Final Report

Group #7
Ben Edelhertz, Ellie Grimes, Carlos Von Borcke, Max Higareda, & Guillermo Maltes

Professor Tao Li
OMIS 115: Predictive Analytics
Santa Clara University
10 June 2021

I. Executive Summary

This data analytics exercise sought to predict the presence of heart disease in patients. Using research data from four medical institutions, an optimal model for binary and multiclass prediction was chosen from four classifiers. The project includes a discussion of the strengths of different model assessment metrics. Confusion matrices visualize the models' predictions.

II. Introduction

This data analytics exercise sought to predict the presence of heart disease in patients. Heart disease is the number one killer of men and women in the United States. It also is among the least understood, least researched, and least discussed chronic diseases. Predicting and diagnosing heart disease is a huge challenge in the medical industry and relies on factors such as the physical examination, symptoms and signs of the patient. Because of the life-or-death nature of the research problem, our team wanted to see how accurately we could predict the existence of heart disease in a patient. To create additional value for the model, we also aim to classify the heart disease by its stage/severity to determine which patients' treatment to prioritize.

III. Data & Methodology

Data

This data science exercise utilized research data from four medical institutions across the world. The data set is from 1988 and consists of four databases: Cleveland, Hungary, Switzerland, and Long Beach V. It contains 76 attributes, including the predicted attribute, but all published experiments refer to using a subset of 14 of them. The "target" field refers to the presence of heart disease in the patient and the stage of the heart disease. As shown in Exhibit 1, the target variable within the dataset is imbalanced, but when the target variable is converted to a binary, it is balanced.

Exhibit 1 – Multi-class Target Variable

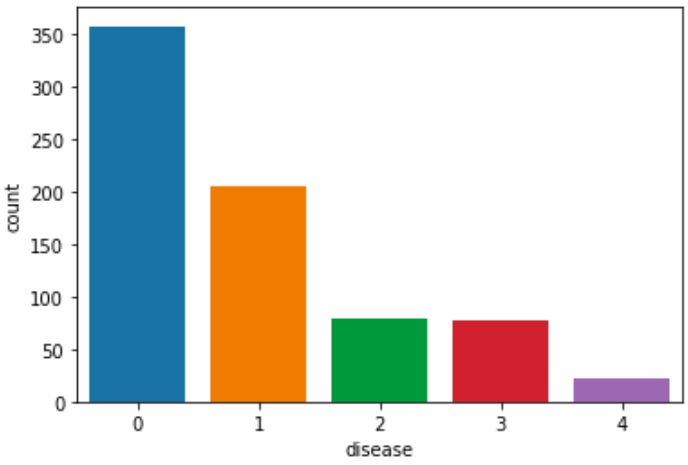
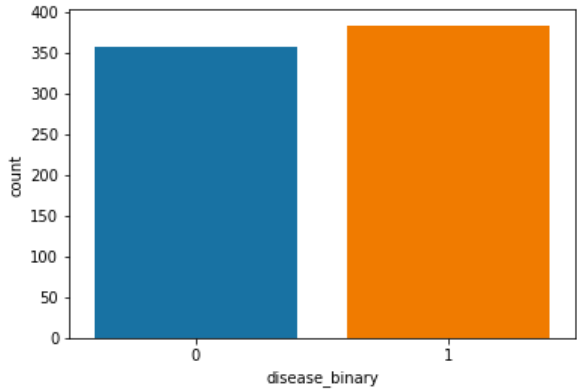


Exhibit 2 – Binary Target Variable



Cleaning

Before cleaning, the dataset contained non-calculable, categorical notations for most of its attributes. Dummy variables were created in order to make a binary variable that indicates whether our categorical variables take on a specific value. Additionally, a disease_binary field was created to indicate the existence of disease in the patient in any stage. Three columns that had large amounts of N/A values were dropped from the dataframe, so they would not affect the model. The final data frame is displayed in Exhibit 3.

Exhibit 3 – Cleaned Dataframe

	age	male	trestbps	chol	thalch	oldpeak	cp_atypical angina	cp_non-anginal	cp_typical angina	fbs	restecg_lv hypertrophy	restecg_st-t abnormality	exang	disease	disease_binary
0	63	1	145.0	233.0	150.0	2.3	0	0	1	1	1	0	0	0	0
1	67	1	160.0	286.0	108.0	1.5	0	0	0	0	1	0	1	2	1
2	67	1	120.0	229.0	129.0	2.6	0	0	0	0	1	0	1	1	1
3	37	1	130.0	250.0	187.0	3.5	0	1	0	0	0	0	0	0	0
4	41	0	130.0	204.0	172.0	1.4	1	0	0	0	1	0	0	0	0

Methodology

The methodology for this project was to use various classifiers to complete a binary and multi-class classification with optimal results given the context of the data. First, SMOTE was used to balance the data with an oversampling method. Then, a pipeline was created that scaled the training data, added polynomial features, and classified. A param_grid was utilized to test different classifiers and parameters, and then GridSearchCV determined the optimal classifier and parameters. Confusion matrices display the classification results. This process was repeated for both the binary and multi-class classification.

IV. SMOTE

SMOTE is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. SMOTE was used on the multi-class data, creating a balanced dataset.

V. Multi-class Classification

In multi-class classification problems, we need to categorize each sample into 1 of N different classes. This causes a problem with assessment as some metrics are defined for binary classification. In these cases, by default only the positive label is evaluated, assuming by default that the positive class is labelled 1. In extending a binary metric to multi-class problems, the data is treated as a collection of binary problems, one for each class.

Metrics

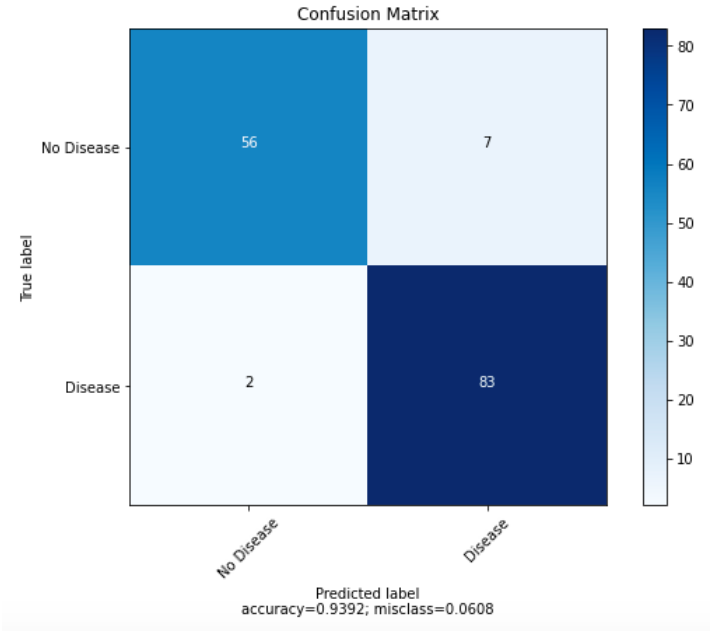
There are then a number of ways to average binary metric calculations across the set of classes, each of which may be useful in some scenarios. These suffixes apply to binary scoring metrics, which then applies the metrics to a collection of binary problems and uses these different strategies to find averages. Macro simply calculates the "mean of the binary metrics, giving equal weight to each class" (scikit-learn.org). In other words, it takes the binary score on each class, and finds the mean of all the classes. So, no matter how many data points are represented in each class, each class holds the same weight in scoring. When infrequent classes are still important, macro-averaging can help increase the relative importance of those classes. However, it is often the case that infrequent classes are not as important as frequent classes, so macro-averaging can over-emphasize the performance on infrequent classes. Micro gives each "sample-class pair an equal contribution to the overall metric" (scikit-learn.org). Micro does not take into account the binary class scoring, instead it takes each individual classification of a data point to score the model. It sums the "dividends and divisors that make up the per-class metrics to calculate an overall quotient" (scikit-learn.org). Therefore, the infrequent classes do not get equal weight, because each individual data point holds the same weight. Weighted accounts for class imbalance by "computing the average of binary metrics in which each class's score is weighted by its presence in the true data sample" (scikit-learn.org). With weighted, it still calculates the score for each binary class, but then as opposed to macro, it weights each class by its representation in the data before finding the mean score. As for AUC, it must be specified as _ovo or _ovr in multi-class cases, but weighted can also be specified. Ovo, which is one-vs-one, computes the average AUC of all possible combinations of classes. Ovr, which is one-vs-rest, computes the AUC of each single class against all other classes. Ovr is sensitive to class imbalance because imbalance affects the configuration of the rest of the classes for each AUC calculation. There is no 'best' metric, as each unique situation may call for a different metric.

VI. Results

Binary Classification

Gleaning insights from the confusion matrix, the model is quite effective, misclassifying only nine out of 148 cases in the testing set. Furthermore, since the model is predicting deadly diseases, avoiding false negatives is of utmost importance. In that vein, this model does predict noticeably less false negatives (2) than false positives (7).

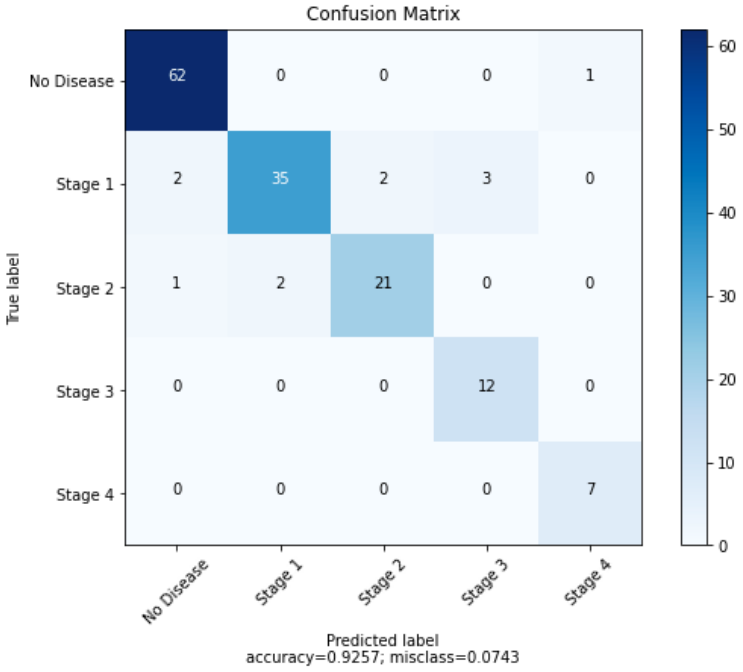
Exhibit 4 – Binary Classification Confusion Matrix



Multi-class Classification

For the multi-class classification model, scores are slightly lower than the binary classification, which is expected as adding categories makes it more difficult to predict. Our model is still very good at classifying cases in the multi-class scenario, accurately predicting the stage of heart disease for the most part with an accuracy score of over 92%.

Exhibit 5 – Multi-class Classification Confusion Matrix



While in the binary case we wanted to avoid false negatives, false negatives don't apply necessarily in multi-class classification because there is more than just a simple positive and negative. Recall or precision averages, macro, micro, or weighted, can still be used, however, in this case they overlap for the purpose of this project specifically.

Since higher stages are most severe, avoiding misclassifying those as no disease or low stage is most important. For example, we really want to avoid having cases that predict no disease when it is actually stage 4. So we could use recall, to make it so that we minimize the misclassification of stage 4. However, recall would also try to minimize misclassifications of no disease, which we are not really concerned about because if no disease is classified as stage 3 or 4, it is not a huge concern. On the other hand, we could maximize precision with no disease predictions which would help, but then we aren't really worried about precision with stage 4. This is why we use SMOTE and accuracy. SMOTE balances the data so that the accuracy of the higher stages take on greater significance even with less data points in the original data set. Then we can maximize accuracy with more weight applied to the higher stages, since recall and precision do not really maximize what we want.

The reason that is the case is because our classes are ranked. Zero to four, best to worst, lowest to highest. This makes it so that emphasizing recall or precision does not work across the board as it might in other cases. This could become a regression problem with the prediction ranging from 0-4, however, if the range is not linear, i.e. the difference between stage 1 and 2 is different from the difference between stage 3 and 4, as is likely the case, then the regression would not suffice.

With our data, we want to minimize the predictions in the lower left triangle of the matrix as shown in Exhibit 5, that is predictions that predict a lower stage than is actually the case. Unfortunately, there is no easy way to do that as might be the case in binary classification, however, with SMOTE or weighting, and accuracy we can still create a very good model to predict heart disease stages. And recall for the high stages is 100% with this strategy. We could also have used some of the other scoring techniques discussed earlier, like macro recall to weight higher stage data more than its representation in the data, but this strategy here with SMOTE got us the best score in this particular case.

VII. Conclusion

The learning point for this project is understanding how the same metrics and strategies translate from binary to multiclass classification so that the best strategy can be chosen based on the given situation. As is the case with binary classification, there is no universal best strategy, so each problem must be addressed on a case-by-case basis. In understanding the goals of the given model, we can utilize the applicable metrics and strategy to find the model that best fits those goals, whatever they may be.