Predicting Heart Failure using Ensemble Learning

Stéphane Collot[a*], Yannick Le Cacheux[b], Rishikesh Kulkarni[b]

[a] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[b] School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA
[*] Corresponding author: Present Address: 1065 Hampton Street, Atlanta, GA, USA Email address: stephane.collot@gatech.edu

Keywords: healthcare informatics, prediction, ensemble learning, heart failure, Spark, bagging, logistic regression, random forests

Abstract

Objective: our objective in this paper is to be able to automatically predict whether or not a given patient is likely to suffer from heart failure (Congestive Heart Failure) in the near future given his medical history.

Methods: to achieve this, we use medical records of more than 10,000 patients from the ExactData dataset to construct features based on Diagnoses, Risk Factors, Medication History, and Laboratory Test history in a five-year window. Then, we train classifiers using ensemble machine learning methods, mainly bagging with logistic regression and random forests. We then use k-fold cross-validation to evaluate performance and choose classifier parameters using the ML pipeline framework in Spark.

Results: we were able to achieve an accuracy of over 98% in a testing set consisting of 50% of individuals who did not suffer from this disease and 50% of individuals who were diagnosed with heart failure, while maintaining a low false negative rate.

Conclusion: we have excellent results with the confusion matrix; we sit back and discuss them by considering potential bias from the data set.

INTRODUCTION

In recent years, an increasing interest has been devoted to the use of big data analysis in the field of healthcare to improve the quality of care provided to patients or to advance medical research. An increase in the amount of digital medical data available has made it easier for researchers to analyze it and make predictions, as evidenced by the growing number of articles related to this topic. [1]

In this paper, we present our methodology and the results we obtained trying to make a prediction of the likelihood of a specific patient to suffer from heart failure in the near future based on the analysis of his medical record.

We decided to focus on heart failure for three main reasons: first, heart failure is a very common and deadly disease: approximately 5 to 10% of the population over 65 years old in developed countries suffer from heart failure; more specifically, an estimated 5.1 million persons are suffering from this disease in the US alone, and heart failure is involved in more than 10% of the total number of deaths occurring in the US. Its estimated cost to the country is more than $32 billion a year. Second, heart failure can be at least partly predicted thanks to various annunciator symptoms, such as diabetes, hypertension or high cholesterol, which makes it an interesting topic for predictive modeling. Finally, prediction of heart failure is useful since early prevention with specific recommendations for patients at risk can lead to significantly decreased risk [4].

BACKGROUND

There is already plenty of literature in the medical field on how to detect congestive heart failure risks in advance - for example [6] or [7] - as well as the key factors to monitor, such as coronary heart disease, high blood pressure or diabetes. However, these approaches are designed to help a physician to manually check for heart failure risks, and do not rely on recent progress in machine learning and predictive analysis.

Due to the importance of heart failure - cf. previous section, a number of more data-oriented research papers related to the prediction of heart failure have also been published, for example [5]:

this paper describes how to use natural language processing and predictive modeling to draw conclusions on risk factors for heart disease.
Contrary to these studies, we are here trying to create a classifier able to directly predict whether a patient is at risk based on an automated analysis of digital medical data about patients recorded over a period of time of about a decade.

We used basic machine learning algorithms from Spark MLLib and implemented ensemble learning algorithms that work with the ML pipeline to facilitate cross-validation and parameter tuning. In particular, we implemented a Bagged Logistic Regression classifier and a wrapper for the Random Forest algorithm in MLLib to work with ML pipeline.

METHODS Data preparation
The features we consider to make our predictions are the history of diagnostics, medications, laboratory results and vital features (e.g. BMI) of each patient. To train and test our classifiers, each patient needs to have a vector of features of the same dimension as the feature vectors of all the other patients. We therefore consider each possible information about a patient as an element of this vector of features. For example, If we have d different possible diagnostics, m different possible medications, l different possible laboratory results and v different vital features in our dataset, then our feature vector will be of dimension d + m + l + v. The elements of this vector can be either binary, e.g. 0 or 1 depending on whether the patients was diagnosed with a particular icd9 code in the past, or continuous, for example to correspond to the value - or the average of the values - of a laboratory result or a vital feature if there is information about this value.

In addition, we have to distinguish two categories of patients: the patients who were diagnosed with the icd9 code we study (labeled as "YES" or 1), and the "healthy" patients, i.e. those who were not diagnosed with this disease (labeled as "NO" or 0). We simply check for the existence of such a diagnostic in the medical records of a specific patient to label this patient.

Furthermore, our results would obviously be extremely biased if we considered medical records subsequent to the diagnosis of the disease, for example medications taken because of this diagnosis. We therefore discard any medical record posterior to a date corresponding to a year before the diagnosis in case patients. To make sure that the healthy patients did not develop the disease immediately after the end of our observation window, we only consider the first half of this window - i.e. a time period of approximately 5 years - and discard the second half of healthy patients.

Finally, the percentage of patients who suffered from heart failure is relatively low in our dataset - around 7%. A classifier outputting "NO" for every patient would therefore have a 93% accuracy, even though it wouldn't really be able to make predictions. Worse, it would classify as healthy patients who have actually very high risk of contracting the disease. In other words, it would output a lot of false negatives, which is exactly the opposite of what we want to achieve. To prevent this, we randomly discard most of the healthy patients from our dataset to have approximately 50% case patients and 50% healthy patients in our training sets.

Algorithms used

Our classifiers are ensembles of Logistic Regression models or Decision Trees. Both Logistic Regression and Decision Trees are known to have high variance and low bias. They are suited to build ensembles. We therefore used ensemble learning by the way of Bagging for Logistic Regression models and growing Random Forests of Decision Trees to reduce variance and improve accuracy.

Bagging is implemented as follows: b resampled training sets are created from the original training set using resampling, where b is the bag size. Resampling involves drawing random samples of a fixed size from the original dataset until the resampled set reaches a desired size. The size of the resampled set is a parameter that can be tuned for performance. The resampling rate determines how many samples are drawn and what the size of each sample is. The resampling process is repeated until we have as many resampled sets as the size of the bag. Models are learned by training a classifier on each of the resampled training sets. A bag of these models constitutes the final classifier. To make a prediction with the final classifier, predictions from each model are tabulated and the label that gets the most votes is chosen as the final label.

Random Forests are ensembles of decision trees. Like bagging, resampling is used to build multiple training sets from the original dataset. A decision tree model is learned by training the decision tree classifier on the resampled sets. At each node in the tree, a random subset of the features is selected and the best split on this subset is used to split the node. The size of the random subset is same for all trees in the forest. Each decision tree represents a model and the Random Forest is a collection of these decision tree models. To make a prediction, predictions from each decision tree are tabulated and the label that gets the most votes is chosen as the final label.

Bootstrap resampling is a sampling method where, after each sample is drawn, it is returned to the dataset, such that each sample drawn is independent of all others.

Stratified sampling is a sampling method where samples are drawn from each stratum of the data so that all subpopulations of the data have proportional representation in the resampling.

We trained a bag of basic Logistic Regression models and a Random Forest for classification. We set the resampled set size to be the same size as the original training set and used a sample size of 0.001. We used **Stratified Bootstrap resampling** to build the resampled sets. For our classifier, the strata were the case and control patients. We used a 1:1 sampling proportion to build the resampled training sets. We resampled the case patient data until we had half as many samples as the original data set and then resampled the control patient data to fill the other half of the resampled set.

Cross-Validation and ML Pipeline

ML pipeline offers an easy interface for k-fold cross-validation and parameter tuning. The CrossValidator class can be instantiated with a parameter grid to iterate over and choose the combination of parameters that give the best results on cross-validation.

The algorithm is run for every combination of the parameters in the grid on each fold of the dataset. Then the parameters that give the best performance are stored as the best model. This model can then be used to make predictions on the testing set.

For bagging, we experimented with parameters like regularizer, bag size, maximum iterations etc. For Random Forest, we experimented with type of learning, feature subset strategy, number of trees etc.

Table 1. Parameter Grid of ML CrossValidator

Parameter 1	Parameter 2	…..	Parameter m
value 1	value 1	…..	value 1
value 2	value 2	…..	value 2
…..	…..	…..	…..
value n1	value n2	…..	value nm

RESULTS Experiment Setup

We use a dataset from ExactData which covers a time period of 10 years from 2005 to 2015. This data contains different tables; from this data, we used: 772.189 laboratory results, 313.921 diagnostics, 120.953 prescription medications, 269.301 vital signs. In total there are 10.460 patients, including:

- 4.712 patients diagnosed with Diseases Of The Circulatory System: ICD9 codes within [390.0; 495.0]
- 879 patients diagnosed with Congestive heart failure: ICD9 code equal to 428.0

The feature construction presented before - data preparation subsection in the methods used section - produces a 213-dimensional feature vector consisting of 60 different medications, 52 different diagnostics, 100 different laboratory tests and 1 vital sign - here Body Mass Index (or BMI).

Table 2. Features

Feature Group	Features	Type	Example	Aggregation
Diagnosis	ICD9 Code	Categorical	498.00	Count
Medication	Medication Name	Categorical	Aspirin	Count
Lab	Loinc Code	Categorical	14647-2	Count

Lab	Value	Numerical	40.0	Mean
Vital Signs	BMI	Numerical	30.5	Mean

For the technology stack, we used Spark version 1.3.0 with its Scala API. In particular, we used Spark MLlib containing common machine learning algorithms. We also used Spark ML which is a new high-level API for Spark MLlib: it defines a pipeline containing standard component like Cross-validation or metrics. More precisely, we modified the implementation of the Logistic Regression to add bagging, and we integrated the Random Forest algorithm within the pipeline.

We ran our code on a cluster of 3 machines m1.large with 2 virtual CPU and 7.5 GB of RAM each in Amazon Web Service.

We used 10-fold cross-validation for both Bagged Logistic regression and Random Forest. The parameter grid resulted in a total of 4 (Regularizer) x 5 (Bag Size) x 4 (Max Iterations) x 10 (Folds) = 800 models for Bagged Logistic regression and 1 (Training Strategy) x 4 (Feature Subset Strategy) x 6 (Number of Trees) x 4 (Max Depth) x 10 (Folds) = 960 models for the Random Forest.

Results with metrics

Table 3 and Table 4 present the confusion matrices for our two models run over a testing set. This testing set was in the same format as the one used for training, but contained records from different patients in order to be able to detect overfitting or other biases susceptible to artificially increase accuracy. As we can see, we have very few false negatives: 3 for Logistic Regression - approximately 3.5% of case patients - and 0 for Decision Trees. Therefore, our results meet our goal to minimize the number of false negatives. Figure 1 and Figure 2 display the Receiver Operating Characteristic (ROC) curves and the percentage of accuracy; here random forest seems slightly more accurate than logistic regression.

Table 3. Confusion Matrix for Logistic Regression - Bagging

		Prediction	
		0 Healthy	1 Case Patient
Actual	0 Healthy	True negative 73	False positive 3
	1 Case Patient	**False negative** **3**	True positive 83

Table 4. Confusion Matrix for Decision Tree - Random Forest

		Prediction	
		0 Healthy	1 Case Patient
	0 Healthy	True negative 74	False positive 2

Actual	1 Case Patient	**False negative** **0**	True positive 86

Figure 1. Receiver Operating Characteristic (ROC) curve for Logistic Regression – Bagging

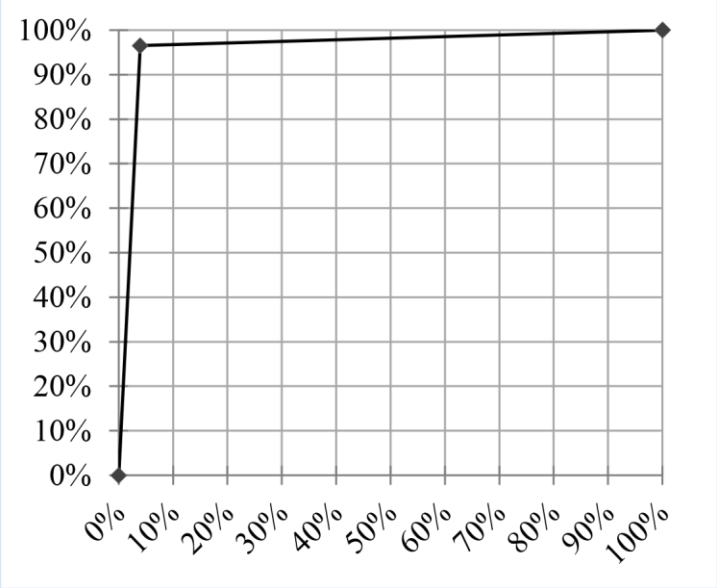
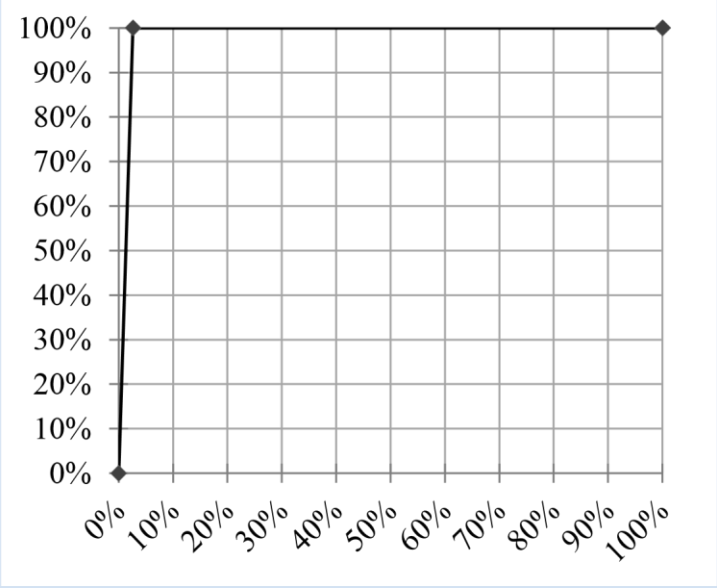


Figure 2. Receiver Operating Characteristic (ROC) curve for Decision Tree - Random Forest



DISCUSSION AND CONCLUSION

The results are surprisingly good when compared to the industry standard. To explain this, we can raise several hypotheses and concerns about the data used:

- The data set from ExactData we used is not real data from real patients and hospitals. ExactData generates data for test purposes and try to keep consistency and realism. The advantage is that it is very easy to access the data from a legal, security and privacy points of view. But the data set could contain some patterns, which could be learned by our algorithms, resulting in an artificially high accuracy.
- We don't have data before the observation windows. So for example, we could have patients diagnosed with heart failure before the beginning of the data set, and therefore later laboratory results and prescription medication could be very much correlated with the disease.
- There may be other medical features that are obviously correlated with heart failure before the diagnosis from a medical point of view. Therefore, we might need medical expertise to help confirm this hypothesis and identify potentially biases in our data.

We can finally highlight that our algorithms are also highly scalable since each decision tree and each logistic regression can be run in parallel on different CPUs, resulting in a significant speedup on clusters with a high number of nodes.

REFERENCES

1 Deborah Helen Selma. Paradigm of Prediction: Predictive Analytics to Prevent Congestive Heart Failure. In: *OJNI Volume 18, Number 2*.

2 Leo Breiman. Bagging Predictors. In: *Machine Learning, 24, 123–140.* 1996.

3 Eric Bauer, Ron Kohavi. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. In: *Machine Learning, July 1999, Volume 36, Issue 1-2, pp 105-139.*

4 Heart failure fact sheet from the CDC. In: *http://www.cdc.gov/dhdsp/data_statistics/fact_sheets/docs/fs_heart_failure.pdf.*

5 IBM Predictive Analytics to Detect Patients at Risk for Heart Failure.

6 American Heart Association. Understand your risk for heart failure.2012.

7 Dungan, K., Binkley, P., Nagaraja, H., Schuster, D., & Osei, K. The effect of glycaemic control and glycaemic variability on mortality in patients hospitalized with congestive heart failure. 2011.

Response to comments

The major concern voiced by many reviewers was that the sampling method used to construct the datasets led to inaccurate and biased models. The reason we chose to balance the classes was so that the classifier learned a model which had a positive bias towards identifying heart failure. While this may seem counter-intuitive, it is in fact a loss-minimization strategy, the loss here being the number of *false negatives*. We biased the classifier so it would learn to predict positive instances even if it meant that it would have a higher rate of false positives, given how important the prediction of a major illness like heart failure is.

Another concern voiced was that it was not clear how we split the data into training and testing sets and whether the cross validation stats cited were for the entirety of the dataset. We divided the case patients in an 80:20 split, combined the 80% with an equal number of controls to build the training set. The cross validation was done on only this set to measure accuracy and tune parameters. The other 20% cases were combined with the rest of the controls to form the testing set on which we performed our final evaluation and plotted the ROC curves.

Another concern voiced was the mention of patients with “diseases of the circulatory system” and whether we counted these as cases. We did not count these as cases. Our cases were only patients diagnosed with Congestive Heart Failure (CHF – 498.0).

Yet another concern was feature selection. We did not do any feature selection step because our features’ dimensionality was manageable and we did not see the need to do any kind of selection or normalization as our performance demonstrates.

We did run our project on a cluster as described in our presentation but did not see any significant speed up, given that after cleaning and filtering our data it was not larger than a couple gigabytes. The performance did go up marginally but it was not significant enough.