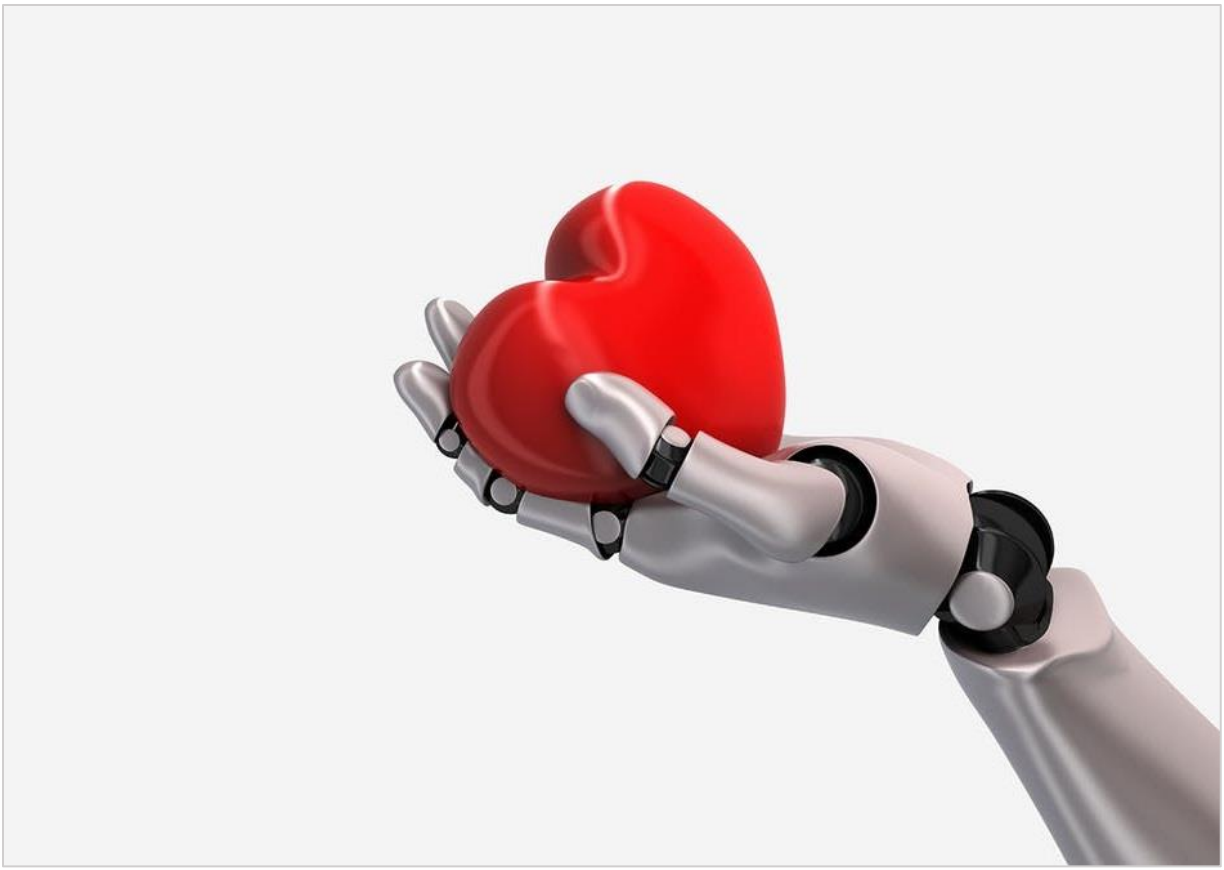


depositphotos

Heart Disease Diagnosis via Efficient Machine Learning

Su Kara

Advisor: Cheryl A. Johnson

Grade: 9

Capistrano Valley High School

Mission Viejo, CA

Presented at OCSEF

March 20-22, 2019

Abstract

Heart disease is the leading cause of death in the world, claiming 1 in 3 deaths globally. Physicians run several tests to identify whether a patient is at risk, but it's hard to rely on the result of a single test to make their final decision. Instead, it's a combination of several factors, and sometimes it may be hard to pinpoint the potential risk among too many parameters. My objective is to develop a web application for physicians to predict a patient's heart disease diagnosis by running a machine learning algorithm behind the scenes. Medical personnel can also add new patient data into the model to improve its accuracy over time. The tool can be used on any computer or mobile device.

Table of Contents

Introduction

Physicians have a hard time determining whether a patient's heart is healthy or not while taking several factors into account. My goal is to improve their decision-making process by creating a machine learning model from other patients' data and using that model to predict the current patient's health status.

Cardiovascular diseases (CVDs) kill an American every 34 seconds, and half of the population has some form of cardiovascular issue (Centers for Disease Control and Prevention, 2017). More people die annually from cardiovascular diseases than from any other cause. Around 17.9 million people died from CVDs in 2016 in the world, representing 1 in 3 deaths globally. 85% of these deaths are due to heart attack and stroke (World Health Organization, 2017).

People who have cardiovascular diseases or high risk need early detection and management using counselling and medicines. Physicians run several tests to diagnose coronary heart disease, which range from echocardiogram to stress test, from CT scan to MRI (National Institute of Health, 2018).

Physicians may also take other factors into account such as the age and gender of the patient, or their cholesterol levels. However, it may turn into a daunting task to figure out whether the patient has a potential for high risk among all those factors. Machine learning can be of great help to build a model from existing patient data and to predict the health status of new patients by using that model.

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions,

relying on patterns and inference instead (Wikipedia, 2019). Machine learning algorithms build a mathematical model of "training data" to make predictions without being programmed to perform the task. The types of machine learning algorithms differ in their approach based on the type of task or problem that they are intended to solve.

Problem Types

Machine learning problems can be grouped into classification and regression. Classification is the task of predicting a discrete class label, while regression is the task of predicting a continuous quantity (Machine Learning Mastery, 2019).

Classification is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y). The output variables are often called labels or categories. The mapping function predicts the class or category for a given observation. For example, an email of text can be classified as “spam” and “not spam”.

- Input variables can be real-valued (numbers) or discrete (nominals).
- Examples should be classified into one of two or more classes.
- Examples with two classes are often called two-class or binary classification.
- Examples with more than two classes are often called multi-class classification.

The most common way to estimate the skill of a classification model is to calculate the classification accuracy. The classification accuracy is the percentage of correctly classified examples out of all predictions made.

Regression is the task of approximating a mapping function (f) from input variables (X) to a continuous output variable (y). A continuous output variable is a real-value, such as an integer or a floating-point number. These are often quantities, such as amounts and sizes. For example, a house may be predicted to sell for a specific dollar value, perhaps in the range of $100,000 to $200,000.

- Input variables can be real-valued (numbers) or discrete (nominals).
- Requires the prediction of a quantity (number).
- A problem with multiple input variables is often called multivariate regression.

The most common way to estimate the skill of a regression predictive model is to calculate the root mean squared error.

Learning Methods

Machine learning methods can be grouped by the learning type (Machine Learning Mastery, 2019).

Supervised Learning: All data is labeled, and the algorithms learn to predict the output from the input data.

Unsupervised Learning: All data is unlabeled, and the algorithms learn to inherent structure from the input data by clustering and association.

Semi-supervised Learning: Some data is labeled but most of it is unlabeled and a mixture of supervised and unsupervised techniques can be used.

Algorithms

Machine learning algorithms, also known as classifiers, can be grouped in terms of their functionality (Machine Learning Mastery, 2019).

Regression algorithms model the relationship between variables, which is iteratively refined by using a measure of error in the predictions. The most popular regression algorithms are Linear Regression and Logistic Regression, where the former is used for regression problems and the latter is used for classification problems.

Instance-based algorithms are used to solve decision problems with instances or examples of training data by using a similarity measure to find the best match and make a prediction. The most popular instance-based algorithm is k-Nearest Neighbor (kNN).

Decision tree algorithms construct a model of decisions made based on actual values of attributes in the data. Decisions fork in tree structures until a prediction is made for a given record. Decision trees are trained on data for classification and regression problems. The most popular decision tree algorithms are Classification and Regression Tree (CART), C4.5, and C5.0.

Bayesian algorithms are those that explicitly apply Bayes' Theorem for both classification and regression problems. The most popular Bayesian algorithms are Naive Bayes, Gaussian Naive Bayes, and Multinomial Naive Bayes

Clustering algorithms are concerned with using the inherent structures in the data to best organize the data into groups of maximum commonalities. The most popular clustering algorithms are k-Means and k-Medians.

Deep learning algorithms are concerned with building much larger and complex neural networks with semi-supervised learning problems where large datasets contain very little labeled data. The most popular deep learning algorithms are Deep Boltzmann Machine (DBM) and Convolutional Neural Network (CNN).

Ensemble algorithms are models composed of multiple weaker models that are independently trained and whose predictions are combined in some way to make the overall prediction. The most popular ensemble algorithms are Bootstrapped Aggregation (Bagging), Gradient Boosting Machines (GBM), and Random Forest.

Performance

When building a machine learning model, one of the most important steps in the process is evaluating the model's performance (Towards Data Science, 2018).

Confusion matrix shows the predicted vs. actual classification in a table as below:

	Negative (predicted)	**Positive (predicted)**
Negative (actual)	true negatives (tn)	false positives (fp)
Positive (actual)	false negatives (fn)	true positives (tp)

Accuracy tells us about the model's performance in terms of how correct the predictions are, but it doesn't give detailed information regarding its application to the problem.

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

Precision is the positive predictive value of the model as the fraction of relevant instances among the retrieved instances. It's the ratio of true positives to total predicted positives. Precision is a good measure when the cost of false positives is high such as email spam detection.

$$Precision = \frac{tp}{tp + fp}$$

Recall is the sensitivity of the model as the fraction of relevant instances that have been retrieved over the total amount of relevant instances. It's the ratio of true positives to total actual positives. Recall is a good measure when the cost of false negatives is high such as sick patient detection.

$$Recall = \frac{tp}{tp + fn}$$

Both precision and recall are based on an understanding and measure of relevance. Precision is how useful, and recall is how complete. High precision means that an algorithm returns substantially more relevant results than irrelevant ones, while high recall means that an algorithm returns most of the relevant results. Maximum precision is achieved when there are no false positives, while maximum recall is achieved when there are no false negatives.

F1 Score is a helpful measure of a test's accuracy. It is a consideration of both precision and recall, and an F1 score is considered perfect when it is 1 and a total failure when it is 0.

$$F1\ Score = \frac{2 \times precision \times recall}{precision + recall}$$

Balanced accuracy in PMLB computes each class' accuracy on a per-class basis using a one-vs-rest encoding, then computes an unweighted average of the class accuracies (Olson, La Cava, Orzechowski, Urbanowicz, and Moore, 2017).

Materials & Methods

I used the following hardware and software tools to develop this application:

- MacBook Pro by Apple
- Heart Disease Data from the UCI ML Repository
- Weka 3.8.3 by the University of Waikato, New Zealand
- Penn Machine Learning Benchmarks (PMLB) Tools
- Visual Studio Code 1.30.2 by Microsoft
- Anaconda Python 3.7
- Python Flask 1.0.2
- Scikit-learn 0.20.1
- HTML5 and JavaScript

I downloaded the Cleveland heart disease dataset from the UCI Machine Learning Repository website, which contained the data of 303 patients (UCI Machine Learning Repository, 2018). However, the data still had symbolic strings rather than nominal values, which needed to be cleaned up.

I read the Penn Machine Learning Benchmark (PMLB) research paper (Olson, La Cava, Orzechowski, Urbanowicz, and Moore, 2017) that compares the performance of machine learning algorithms. PMLB provided sample source code to evaluate those algorithms on any dataset. It also contained the exact same heart disease dataset in a proper format with nominal values for symbolic fields. I checked the nominal values for each attribute against the UCI heart disease dataset to identify the mappings between them. I noticed that healthy was set to one and sick was set to zero, so I swapped those settings to have sick as the positive value to match the

performance definitions explained earlier. Even though this operation didn't change anything at all in the machine learning model, it still looked more appropriate as a label setting.

I installed Weka 3.8.3 by University of Waikato, New Zealand, to examine the heart disease data in a graphical user interface (Weka, 2018). Since Weka requires the dataset in a specific format, I converted the PMLB heart disease dataset from "tsv" (tab separated values) into the "arff" format of Weka.

After getting familiar with the data in Weka, I used the Select Attributes tab for feature selection. I ran several methods to identify the most important attributes and created a new dataset with a subset of the original attributes.

I ran the PMLB toolset to evaluate the performance of several machine learning algorithms on the new heart disease dataset to identify the best performing algorithm and the best parameter combinations for each algorithm.

I installed Anaconda3 that contained Python 3.7, Visual Studio Code, Flask, and Scikit-learn packages (Anaconda, 2019). I used Visual Studio Code by Microsoft to develop the machine learning program for building a model from the new heart disease training data.

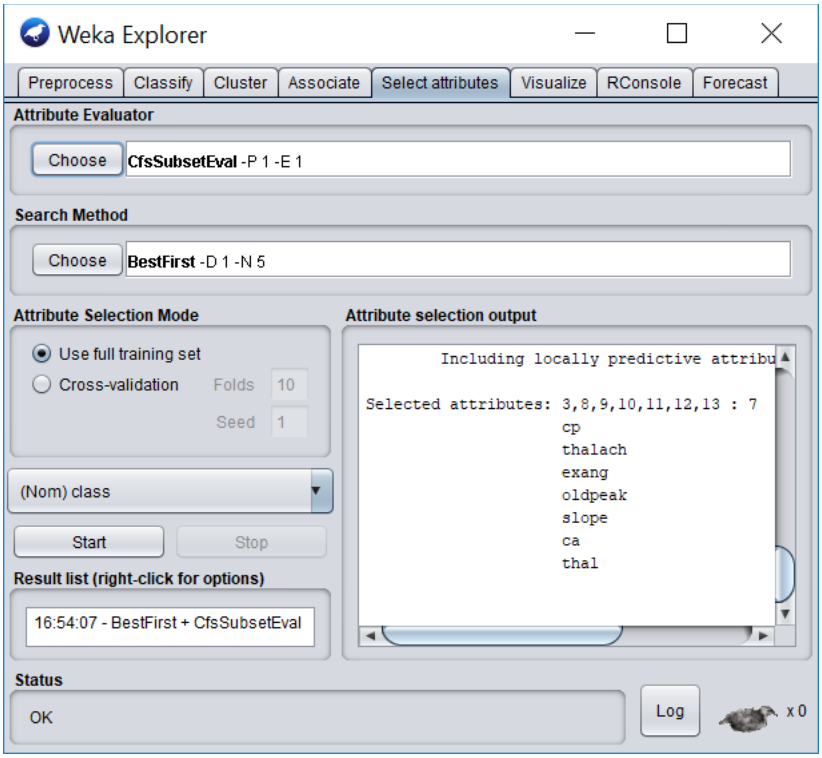
I developed my application in Python to build a machine learning model from heart data and to predict the health status of new patients. I created a web page in HTML5 and JavaScript, and a web service in Python Flask to communicate between the web page on the client machine and the machine learning application on the server.

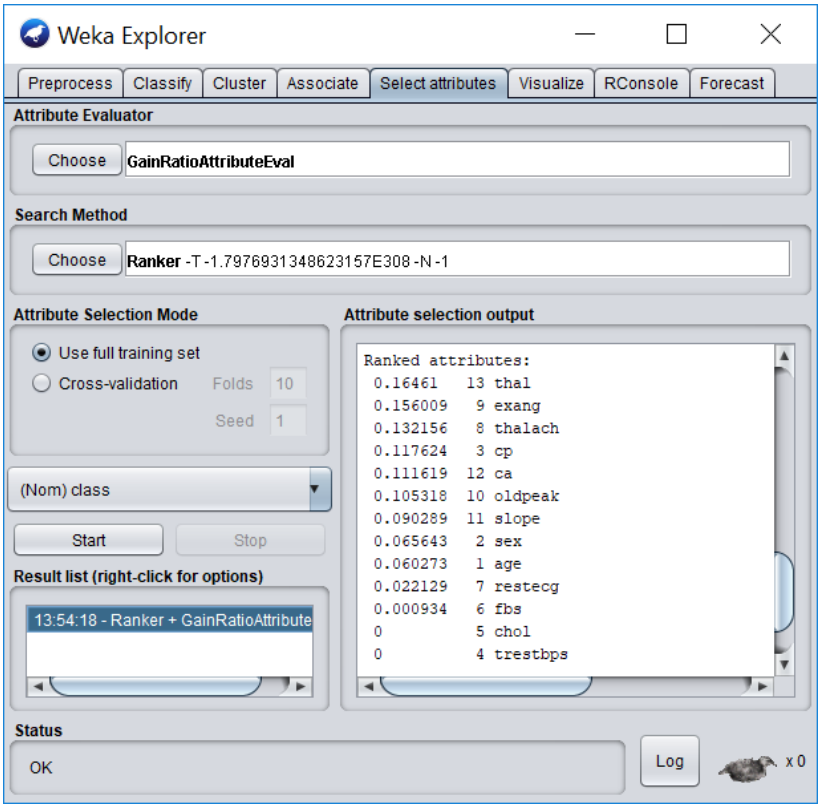
Results

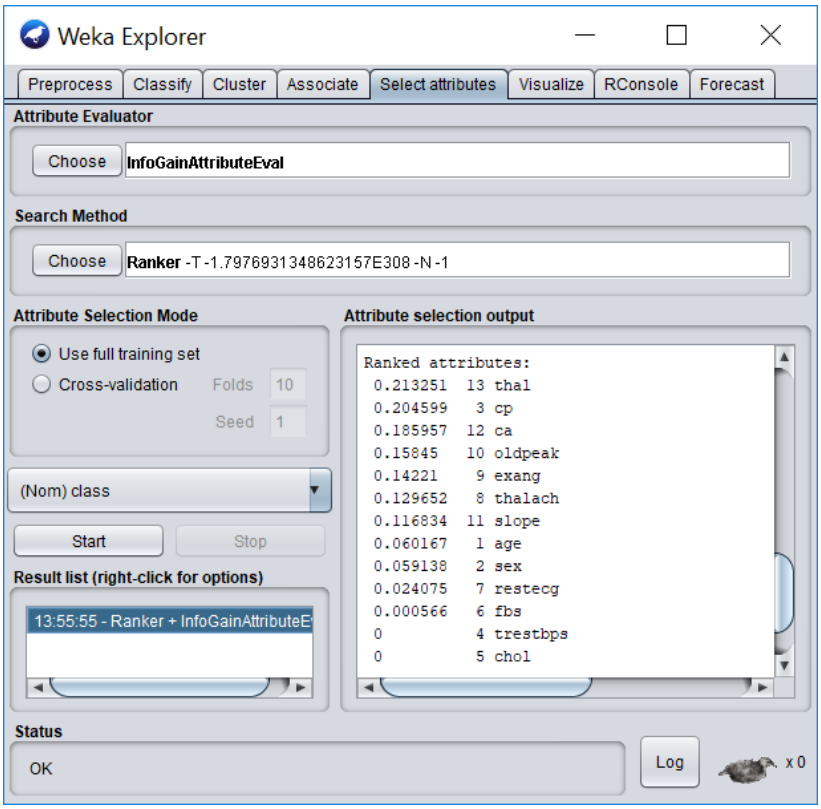
I used several tools to select the right set of attributes, to choose the best performing algorithm, and to identify the most efficient parameters for that algorithm.

Feature Selection

I used Weka to select the most important attributes. I applied several scheme-independent evaluators as shown in the following snapshots, and they all selected the same 7 attributes consistently on top: cp, thalach, exang, oldpeak, slope, ca, thal.




Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
RConsole
Forecast
Attribute Evaluator
Choose
GainRatioAttributeEval
Search Method
Choose
Ranker -T -1.7976931348623157E308 -N -1
Attribute Selection Mode
Use full training set
Cross-validation
Folds
10
Seed
1
(Nom) class
Start
Stop
Result list (right-click for options)
13:54:18 - Ranker + GainRatioAttribute
Attribute selection output
Ranked attributes:
0.16461 13 thal
0.156009 9 exang
0.132156 8 thalach
0.117624 3 cp
0.111619 12 ca
0.105318 10 oldpeak
0.090289 11 slope
0.065643 2 sex
0.060273 1 age
0.022129 7 restecg
0.000934 6 fbs
0 5 chol
0 4 trestbps
Status
OK
Log
x 0

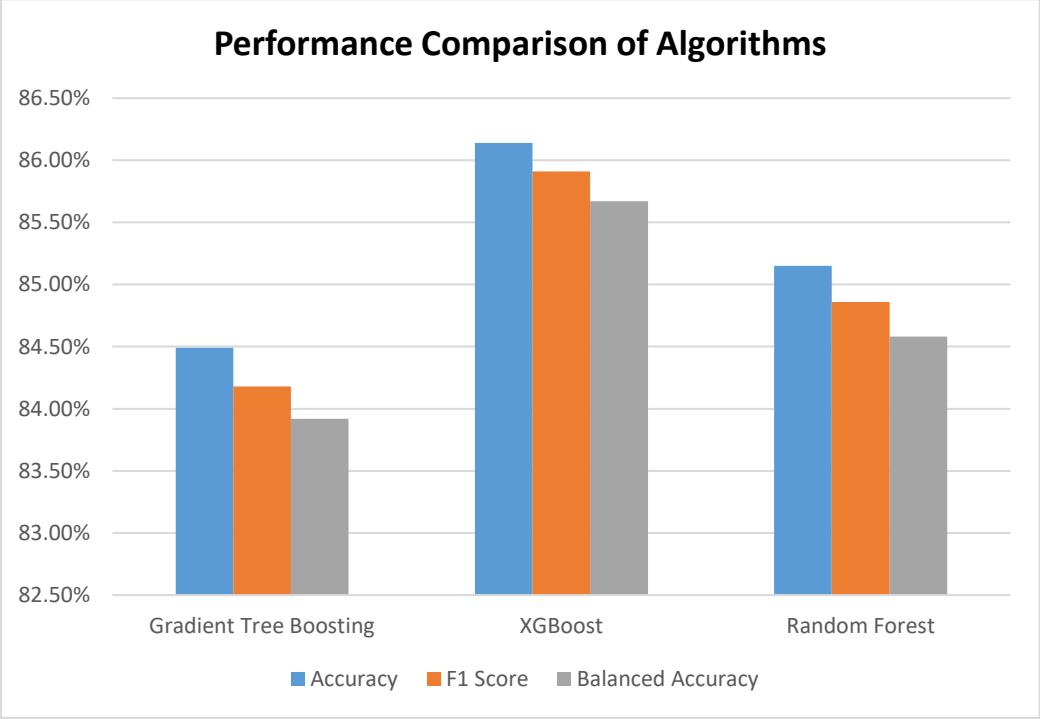

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
RConsole
Forecast
Attribute Evaluator
Choose
InfoGainAttributeEval
Search Method
Choose
Ranker -T -1.7976931348623157E308 -N -1
Attribute Selection Mode
Use full training set
Cross-validation
Folds
10
Seed
1
(Nom) class
Start
Stop
Result list (right-click for options)
13:55:55 - Ranker + InfoGainAttributeE
Attribute selection output
Ranked attributes:
0.213251 13 thal
0.204599 3 cp
0.185957 12 ca
0.15845 10 oldpeak
0.14221 9 exang
0.129652 8 thalach
0.116834 11 slope
0.060167 1 age
0.059138 2 sex
0.024075 7 restecg
0.000566 6 fbs
0 4 trestbps
0 5 chol
Status
OK
Log
x 0

Algorithm Selection

I used the PMLB toolset to compare the performance of Gradient Tree Boosting, XGBoost, and Random Forest algorithms with 10-fold stratified cross validation on the new dataset:

	Gradient Tree Boosting	XGBoost	Random Forest
Accuracy	84.49%	86.14%	85.15%
F1 Score	84.18%	85.91%	84.86%
Balanced Accuracy	83.92%	85.67%	84.58%

XGBoost showed the best overall performance with a balanced accuracy score of 86%.



Parameter Selection

I updated the PMLB evaluation source code to return the best parameters for each algorithm by fine-tuning all possible parameter combinations in a grid search fashion. I ran 34,160 evaluations in almost 7 hours for the following parameter combinations of each classifier:

	Parameter Settings	Evaluations	Time to Run
Gradient Tree Boosting	n_estimators_values = [10, 50, 100, 500] min_impurity_decrease_values = np.arange(0., 0.005, 0.00025) max_features_values = [0.1, 0.25, 0.5, 0.75, 'sqrt', 'log2', None] learning_rate_values = [0.01, 0.1, 0.5, 1.0, 10.0, 50.0, 100.0] loss_values = ['deviance', 'exponential']	**7,840** (4x20x7x7x2)	**1:58:01** (0.90 sec/eval)
XGBoost	n_estimators_values = [10, 50, 100, 500] learning_rate_values = [0.01, 0.1, 0.5, 1.0, 10.0, 50.0, 100.0] gamma_values = np.arange(0., 0.51, 0.05) max_depth_values = [1, 2, 3, 4, 5, 10, 20, 50, None] subsample_values = np.arange(0.0, 1.01, 0.1)	**25,200** (4x7x10x9x10)	**4:04:27** (0.58 sec/eval)
Random Forest	n_estimators_values = [10, 50, 100, 500] min_impurity_decrease_values = np.arange(0., 0.005, 0.00025) max_features_values = [0.1, 0.25, 0.5, 0.75, 'sqrt', 'log2', None] criterion_values = ['gini', 'entropy']	**1,120** (4x20x7x2)	**41:13** (2.21 sec/eval)
	Totals	**34,160**	**6:43:41**

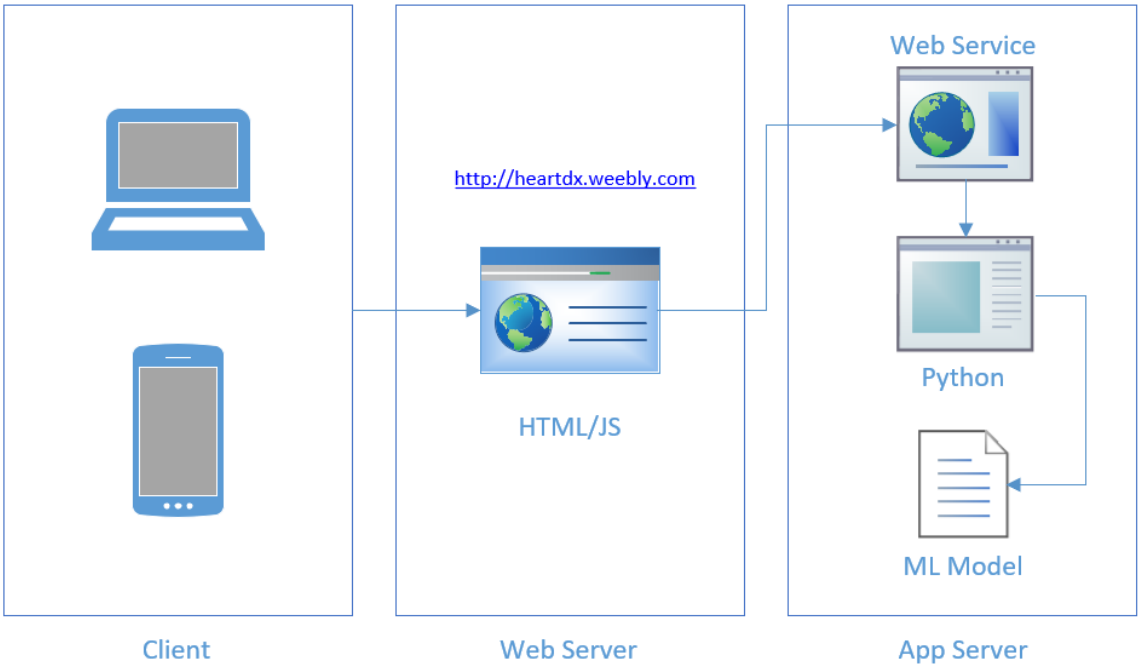
Here are the parameter settings of each classifier for the best performance:

	Gradient Tree Boosting	XGBoost	Random Forest
Parameter Settings	n_estimators=10 min_imp_dec=0.00025 max_features=0.1 learning_rate=0.5 loss=exponential	n_estimators=50 learning_rate=0.1 gamma=0.4 max_depth=3 subsample=0.2	n_estimators=100 min_imp_dec=0.00325 max_features=sqrt criterion=gini

Application Design

Weebly hosts the web page that contains HTML and JavaScript in it. However, it doesn't let me upload my Python code to run the machine learning model and make predictions. Therefore, I needed an additional server to host my server-side application code.

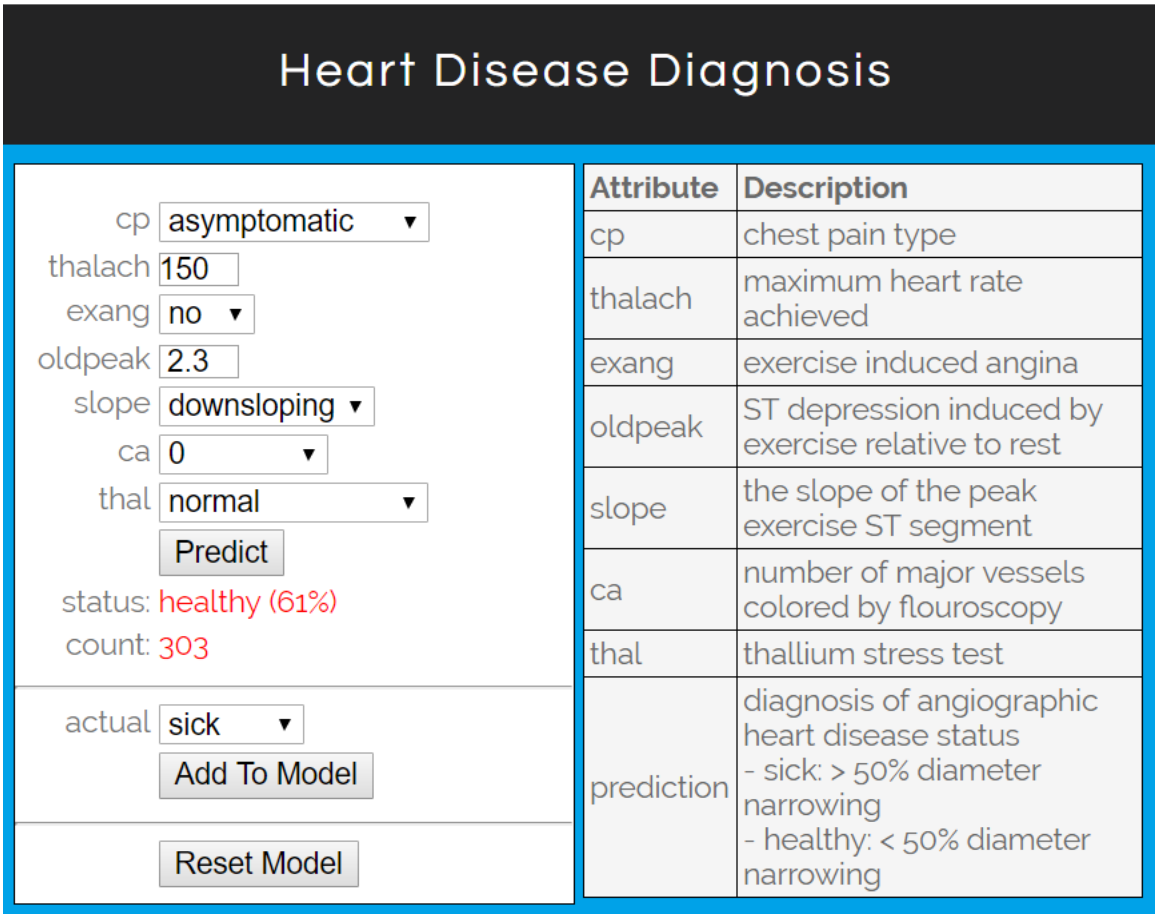
Here is a snapshot my overall architecture and design:



Application Development

The web page had around 240 lines of HTML and 60 lines of JavaScript to pass the user input to the machine learning program through the web service. The machine learning code in Python had about 100 lines, while the web service code had only 20 lines. The total number of lines of code was 420 for the whole application including client-side, server-side, and web service combined.

Here is a snapshot of the web page:



Discussion

The original heart disease dataset from the UCI Machine Learning Repository had 13 attributes for each patient record. I ran a feature selection process by using the Select Attributes tab in Weka to find out which attributes contributed to machine learning models significantly. I applied several different methods, and all of them consistently picked the same 7 attributes as the most important ones. Since the remaining 6 attributes didn't have a strong contribution, I removed those attributes, and created a new heart disease dataset.

The Penn Machine Learning Benchmark (PMLB) research paper analyzed 13 state-of-the-art machine learning algorithms on 165 publicly available classification datasets with 5.5 million parameter evaluations to provide recommendations to current researchers. Tree-based ensemble algorithms such as Gradient Tree Boosting, and Random Forest performed as the top two algorithms on most of the datasets.

When I downloaded the PMLB toolset and checked the benchmark values for the heart disease dataset, I noticed that they have added the XGBoost classifier as the 14th algorithm into their test results even though it wasn't mentioned in their research paper. Since XGBoost was the most popular tree-based ensemble algorithm with its speed and accuracy, I decided to run the PMLB evaluation tools on the new dataset after feature selection for the following three algorithms with 10-fold stratified cross validation: Gradient Tree Boosting, XGBoost, and Random Forest.

Even though the results were very close to each other, the XGBoost algorithm was the top performer with an 86% balanced accuracy score on the new dataset. It was also the fastest

algorithm that completed each evaluation in 0.58 seconds. So, I picked the XGBoost algorithm as the classifier with the identified parameter settings to train my model.

First, I developed the machine learning program in Python by using the PMLB toolset. It was able to diagnose a new patient as sick or healthy based on the model generated by the training data. However, it wouldn't be practical to distribute such an application to people in the medical field as it would be too specific to my development environment. That's when I decided to turn it into a web application for anyone to run from any device.

Finally, I built a web application in HTML and JavaScript to interact with the physician and send that information to the machine learning program through a web service. The user can enter a new patient's heart data and click a button to predict whether the patient is healthy or sick. The physician can correct the prediction with the actual health status of the patient and add the new record to the model to improve its performance as it grows with additional data.

Conclusion

Bioinformatics is an exponentially growing field that combines the power of math and science through machine learning. There are many biomedical classification problems including disease diagnosis, which rely on machine learning techniques. Cardiovascular diseases are the leading cause of death in the world, and the UCI Machine Learning Repository offers a heart disease dataset of 303 patients for research purposes.

The original dataset contained 13 attributes, but it wasn't clear whether all of them were relevant for a machine learning model. The feature selection process dropped the number of attributes from 13 to 7. Even though it didn't improve the accuracy score significantly, it provided a smaller set of relevant attributes only, and simplified the user interface for the medical personnel to enter new patient data.

Expert advice was the key to success during the process of feature selection. Since the data was considerably old as being from 1990, some of the 13 attributes selected at the time were not relevant anymore based on the latest developments in cardiology. Dr. Nafiz Kiciman, cardiologist at UCI, noticed those features even before I ran the feature selection tool in Weka. The tool just confirmed the suggestions and proved that it was a good dataset to identify irrelevant attributes successfully. However, if the tool generated contradictory results, expert advice would override them as it could be an outcome of wrong data or a small dataset.

PMLB research paper and toolset helped me to compare the performance of algorithms on the new dataset after feature selection. I strongly recommend other data scientists to read the findings in this paper and utilize the provided evaluation tools on their own datasets. This

approach also gave me more time to work on application development as their tools helped me to speed up the comparison process.

Developing the machine learning program in Python by using the XGBoost classifier on the new dataset was a straightforward task as there are readily available tools such as PMLB and libraries such as scikit-learn. However, letting other users connect to this program through the web was challenging. It required the development of a web page, and a web service layer to communicate with the machine learning program in the back-end. The user interface not only lets the medical personnel predict the health status of a new patient, but also lets them correct the prediction and add the new patient data into the model. This process will improve the accuracy of the model as it grows with more patient data over time.

This project provides the basic framework to run a machine learning program on a server and to communicate with it from a web page through a web service in the middle. Even though it is specific to the heart disease dataset, it can easily be modified to handle any dataset by replacing the training file and the accompanying user interface items.

Please feel free to run the application on any computer or mobile device at http://heartdx.weebly.com.

Further Research

The heart disease dataset is old as it's from 1990. It would be great if we could have more recent data. It only had 303 patient records in it. Having more data would improve the accuracy. The old dataset had a set of old attributes, and some of them turned out to be irrelevant and got removed as part of feature selection. A new medical study may introduce new and more relevant attributes based on the innovations and improvements in the last 30 years.

Acknowledgments

I'd like to thank *Nafiz Kiciman, MD, Division Chief of Pediatric Cardiology Faculty at UCI*, for his expert advice. I also thank *Ms. Cheryl Johnson*, my science teacher, for reviewing my work. Finally, I thank my father for helping me deploy the application on a public server.

References

- Anaconda, 2019, https://www.anaconda.com
- Centers for Disease Control and Prevention, 2017, https://www.cdc.gov/heartdisease/facts.htm
- Machine Learning Mastery, 2019, https://machinelearningmastery.com
- National Institute of Health, 2018, https://www.nhlbi.nih.gov/health-topics/coronary-heart-disease
- Olson, La Cava, Orzechowski, Urbanowicz, and Moore, PMLB: a large benchmark suite for machine learning evaluation and comparison, 2017, https://biodatamining.biomedcentral.com/articles/10.1186/s13040-017-0154-4
- Towards Data Science, Accuracy, Precision, Recall or F1, 2018, https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9
- UCI Machine Learning Repository, Heart Disease Data Set, 2018, https://archive.ics.uci.edu/ml/datasets/heart+Disease
- Weka, Machine Learning at the University of Waikato, 2018, https://www.cs.waikato.ac.nz/ml/index.html
- Wikipedia, Machine Learning, 2018, https://en.wikipedia.org/wiki/Machine_learning
- World Health Organization, 2017, https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)