




PREDICTING HEART ATTACK



Prateek Agrawal-A20358932
Ajay Raghunathan-A20349314

Table of Contents

Problem Statement	2
Proposed Solution	2
Iterative Dichotomiser 3 (ID3)	2
Neural Networks	4
Support Vector Machine	6
Implementation details	8
Result and Discussion	9
References	11

Problem Statement

In today's world cardiovascular diseases or heart diseases are the primary cause of the increasing death-rate worldwide. Our aim in this project relies on predicting the presence and absence of different classes of the predefined category of heart diseases. We would be using machine learning approaches for building the model for the prediction of the heart diseases. There is the paper we use ID3, Neural Network and Support vector machine to predict the possibility of heart attach for given features of the patients. We use the approach of PCA to reduce the noise in the data and also algorithm boosting to improve the accuracy of the classifier. Data set that we will be used is available on the UCI data repository containing 14 features such as age, sex, type of chest pain etc.

Proposed Solution

We use three different algorithms to classify the given data set. All the algorithms are well known for its ability to classify complex data. Data from the UCI data repository is divided into training and testing data. Missing data are imputed in each feature using the mode. Following are the step by step explanation of the all the algorithm used.

Iterative Dichotomiser 3 (ID3)

ID3 algorithm is a decision tree algorithm which is a precursor to the C4.5 algorithm. The main applications of the ID3 lies in the machine learning and Natural Language processing. The main ideas behind ID3 algorithm are as follow :

- Each non-leaf node is an input feature and each originating from the node is a particular value of the input feature.
- In this algorithm each node of the tree should correspond to a feature with the highest information gain (or the lowest entropy). This is because we would always want to predict the output using the smallest possible questions.

- We have used Entropy and Information gain to determine how informative is a feature. Entropy can be defined as the measure of uncertainty in a system. Below is the formulae for the entropy.

$$H = - \sum_i p_i (\log_2 p_i)$$

The ID3 algorithm starts with the original set of attributes X as the root node. We first check If all the labels originally are '1' or '0' for the whole dataset. If they are, then only a single node is created with the label as that of the labels in the dataset. If the number of attributes at this stage is 0 then we would assign the class label as the one with most number of classes in the data. If this condition does not hold then we have implemented an iterative solution which begins originally with all the features ('X') from the dataset. Now, in order to find the best attribute to be split on we have used the concept of entropy and information gain. Below is the formulae for the entropy and the information gain.

$$E(data) = - \sum_i p(x) \log(x)$$

Here 'i' belongs to all the classes in the dataset. The Entropy is calculated before and after the split on the feature. The information gain is calculated as follows.

$$IG(Attr) = E(data) - \sum_t p(t)E(t)$$

Here 't' belongs to the subsets that are created after splitting the data on the feature. At this stage we have computed the best feature to be split at the current node. Now, for each possible value of the current attribute a child node is created and the whole process is followed again, but this time with features as the { Total Features - Current Feature }

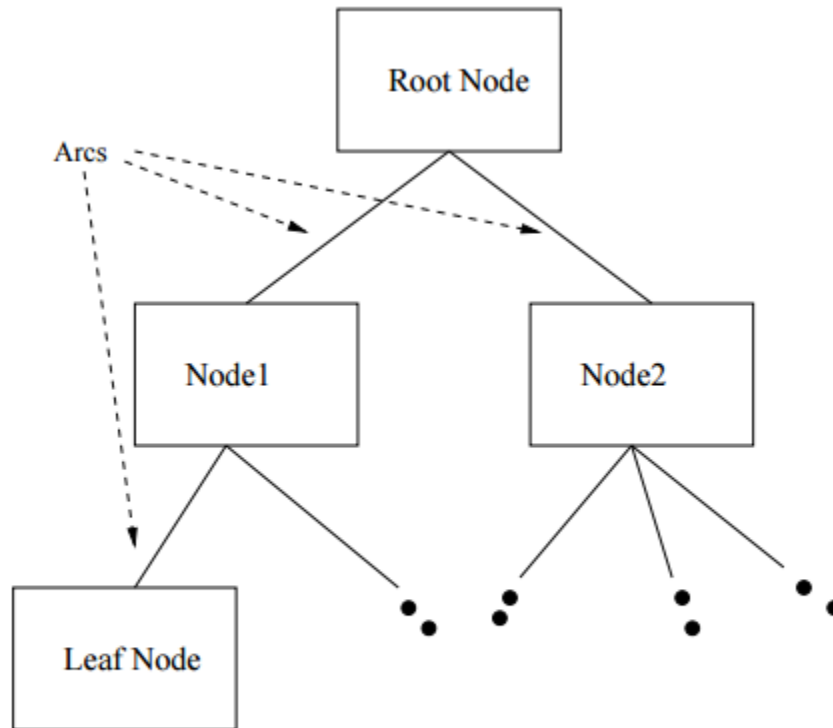


Figure 1: Basic decision tree structure

Neural Networks

Neural networks are more successful as a classifier. We use feed forward approach this problem. There are two layers in the model. Hidden layer has H unit and output layer has one unit. Hidden layer has a weight of W for each unit and output layer has weight V for each unit. Following diagram better explains the concept of Neural Network.

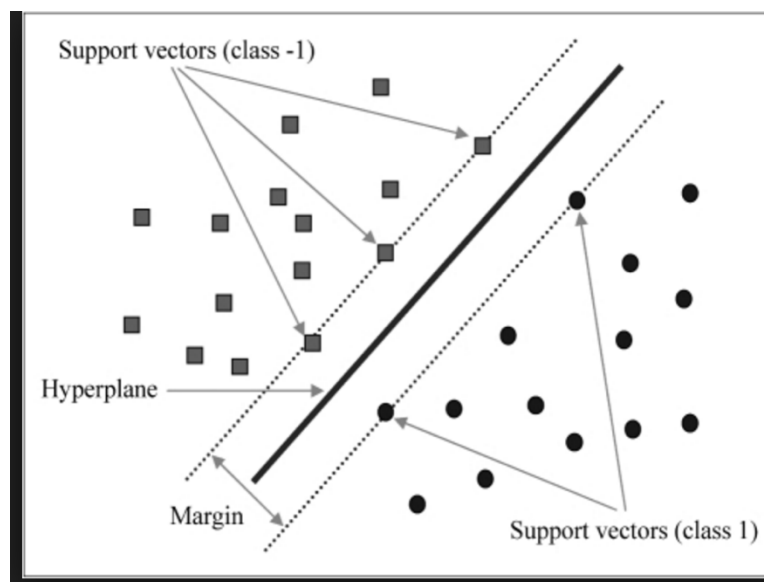
$$W_j = W_j - \eta \sum_{i=1}^m \left(\sum_{l=1}^k (h\Phi_j(Z_i) - \text{indicator}(y = j)) * Z_{ij} * (1 - Z_{ij}) \right) . X(i)$$

Following is the algorithm followed.

- Start by guessing V_j and W_j .
- Compute Z (output of hidden layer) and Y (output of output layer) using V_j and W_j .
- Update V and W using the computer value in the above step.
- Follow the above steps until the objective (log likelihood) stops changing.

Support Vector Machine

The basic approach that we follow here is drawing a line between different data points discriminating one class from other. SVM is different from Discriminative learning in the sense that it generalizes better as it tries to draw a line for the clusters are far as possible.



In the above diagram the SVM tries to maximize the margins and identify the support vectors. These support vectors will be used for the purpose of classifying a new example.

As shown in the above diagram our main aim here is to identify the support vectors and draw a hyper plane to discriminate the 2 labeled classes. The line drawn to discriminate the classes has the following equation.

$$d(X) = Wt * X + W^o$$

Here W^o is the bias or the distance from the origin. ' Wt ' is the normal to the discriminating line. Our main aim here is to maximize the geometrical margin shown the above diagram.

To do this we minimize the primal problem given below.

$$Lp = \frac{1}{2} \text{tranpose}(W) * W$$

Subjected to the constraint that $y(i)(Wt * X(i) + W^o) > 1$

We redefine $Lp = \frac{1}{2} \text{tranpose}(W) * W - \sum_{i=1}^m \alpha_i (y(i)(Wt * X(i) + W^o) - 1)$

Here α_i is the LaGrange multiplier and is defines as the penalty we give for each wrong classification.

Next we minimize Lp with respect to W and W^o , and then maximize the result with respect to α_i . Taking partial derivative of Lp with respect to W and W^o we get the following results.

$$W = \sum_{i=1}^m \alpha_i * y_i * x_i$$

$$\sum_{i=1}^m \alpha_i * y_i = 0$$

Substituting W is the equation of Lp we get,

$$Ld = -\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i * \alpha_j * y_i * y_j * \text{treanpose}(X_i) * X_j + \sum_{i=1}^m \alpha_i$$

Now is maximize this Ld with respect to α_i using the Qp solver. After solving this we get all the values of the α_i . Using the value to α_i we identify all the support vectors. All the other points should have a value equal to zero but support vectors would be greater than zero. Using this α_i we can compute the value of W_s .

Same can be used to compute the W^o with the following formulae.

$$W^o = \frac{1}{\#SV_s} \sum_{X \in SV_s} (Y_i - \text{transpose}(W) * X_i)$$

Using these values, we classify a new given feature vector.

If: $Wt * X + W^o > 0$, we classify as class 1 else class 0

The algorithm discussed above is known as hard margin. We can also give some slack to the classifier and decrease the penalty it gives every time a wrong classification is done. For doing this our basis equation changes to:

$$Lp = \frac{1}{2} \text{transpose}(W) \cdot W + C \sum_{i=1}^m S_i$$

Here C is the weight we give to each slack variable and S_i is the slack variable for each example.

For classifying data where the examples are not separable with a linear function we use the same approach but in addition add the kernel trick so that we do not have to map the examples to the higher dimension.

$$X_i * X_j = \phi(X_i) * \phi(X_j) = k(X_i, X_j)$$

Here the function k is the kernel function and gives us the similarity between two vectors. Several kernel functions available are radial, linear, polynomial, Gaussian etc. In this report we have used Polynomial and Gaussian.

Implementation details

Data used for this project is available at UCI data repository. It has 13 features and 303 rows. The missing values are imputed in the data with the mode of the feature of the missing value.

- Principal Component Analysis is used to filter the noise from the data and convert into smaller subsets of features.
- Norm () function is used to normalize the data with the following formulae.

$$\mu_j = \frac{1}{m} \sum_{i=1}^m X_{ij}$$

$$\sigma_{2j} = \frac{1}{m} \sum_{i=1}^m (X_{ij} - \mu_j)^2$$

$$X_{ij} = (X_{ij} - \mu_j) / \sigma_j$$

- We apply PCA to decrease the noise in the data and reduce the number of features. To implement this, we use the computation of the maximum d Eigen value and vector.

$$\sum^* W = \lambda W$$

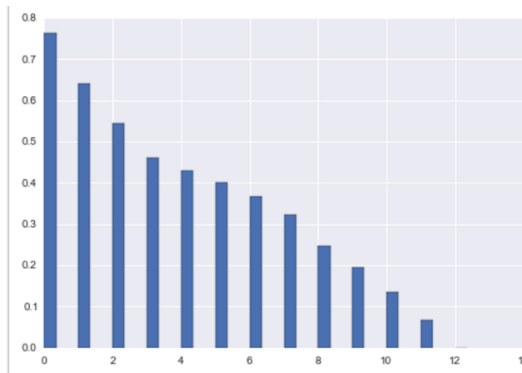
Here Σ is the covariance matrix of the data and 'W' are the Eigen vector and lambda is the Eigen values. We choose the first d Eigen vector which has variance loose less than 10%. We then compute the new data Z using the following formulae.

$$Z_i = W^t * X_i$$

- Data matrix obtained from above is used to as an input to three algorithm stated earlier.
- To boost the accuracy of the algorithm we use bag boosting approach were we take prediction from all the algorithms and find the mode of it as the actual prediction. For example, If the prediction is {1,1,0} from the three algorithms we choose 1 as the actual prediction.
- To understand the data being used we use exploratory data analysis and disregard some of the features that have a very high or very low variance, as these features won't help in the prediction.
- We than compute the confusion matrix, Precision and Recall to asses our classifier.
- Parameters in each algorithm are compute on the basis of the maximum accuracy provided by them.

Result and Discussion

- After applying the Principal Component Analysis, we found that all the 14 feature used in the data shows variance lose less then 10 percent. So, we just use the normalized data as the input to the algorithms.



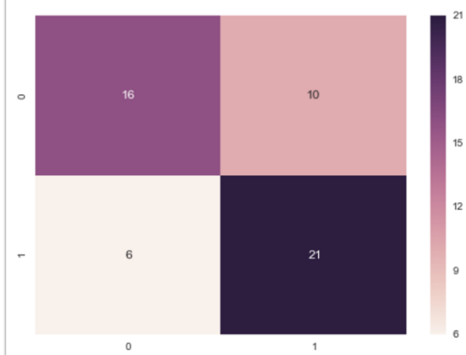
- Data set is divide in 250 rows for training and 53 rows for testing.
- 70 percent of the testing data was classified correctly by the ID3 algorithm.

Accuracy of the Decision Tree is: 69.8113207547
 Below are the various statistics for the Decision Tree
 =====

	precision	recall	f1-score	support
0.0	0.73	0.62	0.67	26
1.0	0.68	0.78	0.72	27
avg / total	0.70	0.70	0.70	53

 =====

Confusion Matrix
 =====

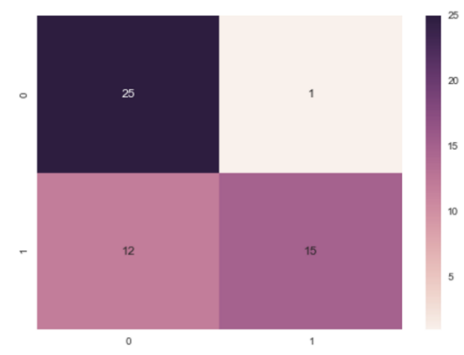


Accuracy of the SVM Model is: 75.4716981132
 Below are the various statistics for the SVM Model
 =====

	precision	recall	f1-score	support
0.0	0.68	0.96	0.79	26
1.0	0.94	0.56	0.70	27
avg / total	0.81	0.75	0.74	53

 =====

Confusion Matrix
 =====



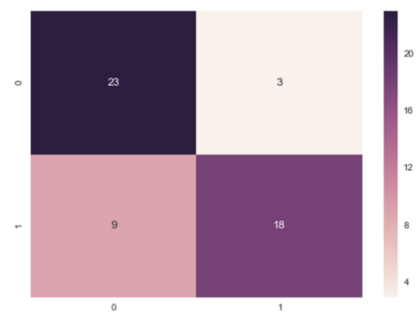
- 75 percent of the data was classified correctly by Gaussian SVM with value of parameter Epsilon as 0.0001 and slack variables as 5.
- 78 percent of the data was classified correctly by neural network using two-layer network with Number of hidden units as 50 and learning rate as 0.001.

Accuracy of the Neural Network Model is: 77.358490566
 Below are the various statistics for the Neural Network Model
 =====

	precision	recall	f1-score	support
0.0	0.72	0.88	0.79	26
1.0	0.86	0.67	0.75	27
avg / total	0.79	0.77	0.77	53

 =====

Confusion Matrix
 =====

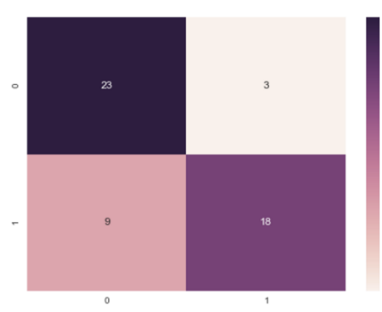


Accuracy of the Bagging approach is: 77.358490566
 Below are the various statistics for the Ensemble model
 =====

	precision	recall	f1-score	support
0.0	0.72	0.88	0.79	26
1.0	0.86	0.67	0.75	27
avg / total	0.79	0.77	0.77	53

 =====

Confusion Matrix
 =====



- Using the bag boosting approach with accuracy remains same as 78 percent.

References

- http://en.wikipedia.org/wiki/Decision_tree_learning
- <https://github.com/scikitlearn/scikitlearn/blob/51a765a/sklearn/tree/tree.py#L482>
- Element of statistical learnin
- Data : <https://archive.ics.uci.edu/ml/datasets/Heart+Disease>
- Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J., Sandhu, S., Guppy, K., Lee, S., & Froelicher, V. (1989). International application of a new probability algorithm for the diagnosis of coronary artery disease. *American Journal of Cardiology*, 64,304--310.
- David W. Aha & Dennis Kibler. "Instance-based prediction of heart-disease presence with the Cleveland database."
- Gennari, J.H., Langley, P, & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, 40, 11--61.
http://ijircce.com/upload/2014/november/37P_Predicting.pdf