Predicting Heart Disease with a Logistic Regression Model

Ianna Lewis

June 2, 2021

Introduction

Heart disease is the leading cause of death worldwide — causing about 18 million deaths in 2019, and the death toll continues to rise.[1] The term heart disease refers to a variety of heart conditions, the most prevalent being coronary heart disease, which can lead to death.[2] Early and accurate diagnosis of this condition is critical in order to minimize damage and save patients' lives. Modern technology plays an important role in a variety of industrial sectors, allowing them to advance more rapidly. Machine learning applied to medical data, in particular, may be a useful method for analyzing patient survival rates and identifying key risk factors. Chest pain is the most common symptom among patients for heart disease. However, chest pain is not a sufficient condition — there may be other symptoms or factors like age, ECG signal, serum cholesterol in the blood, and more. Thus, it is possible to establish the relationship between these factors and heart disease through machine learning.

Several machine learning algorithms are increasingly being used to predict cardiovascular disease. Our goal, through this project, is to evaluate the overall predictive accuracy of machine learning algorithms in heart disease based on various attributes (sex, age, cholesterol level, etc.) that a user inputs into our application. We have used the heart disease dataset from University of California, Irvine, that originates from 4 different databases from hospitals in Cleveland, Hungary, Switzerland, and VA Long Beach to build our model.

Over the course of the quarter, our team has delegated tasks to create a Logistic Regression model that trains and tests on attributes in the heart disease dataset. In this report, we've detailed the frameworks of this project in order to explain how the model was built and how it functions. The literature review describes how our team analyzed trends in similar heart disease datasets and the data sections outline how the data was analyzed and preprocessed to find the best attributes to use. The proposed solution and experimental findings show how we implemented our solution and devised the best algorithm to assess our dataset, concentrating on certain attributes, as well as the logic behind particular machine learning decisions and subsequent outcomes.

Literature Review

To help us deduce the most optimal approach for our project, we scoped existing datasets and research papers to help us come up with correlations in the data, and to help us further understand our dataset. From our thorough background research, we discovered a dataset that was quite comparable to ours, called the "Framingham Heart Study" dataset, which is a cardiovascular study on residents of the town of Framingham, Massachusetts.[3] It evaluates the ten-year risk of developing cardiovascular disease.

When going through this dataset, and evaluating the correlation map, we observed that there were no features with more than 0.5 correlation with the ten-year risk of developing coronary heart disease. We concluded that this observation showed that the features were poor predictors, and that the results from the correlation matrix prompted the need for feature selection. Additionally, we found that the features with the highest correlations were age, prevalent hypertension, and systolic blood pressure. The algorithm that was used by this study for feature selection was the "Boruta Feature Selection Algorithm,"[4] which is a wrapper method built around the random forest classification algorithm. To develop the models, four algorithms — Logistic Regression, k-Nearest Neighbours, Decision Trees, and Support Vector Machine — were used and their performance was evaluated and compared. We saw that age and blood pressure were shared features when we compared the attributes to our dataset.

Through our findings, we deduced that logistic regression was the most appropriate and accurate algorithm for us to use in order to develop our model. A few of the key attributes we planned to focus on were age, blood pressure, and chest pain.

Dataset Description

Overview

The dataset for this project was a set of attributes comprising various physical characteristics of 303 patient instances. This multivariate data was taken from UCI's Machine Learning Repository and included both categorical and numerical attributes. The 75 original attributes were narrowed down to a subset of 14 relevant ones, including a target variable representing the presence or absence of heart disease. Values of 1, 2, 3, and 4 represent the presence of heart disease while a value of 0 represents an absence. [5]

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0

Figure 1: A subset of the data which shows the 14 attributes and range of values.

Description of Attributes

These are all of the attributes we deemed relevant.

- age: age in years
- sex: sex (1 = male; 0 = female)
- cp: chest pain type
  - Value 1: typical angina
  - Value 2: atypical angina
  - Value 3: non-anginal pain
  - Value 4: asymptomatic
- trestbps: resting blood pressure
- chol: serum cholesterol
- fbs: fasting blood sugar $120mg/dl(1 = true; 0 = false)restecg : restingelectrocardiographicresults$
- thalach: maximum heart rate achieved
- exang: exercise induced angina status (1 = yes; 0 = no)
- oldpeak: ST depression induced by exercise relative to rest
- slope: the slope of the peak exercise ST segment
- ca: number of major vessels (0-3) colored by flourosopy
- thal: thalassemia (3 = normal/negative; 6 = fixed defect; 7 = reversable defect)

Data Preprocessing and Visualization

Preprocessing

The data consisted of six instances which possessed missing data. Rather than opt to impute the missing data, our solution was to simply drop these six rows due to the relatively small amount of missing data.

The numerical data was normalized using scikit-learn's MinMaxScaler function to improve the integrity of the data. The data was split into training and test sets with a 70:30 ratio.

Visualization

The occurrence of heart disease was visualized among patients across different genders. Male patients were shown to have a higher likelihood of having heart disease than female patients. [6]

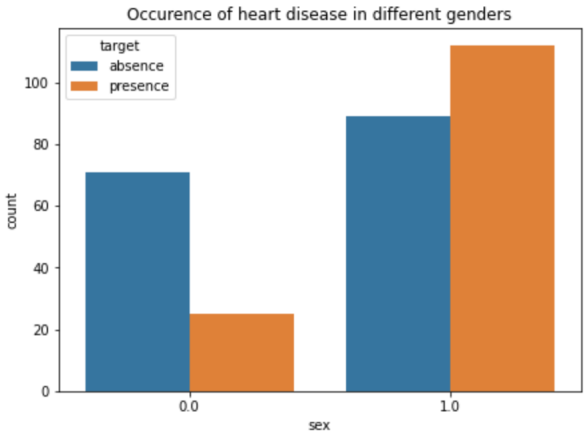


Figure 2: Occurence of heart disease in different genders (Sex value of 0.0 = Female; Sex value of 1.0 = Male)

The occurrence of heart disease among different age groups was also visualized. The presence of heart disease becomes more common as the age of patients increases. The likelihood of the presence remains higher than that of an absence in the range of ages 55-63. [6]

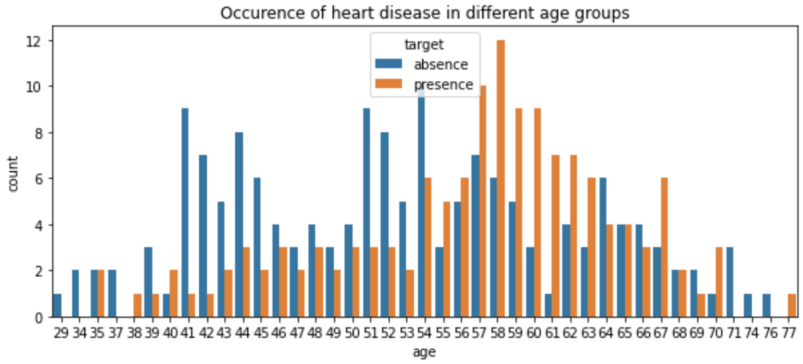
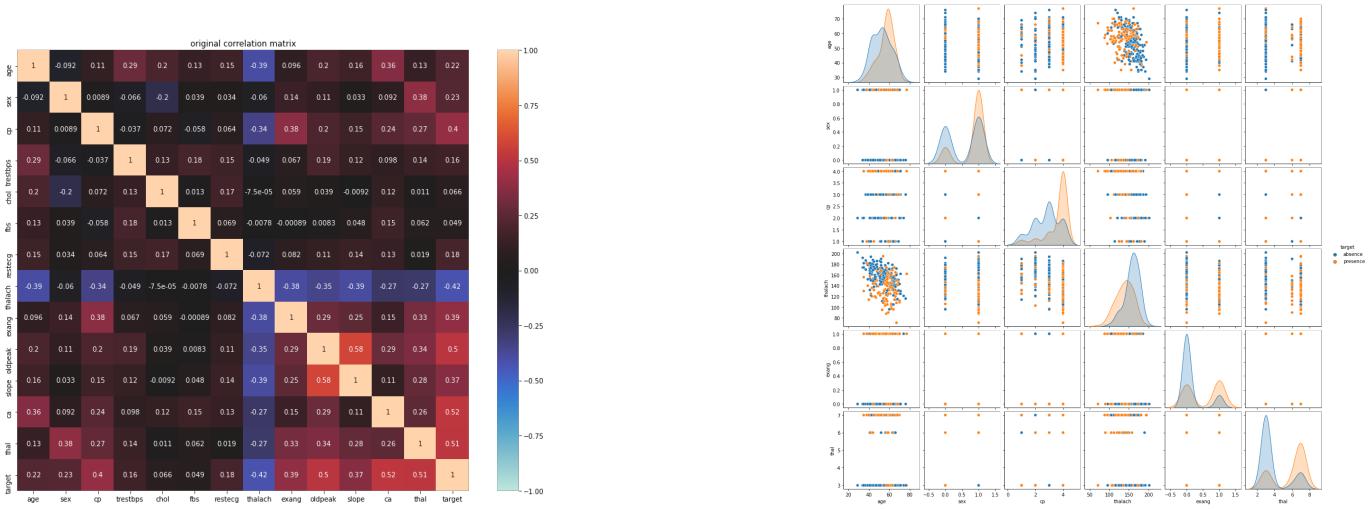


Figure 3: Occurence of heart disease across different ages

The potential linkage between the attributes is further explored using seaborn's heatmap and pairplot functions. Both the heatmap (or correlation matrix) and pairplot reveals exclusive relationships between attribute pairs. For instance, the lower the maximum heart rate achieved (thalach), the higher the chance of having a heart disease presence result.



(a) Heat map

(b) Pairplot

Figure 4: Correlation between attributes

Proposed Solution and Experimental Results

Solution Overview

After the cleaned dataset was been retrieved, the best model to use for predicting the target values had to be determined. Logistic regression seemed to be the most appropriate solution, based on previous research and our experimental outcomes.

Logistic regression, by default, is limited to two-class classification problems. However, in our project, the data are classified into multiple classes; we needed the extension of logistic regression: multinomial logistic regression. Compared to the binary problem, the logistic function is replaced with a softmax function:

$$P(y_i = k \mid X) = \frac{e^{\sum \beta_{ki} x_i}}{\sum_{j=1}^{K} e^{\sum \beta_{ji} x_i}}$$

(1)

Where K is the total number of classes for labels, and $P(y_i = k \mid X)$ is the probability that the ith observation falls into class k. The predicted label will be the class with the largest probability.

We tried several models to test which algorithm would best predict presence of heart disease, but ultimately logistic regression proved to be the most effective.

Tested Models

Logistic Regression

Just to be sure of the effectiveness of our logistic regression model, three other methods were tested out as well: SVM, Random Forest, and Decision Tree.[6]

For SVM the following parameters were tested: gamma was 'auto,' the regularization parameter was [1, 10, 20], and kernel was either rbf or linear.[7] For Random Forest, the number of estimators tested were 1, 5, and 10.[8] For logistic regression, the solver used was lbfgs, and the regularization parameter used was either 1, 5, or 10.[9] Lastly, for Decision Tree, the criterion tested was either gini or entropy.[10][11]

Accuracy, precision, and recall were all collected. When obtaining the accuracy score, the parameters were tested using GridSearch with 10 cross validation folds. This was done because the amount of data we could use for testing was not a lot, about 200 samples, hence this would help us maximize the possible training and testing data sets.

For precision and recall, in order to lower the randomness of scores, the training and test dataset was kept constant for all 4 methods.

As seen in Figure 5, logistic regression comes out on top. However, there were still many hyperparameters that had to be tuned in order to improve the accuracy.

	SVM	Logistic Regression	Random Forest	Decision Tree
Recall	0.2	0.3015	0.3388	0.26
Precision	0.112	0.295	0.3648	0.2513
Cross Validation (Accuracy)	0.5372	0.5777	0.5704	0.4397

Figure 5: Summarized data of all four methods used

The original method used to test our parameters was comparing cross validation scores given some change in parameter. This was done using the max iteration parameter, with the following max iterations tested: [100, 250, 500, 750, 1000, 1500, 2000, 2500, 5000]. Figure 6 shows the cross validation accuracy across the iterations.

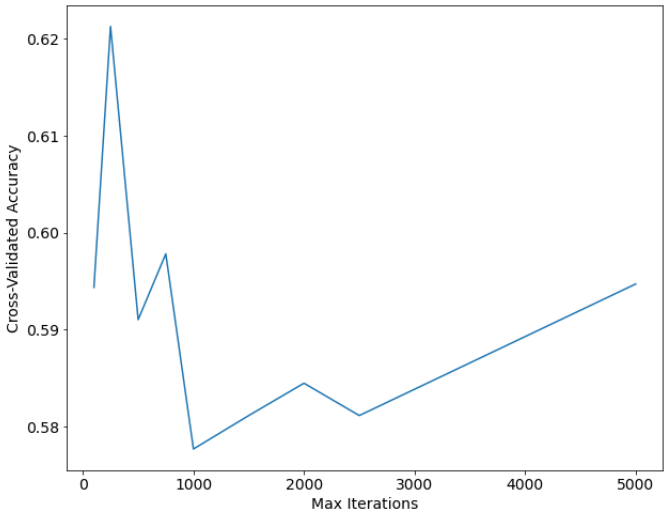


Figure 6: Graph of Max iterations versus Cross Validation accuracy

As seen above, the best accuracy is at 250 max iterations. However, one major issue with pursuing this method of testing is that it would get extremely tedious with the number of combinations of parameters. In order to navigate around this problem, GridSearch was once again implemented.

GridSearch essentially functioned with a similar logic except it was able to test out all possible combinations of the parameters, which can be seen below in Figure 7. [12] [13]

```
param_grid = [
    {'penalty' : ['l1', 'l2', 'elasticnet', 'none'],
    'C' : np.logspace(-4,4,20),
    'solver' : ['lbfgs','newton-cg','liblinear','sag','saga'],
    'max_iter' : [1000,2500, 5000]
    }
]
```

Figure 7: Full list of parameters tested out for Logistic Regression

The final output of the gridsearch yielded the following results as shown in Figure 8, with a final accuracy score of 0.625

```
LogisticRegression(C=0.012742749857031334, class_weight=None, dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=1000, multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='newton-cg', tol=0.0001, verbose=0,
                   warm_start=False)
```

Figure 8: Final parameters for Logistic Regression

Artificial Neural Network

We also wanted to test ANN models to make sure that logistic regression was the model that yielded the best results. We decided to use a 3 layered ANN model with input dimensions of (13, 14) and with our output layer being 1. We chose these dimensions based on GridSearch, which provided us with these layers.

After figuring out the layers that worked best on our model, we decided to test each combination of activation functions and optimizers to see which one worked best out of our activation functions (sigmoid, relu, softmax) and our optimizers (sgd, adam). We also tried changing the number of epochs our program ran for, however it did not change the accuracy of our model drastically.

```
#relu model
model = Sequential()
model.add(Dense(activation = "relu", input_dim = 13,
                units = 8, kernel_initializer = "uniform"))
model.add(Dense(activation = "relu", units = 14,
                kernel_initializer = "uniform"))
model.add(Dense(activation = "sigmoid", units = 1,
                kernel_initializer = "uniform"))
model.compile(optimizer = 'adam' , loss = 'mse',
              metrics = ['accuracy'] )
model.fit(x_train , y_train , batch_size = 8 ,epochs = 100)
history = model.fit(x_train , y_train , batch_size = 8 ,epochs = 100)

[ ] {x:history.history[x][-1] for x in history.history.keys() if x in ['loss', 'accuracy']}

    {'accuracy': 0.6425120830535889, 'loss': 1.0891438722610474}
```

Figure 9: The template for our ANN models

After testing each combination of each ANN model, here is our table with the loss and accuracy rates.

	RELU/RELU/ SIG/adam	relu model with softmax & adam	relu with sdg	sig model with adam	sig model with sgd
Loss	1.08914	1.55555	1.20601	1.16277	1.55410
Accuracy	0.64251	0.17874	0.56038	0.57971	0.17874

Figure 10

As we can see, our best model for ANN was the model that had the activation functions with relu, relu, and sigmoid in each of the layers yielding an accuracy of 0.64251. However, logistic regression was still a better model for predicting heart disease because it had a higher accuracy of 64.5%.

Linear Regression

We tried linear regression to see if we could get a more accurate output in decimals instead of just 0, 1, 2, 3, 4. With the decimals, we could possibly be able to tell our users the percentage of how likely they are to get a heart disease. For example, 0 = 20%, 1 = 40%, 2 = 60%, 3 = 80%, 4 = 100%, and the decimal values we get such as 1.9 would be a percentage between 40% and 60%. However, since the target values in our dataset were integer values, using linear regression would result in a high MSE because our output would be decimals instead of integers. As a result, our MSE

was 72.67%, which was too high. The accuracy for the logistic model was 64.5% and the accuracy for the ANN model was 64.2%, which made both models a better option than a linear regression model.

User Interface

We used the Voila extension on Jupyter notebook to create the user interface.[14] The widgets integrated well with our model, as Jupyter supports both Python and the Voila extension to create an interface. In addition, the UI/UX team isn't familiar with frontend languages like HTML or CSS, so being able to use Python in new ways was the most efficient process for our team. After installing Voila and other necessary extensions in the terminal of the folder we worked in, we began creating widgets in the notebook. These included an image and several types of widgets for user input, including Radio Buttons, textboxes for float type inputs, and dropdown menus to select from certain values. We also used HTML text for the title and subtitle of our page.

All of the user input was then accessed through the values method of the widgets and passed into the model, which was directly copied into the notebook. The model was run and trained locally in the notebook before the input was passed. The interface outputs a single line saying whether or not the model predicts the user has heart disease based on their health information, with nonzero values predicting presence of heart disease and zero predicting no presence of heart disease.

Conclusion and Discussion

With this project, we started with a baseline understanding that there were a plethora of machine learning algorithms being used to ascertain cardiovascular disease risk. Through our work, we hoped to evaluate different types of models ourselves with data from the UCI database.

There are a few considerations that should be made when working on a similar project in the future that each team discovered. Though the data team didn't run into issues per se, there was a disconnect in understanding the relevance and significance of certain technical medical terms. Perhaps having a stronger comprehension of those features would have allowed us to have a stronger conceptual understanding of why certain features had stronger correlations than others. Additionally, in future efforts, imputing missing data rather than dropping it could be an option as well as trying differing train-test ratios, since the less training data there is, the greater variance in the parameter values.

Another consideration was that the GridSearch ended up taking over an hour to complete and longer based on folds used during cross validation, as discussed in the algorithm section. With more time to work on this, we could conduct a GridSearch on support vector machine (SVM) and random forest models since they had similar accuracy levels to the logistic regression.

Our code can be found in our GitHub repository at SITE

References

[1] *Cardiovascular disease.* Wikipedia. URL https://en.wikipedia.org/wiki/Cardiovascular_disease.

[2] *Heart Disease.* cdc.gov. URL https://www.cdc.gov/heartdisease/index.htm#:~:text=Related%20Pages,can%20lead%20to%20heart%20attack.

[3] *Framingham Heart Study Dataset.* National Heart, Lung, and Blood Institute. URL https://biolincc.nhlbi.nih.gov/studies/framcohort/.

[4] Deepanshu. Bhalla. *Select Important Variables using Boruta Algorithm.* 2017. URL https://www.datasciencecentral.com/profiles/blogs/select-important-variables-using-boruta-algorithm.

[5] D. Dua and C. Graff. *UCI Machine Learning Repository.* 2019. URL http://archive.ics.uci.edu/ml.

[6] Shubhankar. Rawat. *Heart Disease Prediction.* towardsdatascience, 2019. URL https://towardsdatascience.com/heart-disease-prediction-73468d630cfc.

[7] *SVM.* scikit-learn. URL https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[8] *Random Forest.* scikit-learn. URL https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[9] *Logistic Regression.* scikit-learn. URL https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

[10] *Decision Tree.* scikit-learn. URL https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[11] *Machine Learning Tutorial Python - 16: Hyper parameter Tuning (GridSearchCV).* codebasics, 2019. URL https://www.youtube.com/watch?v=HdlDYng8g9s&t=444s.

[12] *GridSearchCV.* scikit-learn. URL https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[13] Kunaal. Naik. *P2 : Logistic Regression - hyperparameter tuning.* Kaggle, 2020. URL https://www.kaggle.com/funxexcel/p2-logistic-regression-hyperparameter-tuning.

[14] Hwei. Geok Ng. *How to Create an Interactive Web Application using a Jupyter Notebook.* finxter. URL https://blog.finxter.com/how-to-create-an-interactive-web-application-using-jupyter-notebook/.