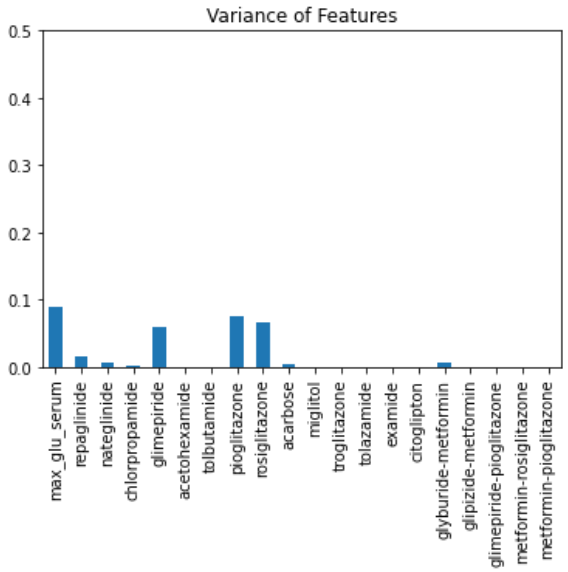

Fig 2. Features that has variance of less than 10%

We did further preprocessing by turning all "object-type" data types into int64 data type. Doing this, we were able to separate the data in each feature numerically and train our models based on these data. We also changed our target data into 2 labels, from "<30", ">30", and "NO" into "0" and "1", with 0 being readmitted and 1 being not readmitted. Doing this simplifies our training process as it becomes a binary classification problem.

III. Model Exploration and Validation

During data exploration, we tried to plot scatter points for each feature to see if there were 2 different features that can separate the data into classes. However, we were unable to conclude any correlation between just 2 features and we decided to train our model using all of the remaining features. As such, we used the following model types and plotted their ROC curves computed on the validation data: Linear Discriminant Analysis, Quadratic Discriminant Analysis, AdaBoost, Bagging, Extra Trees Ensemble, Gradient Boosting, Random Forest, Ridge, SGD, Bernoulli NB, Gaussian NB, K-Nearest Neighbors, MLP, Linear SVC, SVC, Decision Tree, and Extra Trees. We found that the top two performing model types are Gradient Boosting (AUC ~ 0.688) and Random Forest (AUC ~0.687). The next two highest were AdaBoost and Extra Trees. Ensemble methods are therefore the most reliable on this data set.

We then compared the performance of Gradient Boosting and Random Forest on training and validation data as hyperparameters were varied. For both model types, the relevant hyperparameters are:

the number of estimators trained in the ensemble (n_estimators); the method for selecting the maximum number of features for each estimator to be trained on, either the square root of the total, or the binary log of the total (max_features); the minimum number of sample required to split a node (min_samples_split); the minimum number of samples required to form a leaf (min_samples_leaf).

We first tested the two methods for determining the maximum number of features, sqrt and log2. We found that Gradient Boosting outperformed Random Forest, and that the sqrt method outperformed log2 at n_estimators = 1500. This value also maximizes AUC, so we will take n_estimators = 1500.

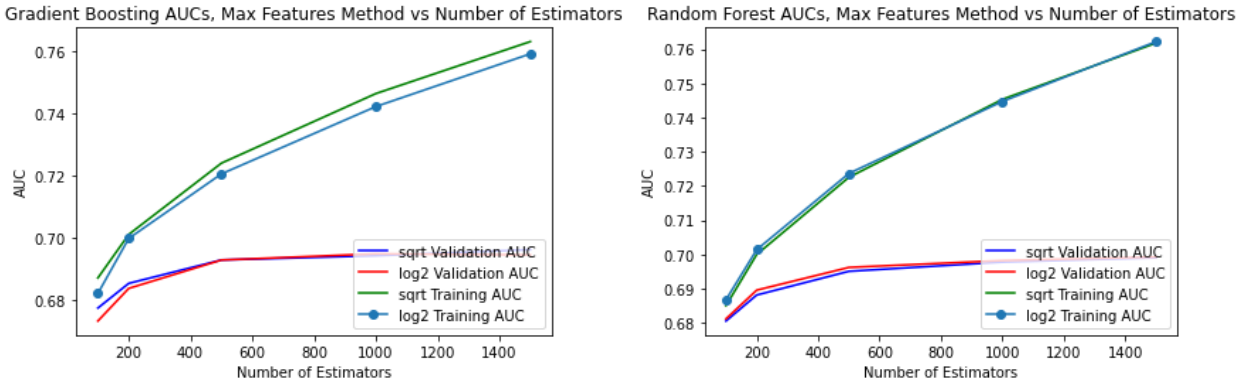

Fig 3. AUC of Gradient Boosting Model and Random Forest Model

We then tested the minimum split and minimum leaf values on Gradient Boosting with n_estimators = 1500. We find that the maximum AUC values fall in the min_samples_leaf = 100 column. In particular, this value is maximized when min_samples_split = 1000. So, the optimal hyperparameters are: n_estimators = 1500, max_features = sqrt, min_samples_split = 1000, min_samples_leaf = 100.

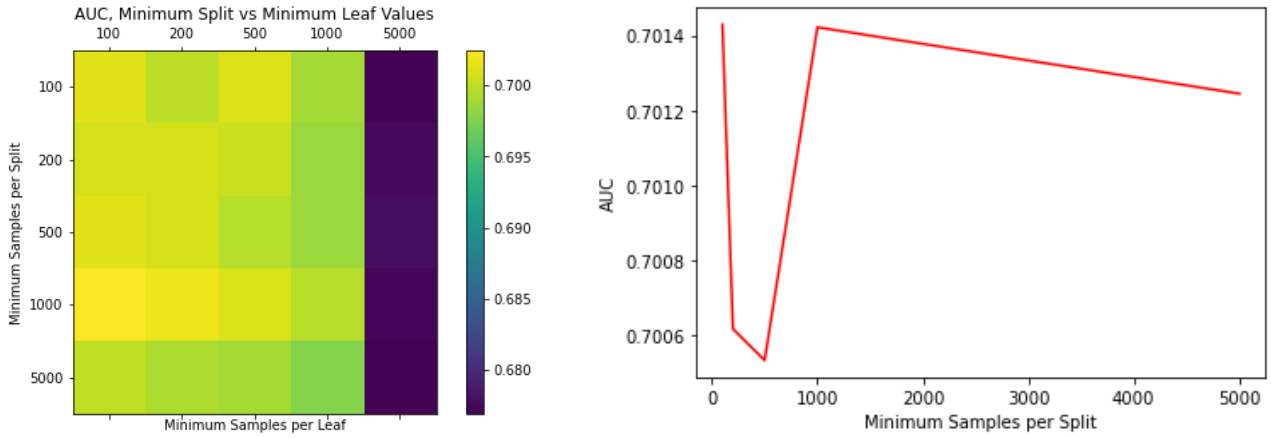

Fig 4. AUC of Gradient Boosting Model and Random Forest Model with different hyper-params

IV. Data Processing and Feature Design

Since tabular data lends itself well to a decision tree approach, that was an obvious place to start. However, design trees struggle with features that contain multiple possible outputs, in this case features such as “Admission Type ID” and “Admission Source ID”, became a problem since there was no obvious way to condense these multi-output features into a binary one. Thus for some models, we were given a choice between either abandoning the features for those particular models, or to find some factual way to split those IDs. Among the IDs for these features were often several unknown or unclear results such as “Not Available”, “Null”, “Not Mapped”. This meant that in order to compress the output ID’s in a meaningful way the feature had to turn into a question of specifics, for example “Could this hospital visit be seen as an escalation above a routine trip?” where IDs like “Emergency”, or “Urgent” would result in a binary output of 1, where everything else, including the not conclusive IDs would warrant an output of 0.

Another important thing to consider for decision trees is information gain. Taking some of the features that weren’t victim to blank outputs we could take a look at the information gain for features in comparison to each other.

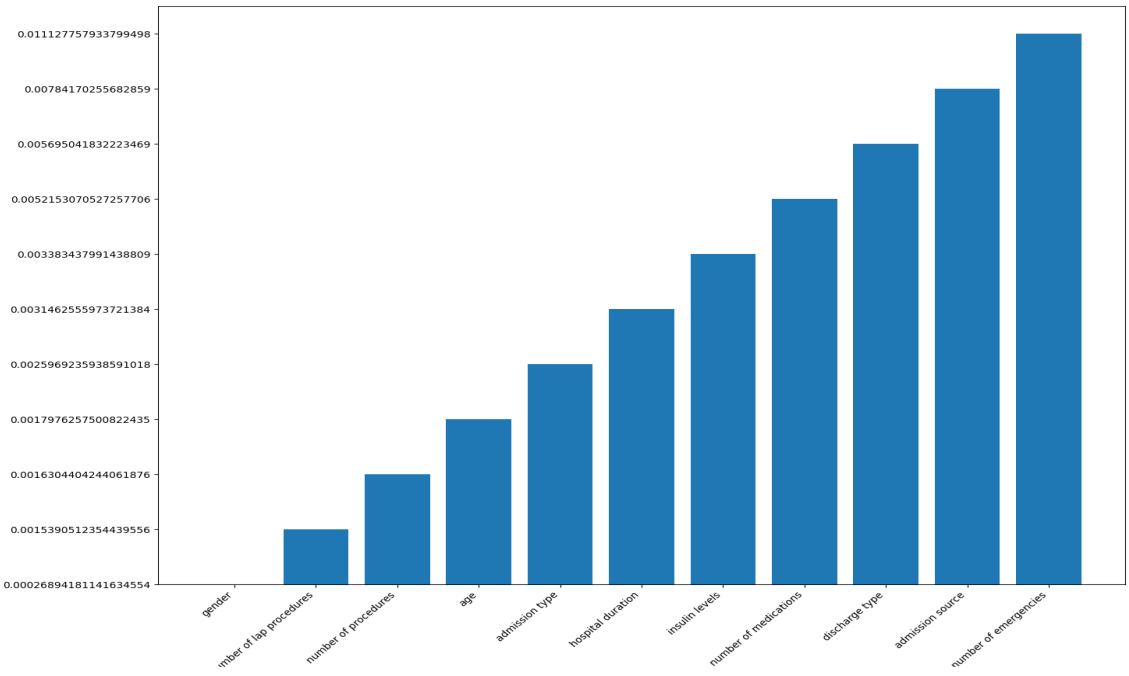

Fig 5. Information gain on different features

From this graph we can see the information rankings for a handful of features. Looking at this graph it is easy to understand what features are important to include and which ones have very little effect

splitting a tree, specifically gender seems to carry very little weight on the conclusion of a person's likelihood to need readmission, therefore that data can then be forgone either in favor of a different feature to be used, or for performance enhancement. With this information, we can decide on features for an initial design tree. The left hand side of a design tree based on information feedback from the above information gain graph, can be seen below.

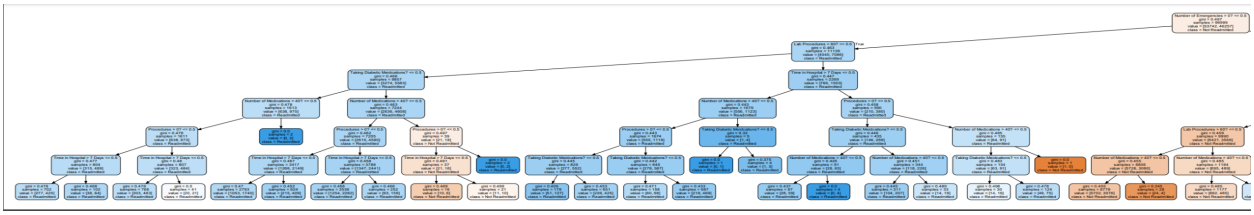

Fig 6. Design tree based on information feedback

For all the models we tested, it was important to cull features that weren't beneficial to the learning process. This mainly resulted in many cutting features which were inconsistently filled. Additionally, stuff that humans can perceive as likely not having recognizable patterns of cause and effects could also be removed from the dataset that we would test and train on, that includes things like patient number and encounter ID. Furthermore, things found to demonstrate little to no recognizable influence on the readmittance of a patient could be determined as unnecessary and could be removed, things like race and gender. Removing such features allows for other, more telling features to have a slightly larger impact on predicting the data. Pictured below is the graphed readmitted percentage per each decade age group.

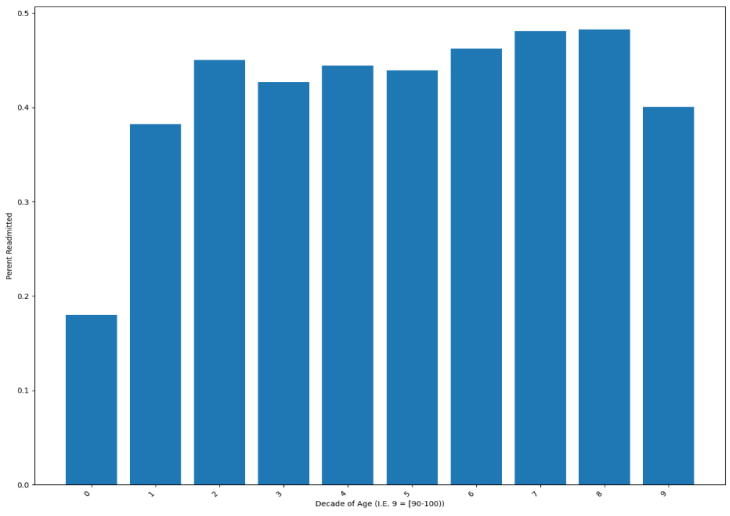

Fig 7. Likelihood that a patient will be readmitted by age group

V. Performance Validation

The easiest way to check the accuracy performance of these models is to cross validate them. As an example the decision tree built based on 12 features, that used information gain to balance what features were considered, had a prediction accuracy of approximately 62.5%, training on 70% of the dataset and testing with the remaining 30%.

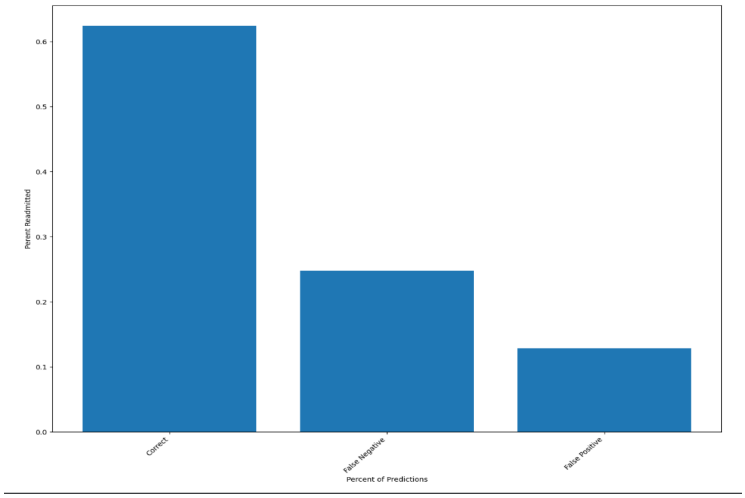

Fig 8. Ratio of correct, false-positive, and false-negative predictions

In the graph above you can see the ratio between false positive and false negative results. In order to further boost performance, this could be addressed and features could be managed in order to reduce false negatives with the hope of increasing the overall correctness of the model. Meanwhile, the models that we found performed best based on their AUC performances, Gradient Boosting and Random Forest, cross validated at a rate of 65.1% and 64.7% respectively, on the same training to test size ratio.

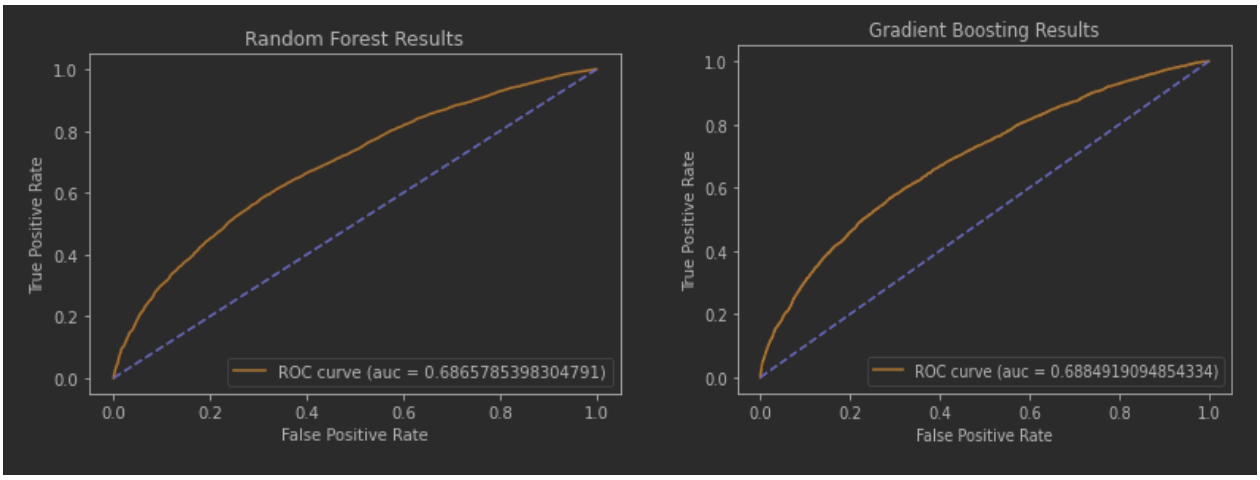

Fig 9. ROC comparison between Random Forest and Gradient Boosting after data processing

VI. Adaptation to Under- and Over-fitting

Since this is a tabular data set, it is amenable to a decision tree analysis. Decision trees have a tendency to overfit, so ensemble methods are more robust. We see that in our analysis, the two ensemble methods Gradient Boosting and Random Forest are the most accurate on validation data, presumably because a normal decision tree overfits on the training data.

While working with Gradient Boosting, there are two main hyperparameters that control for over- and under-fitting concerns: minimum samples per split and per leaf. Higher values for these hyperparameters guard against overfitting. This is because a model that can split nodes with a small number of samples can make branches which correspond to artifacts of the training data; similarly, if a node with many samples can be split into leaves, one of which has very few samples, then this leaf is likely an artifact of the training data. These splits make the model underperform on validation data. So, we tested our Gradient Boosting model on higher values for both hyperparameters, in the above two charts. We find that the model performs best with 1000 minimum samples per split, better than with a smaller value. The better performance is likely due to a decrease in overfitting.

VII. Team Member Roles

Chase Overcash:

Performance Validation, Data Processing, and Feature Design: worked on iterating a decision tree design to evaluate the effects of information gain, dropping and adding features, and different approaches to maximize performance. Cross validated models, investigated individual features.

Jonathan Dedinata:

Data and Model Exploration, Data Preprocessing: explored data and preprocessed data by filtering out features that have low variance and low impact on training predictors. Changed target labels to a binary class. Changed all features to numerical features using an ordinal encoder. Did model exploration by training all predictors and finding out each ROC and AUC values after training.

Josiah Lopez-Wild:

Model Exploration and Performance Validation: trained Gradient Boosting and Random Forests, plotted AUCs for number of estimators, minimum samples per leaf and split, decided hyperparameter values. Adaptation to Under- and Over-fitting: explained design choices for hyperparameters by way of fit.