Jordan Fanapour

Professor Predram Rooshenas

Intro to Machine Learning

May 1, 2023

Machine Learning Models to Predict Heart Disease

Heart disease is the leading cause of death in the world by a decent margin. In just the United States, over 695,000 people died from heart disease in 2021. That’s nearly 90,000 more than cancer and 280,000 more than COVID-19. In just that one year, as many people died from heart disease in America as one of its medium-sized cities, and it happens year after year. Justifiably, there is an enormous amount of research into not only treating and diagnosing heart disease, but also identifying its causes. As such, we already know many leading causes of heart disease: high blood pressure, high low-density lipoprotein (LDL) cholesterol, obesity, unhealthy diet, physical inactivity, etc. However, I was interested if I could accurately predict the occurrence of heart disease in a person from a small subset of surveyed attributes. I wanted to see how various machine learning models could perform on an adequately sized dataset of these datapoints.

From Kaggle’s database, I was able to download a dataset called, “Personal Key Indicators of Heart Disease.” The dataset was compiled by the CDC and the data was gathered from their 2020 annual survey with over 400,000 participants. The final dataset which I used had 319,795 rows, each one being an individual person surveyed in the study, and 18 columns. 17 of the columns are features ranging from whether the person is a smoker to whether they have kidney disease, and the last column is whether the participant has heart disease or not (the label). Using this dataset, I built and tested four different types of learning models: decision-tree classifiers, linear regression, logistic regression, and neural networks.

I spent most of my time trying to improve the performance of these models. For the decision-trees and the neural networks, I tried to improve statistics on the testing data such as accuracy and recall. Particularly, recall is important an application like predicting heart disease because recall indicates that a high proportion of people with heart disease are correctly diagnosed. Therefore, I spent most of my time trying to boost the recall of the models I made. For the regression models, I tried to improve the R-Squared and Mean Squared errors for them. I admittedly did not spend as much time on the two regression models as the two classifying models.

An important problem that I came across while training models on the dataset was the fact that heart disease was relatively uncommon in the dataset. Only 8.56% of the participants had heart disease which meant the label was severely imbalanced. This made it very difficult to build sophisticated models, especially for the decision tree. This imbalance made my models favor always picking the negative label because it would automatically guarantee 90 or so percent accuracy. To counteract this, I tried using stratified k-fold cross validation, but this resulted in a hit in other performance metrics. In the end, all the models had a very similar testing accuracies between 88-92% but the last neural network model I had a decent recall statistic of 0.85 for positive identification of heart disease while all the other models had very recalls; some as low as 0.08.

However, before I even began making the models, I had to preprocess the dataset. Most of the attributes who categorical. For example, the feature “Difficult Walkinig,” only had the values “Yes” and “No.” Unfortunately the two API’s I used (Sklearn and TensorFlow) only worked on numerical data. Therefore, each categorical attribute had to be encoded into numerical values. In the case before each “No” value would be converted to a 0 and each “Yes” would be converted into a 1. I ended up using various preprocessing tools from the sklearn api to automatically encode the values and replace them in the data frame.

Chart, treemap chart

Description automatically generatedAfter preprocessing, I built the first model using a DecisionTreeClassifier from sklearn. I thought decision tree would be a good choice for this project because the dataset has relatively few attributes and it is simple to understand. However, the imbalanced labels made the decision tree classifier perform worse than my expectations. It still got a moderate 86.3% testing accuracy, but it got a 0.25 recall for the positive label. I tried to improve the decision tree model by using stratified k-fold cross validation to help with the imbalance labels. However, after doing this I only got a ROC AUC value of 0.590 which is barely better than the default of 0.5. After doing some research, I found that random subsampling with a random forest model may improve my recall while maintaining a high accuracy. If I had more time, I would attempt to implement this.

After the decision tree classifier, I thought about making a kernelized support vector machine model. I found in practice in my other classes that a SVM was good at handling imbalanced datasets yielding a very high accuracy and recall. However, the datasets that we used in my other classes were much smaller than the one that I am using here and since the runtime complexity of SVM is O(N^3), it would have been infeasible for it to work in the time I had. Instead, I tried using two regression models.

The first one I tried was linear regression. The implementation was simple with sklearn and the model it produced had a low R-Squared value of 0.14 and mean squared value of 0.07 on the testing set. I tried to decrease the R-Squared and mean squared values even more by only using the best features of the dataset. I used RFECV from the feature selection library from sklearn but it only dropped one attribute (Mental Health). Out of curiosity, I made a second linear regression model from the 16 best attributes, but it ended up having the same R-Squared and mean squared values as the previous model. I then tried one logistic regression model using stratified k-fold. It ended up giving me an overall accuracy of 91.55%

Text

Description automatically generatedThe last group of models I made were Neural Networks using TensorFlow and Keras API’s. Initially a made a basic sequential model with two dense hidden layers with a ReLU activation and the output layer with a sigmoid activation. I built the model and only iterated it Chart, line chart

Description automatically generatedover 10 epochs. I graphed the training accuracy vs the validation accuracy. While using a threshold value of 0.5, I only got a recall of 0.1. However, if I changed the threshold of the output to something significantly lower like 0.09, I got a recall of 0.83 for the positive label and Chart, treemap chart

Description automatically generateda weighted accuracy of 0.91.

Chart, line chart

Description automatically generatedFor the last neural network, I used the same hyperparameters as the first one—the same number of layers and nodes. However, I standardized the dataset and trained the model for more epochs. I did this because it is good practice to standardize the inputs for any model and more epochs may have allowed the accuracy to stabilize. In addition, I graph the loss, precision, prc and recall of the second neural network. However, these were the statistics while training, so the threshold of the output was still 0.5. When I ran the test set, I set the output threshold to 0.09 like the first neural network model and a got a accuracy of 91.5% but a lower recall of 0.78 for the positive label.

Chart, treemap chart

Description automatically generatedGraphical user interface, chart

Description automatically generated

Out of all the models I made, the first neural network seemed to have performed the best. It had a high accuracy of 91% and moderately high recall for the positive label of 0.83. The regression models also performed well with their own statistics. However, I couldn’t compute their recall values so they may have both suffered as well from the imbalanced labels. The decision tree classifier was the biggest disappointment because it should have been a good model for this dataset even accounting for the imbalanced labels with stratified k-fold cross validation.

Overall, I got moderately good results from some of my models. Although, I believe I can do better by using different models or modifying the ones I already used. I think a support vector machine would have performed well if I had let the model finished and I used subsampling to account for the imbalanced labels. I also would like to try a random forest model with subsampling to replace the standard decision tree classifier; I’m curious whether it would produce better results. Lastly, I would like to modify one of the neural network’s hyper parameters to see if the performance could be improved. Maybe some different activation functions or maybe the addition of a recurrent layer would make a difference. If I were to revisit this project at some point soon, I would experiment with these models’ variations. I would also do more research about how to properly account for imbalance datasets because it was the main problem that I ran into while building the various models in the project.

References:

* <https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease>
* <https://scikit-learn.org/stable/modules/preprocessing.html#encoding-categorical-features>
* <https://inria.github.io/scikit-learn-mooc/python_scripts/03_categorical_pipeline.html>
* <https://stackoverflow.com/questions/72170947/how-to-use-ordinalencoder-to-set-custom-order> <https://www.projectpro.io/recipes/explain-stratified-k-fold-cross-validation>
* <https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html>
* <https://machinelearningmastery.com/cost-sensitive-decision-trees-for-imbalanced-classification/>
* <https://towardsdatascience.com/getting-started-with-xgboost-in-scikit-learn-f69f5f470a97>
* <https://www.atmosera.com/blog/binary-classification-with-neural-networks/>
* <https://towardsdatascience.com/beginners-guide-to-xgboost-for-classification-problems50f75aac5390>
* <https://www.tensorflow.org/tutorials/structured_data/imbalanced_data>
* <https://www.cdc.gov/nchs/fastats/leading-causes-of-death.htm>
* <https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death>
* <https://www.cdc.gov/chronicdisease/resources/publications/factsheets/heart-disease-stroke.htm#:~:text=Leading%20risk%20factors%20for%20heart,unhealthy%20diet%2C%20and%20physical%20inactivity>.