Predicting Heart Disease by Means of Associative Classification

Hisham Ogbah, Abdallah Alashqur and Hazem Qattous

Applied Science Private University, Faculty of Information Technology, Amman, Jordan

Abstract
In this age of "Internet of Things" and "Smart Cities" the importance of the field of data mining has increased significantly. Data mining is needed because of the huge amounts of digital data that is being generated globally on an hourly basis (referred to as Big Data). Data mining provides methods and techniques to analyze large repositories or fast-moving streams of data at a massive scale. The data mining process extracts useful information and knowledge, from big data, that benefit the decision maker. Classification is one of the data mining techniques used to analyze existing data and predict classes of new data. Association rule mining is another data mining technique that is used to extract frequent patters and association rules relating pieces of data together. A distinguished classification technique called associative classification combines classification and association rule mining to predict class labels of data. Health care is one of the areas in which data mining is extremely useful. Mining medical and patient data can help in discovering patients' conditions and aid in the diagnosis process. In this paper we show how associative classification is applied in the prediction of heart disease, which is one of the most common types of diseases that affect people's lives.

Key words:
Data mining, Association rules mining, Associative classification, Mining health care data

1. Introduction and Background

Two popular data mining techniques are classification [1-5] and association rule mining [6-11]. Each one mines the data in its unique way. These two techniques can be combined in what is called associative classification [12, 13], where association rule mining is performed as a step that precedes classification. Classification in this case is actually constructed based on the association rules discovered in the previous step. In this paper we demonstrate how associative classification can be applied to mine health data to predict heart disease. A brief description of association rule mining and classification as used separately from each other is given below. Then we show how the two are combined to create associative classification. In this paper, we demonstrate how associative classification is can be applied to sample health care data in order to perform heart disease prediction.

1.1 Association Rule Mining

Association analysis refers to correlation that exists among a set of items that appear in a dataset. Associations may exist when data items appear together in a single event [7, 10]. As an example suppose there is a store that sells computers and is called CompStore. We want to know which items are frequently purchased together and how much the purchase of an item influences the existence of the other items in the same transaction. An example association rule is shown as follows.

$$buys\ (P, "laptop") \rightarrow buys\ (P, "camera")$$
$$[support = 3\%, confidence = 45\%],$$

Where *P* is a variable representing a person. A confidence of 45% indicates that if a person buys a laptop, there is a 45% chance that he/she will buy a camera as well. A 3% support indicates that 3% of all transactions being considered contain both laptop and camera simultaneously. In this example, only one predicate appears in the left hand side and in the right hand sides of the rule, which is the predicate *buys()*. In realistic situations, more than one predicate may appear in the left hand side or the right hand side as shown in the following example.
Assume a data mining system discovers the following association rule in the *CompStore* database.

$$age(P,"30..39") \wedge income(P,"50K..59K") \rightarrow buy(P, "laptop")$$
$$[support = 2\%, confidence = 60\%].$$

The rule indicates that 2% of the *CompStore* clients who are in the age range of 30 to 39 years old and have income of $50,000 to $59,000 have purchased a laptop at *CompStore*. The confidence means that there is a 60% probability that a customer in this age range and income group will purchase a laptop. Note that this is an association rule that involves more than one attribute or predicate (i.e., *age(), income(), and buys*()).

Typically, association rules are discarded as uninteresting if they don't satisfy both a minimum support (called *minsup*) threshold and a minimum confidence (called minconf) threshold [7]. The values of *minsup* and *minconf* are normally parameters provided to the data mining tool by the user.

Manuscript received September 5, 2016
Manuscript revised September 20, 2016

1.2 Classification

In many applications, we need to classify data stored in a dataset. Each tuple in the dataset needs to be mapped to a certain class. One column in the data set (which is referred to as class label) stores the class name. Normally there are only a few classes. Many tuples can be mapped to one class. The goal of classification software is to be able to classify new tuples whose classes have not yet been known [7,10]. In order for the system to be able to do this in an automated fashion, it has to have the necessary knowledge that enables it to classify newly inserted tuples. This can be achieved by supplying the data mining system with a dataset whose tuples are already classified. The system then learns (or derives) the criteria used to identify the classes of tuples in the dataset. Once the system learns the classification criteria, it can use these criteria in the future to predict the classification of any new unclassified tuples [7, 12, 13].

In data mining terminology, these classification criteria are sometimes called "classification model." The dataset from which the system learns the classification model, is referred to as training set because it is used to train the system. The classification process goes through two steps. In the first step, the learning step, the classification model is extracted from the training set. Once the system extracts the classification model, and after passing through some testing, the system becomes prepared for the second step. In the second step, which can be called the application step, the system applies the classification model that it learned to new unclassified tuples in order to predict their classes. The above two data mining techniques can be combined to create what is called associative classifier.

The main objective of this paper is to show a case study demonstrating how associative classification can be applied to health care data. The remainder of this paper is organized as follows. In Section 2 we describe in some detail how associative classification is performed based on [13]. In Section 3 we demonstrate how associative classification can straight forwardly be applied to medical data to predict heart disease. Conclusions are given in Section 4.

2. Associative classification

By combining classification and association rule mining, a new accurate classifier can be built to give a new way to construct classifiers and help to solve a number of problems that exist in current classification systems. The main idea is to search for strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels [7].

Many experimental results have shown that this technique achieves higher accuracy than traditional classification techniques. One of those traditional classification techniques is the algorithm C4.5 [14]. What's more, many interesting rules found by associative classification algorithms cannot be discovered by traditional classification methods [13]. Associative classification algorithms produce classification models that are easy to understand by end-user because models are constructed based on such interesting and interpretable rules. Their high accuracy and understandability have made associative classification very popular in real world systems, such as medical diagnoses.

The main objective is to generate the complete set of Class Association Rules (CARs) that satisfy the user-specified minsup and minconf constraints, and to build a classifier form these CARs [13]. Note that the number of rules produced is a function of the minsup and the minconf thresholds. One of the earliest and simplest algorithms for associative classification is CBA (Classification Based on Association) [13]. It consists of two parts, a rule generator (called RG), which is based on Apriori algorithm for finding association rules, and a classifier builder (called CB). In this section we will explain both RG and CB in order to use them in the Section 3 to predict heart disease.

2.1 Rule Generator (RG) Algorithm

The RG algorithm is an extension of the Apriori algorithm [7]. The goal of this algorithm is to discover all classification rules (classrules) that have support above minsup. A classrule is of the form: <rulecond, y>, where rulecond represents the set of items, and y $\in$ Y where Y is a class label that represents a set of class names and y is a class name. The support count of the classrule is the number of tuples in the dataset D that contain the rulecond (called condSupport) and are labeled with y. Each classrule corresponds to a rule of the form: rulecond □ □ y.

Classrules that satisfy minsup are called frequent classrules, whereas classrules whose support are below minsup are called infrequent classrules.

For all the classrules that have identical rulecond, the classrule with the highest confidence is selected by RG as the possible rule (PR) representing this set of classrules. In case of more than one classrule with the same highest confidence, RG randomly selects one classrule. For example, assume we have the following two classrules that have the same rulecond.

1- < {(Attr1, 1), (Attr2, 1)}, (class: 1) >.
2- < {(Attr1, 1), (Attr2, 1)}, (class: 2) >.

Where Attr1 and Attr2 are attributes. Both rules have the same *rulecond*. Assume that support count of the *rulecond* is 3. Also assume that the support count of *classrule* Number 1 is 2, and the support count of *classrule* Number 2 is 1. Let the size of the dataset is |D| = 10. We can compute *confidence* by dividing the support count of *classrule* by the support count of *rulecond*. Therefore the

confidence of *classrule* No.1 is (2/3) = 66.7%. The confidence of *classrule* No.2 is (1/3) = 33.3%. If the *minconf* is set to 50%, then only one of the above two rules is selected, which is shown below.

(Attr1, 1), (Attr2, 1) $\rightarrow$(class, 1)
[sup = 20%, conf= 66.7%].

If the confidence of a rule is greater than *minconf*, the rule is said to be accurate. The set of class association rules that we need to preserve contains all the frequent and accurate PRs.

In the RG algorithm, *k-classrule* represent *classrule* whose *rulecond* has *k* items. Let L_k represents the set of frequent *k-classrules*. Each element of this set is of the following form:

<(*rulecond*, *condSupport*), (*y*, *ruleSupport*)>.

Let C_k denotes the set of candidate *k-classrules*. The RG algorithm is given in Figure 1.

```
1   L1 = {large or frequent 1-classrules};
2   CAR1 = GenerateRules(L1);
3   pCAR1 = Prune(CAR1);
4   for (k = 2; Lk-1 ≠ ∅; K++) do
5       Ck = GenerateCandidate(Lk-1);
6       for each data tuple t ∈ D do
7           Cd = ruleSubset(Ck, t);
8           for each candidate c ∈ Cd  do
9               c.condSupport++;
10              if t.class = c.class then c.ruleSupport++
11          end
12      end
13      Lk = {c ∈ Ck | c.ruleSupport  ≥ minsup};
14      CARk = GenerateRules(Lk);
15      pCARk = Prune(CARk);
16  end
17  CARs = ∪k CARk;
18  pCARs = ∪k pCARk;
```

Figure 1: The RG algorithm [13]

The first pass of the algorithm (lines 1-3) over the dataset determines the frequent 1-*classrules*. The *GenerateRules* function (line 2) generate a set of CARs, called CAR_1, from 1-*classrules* set. The function *Prune* is applied to CAR_k in the first pass and each subsequent pass. It uses the pruning method, which is based on the pessimistic error rate [14]. This method prunes a rule as follows: If rule's pessimistic error rate is higher than pessimistic error rate of the rule obtained by removing one condition from the conditions of the original rule, then that rule is pruned. This pruning can considerably reduce the number of rules generated.

After that, for each pass *k*, the algorithm performs the four main operations as follows. First, the *GenerateCandidate* function generates candidate *classrules* C_k , using the frequent *classrules* L_{K-1} found in the $(K-1)^{th}$ pass (line 5). The *GenerateCandidate* function behaves similar to the function called *Apriori-gen* in the Apriori algorithm as described in [15]. Second, lines 6-14 scan the dataset and compute the support of the candidates in C_k . It uses the *ruleSubset* function which takes a set of candidate *classrules* C_k and a data tuple t as input to find C_k in all the *classrules* whose *ruleconds* match t. Third , it finds the frequent *classrules* for the current pass *k*, namely L_k , and then generates CAR_k set using *GenerateRules* function (lines 15-16). Fourth, the *Prune* function is applied on CAR_k to perform reduction of the number of rules (line 17). Eventually, the RG algorithm produces the final set of class association rules in the set *CARs* and produces the rules after pruning in the set *pCARs*.

Below we show the application of the RG algorithm to a sample dataset. Table 1 shows a sample dataset, including 2 attributes A and B, and a class label C. Assume that *minsup* is given as 15%, and *minconf* is 60%.

Table 1. Sample Dataset

A	B	C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

The RG algorithm is used to generate, from the dataset, all class association rules that have support and confidence larger than or equal to the given thresholds, i.e., *minsup* and minconf. The RG algorithm generates all the frequent *classrules* by making multiple passes over the dataset. In the first pass, it counts the support of each individual *classrule* and determines whether it satisfies *minsup*. The outcome of this pass is demonstrated in Table 2.

In each subsequent pass, it starts with the set of *classrules* found to be frequent in the previous pass. It uses the previous pass's set to generate new set of possibly frequent *classrules*, called candidate *classrules*. The actual supports and confidences for these candidate *classrules* are calculated during the pass over the dataset. At the end of the pass, it determines which of the candidate *classrules* are truly frequent as shown in Table 3, which shows the outcome of pass 2.

Table 2. The output of the Algorithm's First Pass

Details			Sup	Conf
1st pass	C1	[{(A,e)}, (4)], ((C,n),1)	0.1	25
		[{(A,e)}, (4)], ((C,y),3)	0.3	75
		[{(A,f)}, (1)], ((C,n),1)	0.1	100
		[{(A,g)}, (5)], ((C,n),3)	0.3	60
		[{(A,g)}, (5)], ((C,y),2)	0.2	40

		[{(B,p)}, (3)], ((C,n),1)	0.1	33
		[{(B,q)}, (5)], ((C,n),2)	0.2	40
		[{(B,w)}, (2)], ((C,n),2)	0.2	100
		[{(B,p)}, (3)], ((C,y),2)	0.2	67
		[{(B,q)}, (5)], ((C,y),3)	0.3	60
	F1	[{(A,e)}, (4)], ((C,y),3)	0.3	75
		[{(A,g)}, (5)], ((C,n),3)	0.3	60
		[{(B,w)}, (2)], ((C,n),2)	0.2	67
		[{(B,p)}, (3)], ((C,y),2)	0.3	60
		[{(B,q)}, (5)], ((C,y),3)	0.2	100

Table 3. The output of the Algorithm's Second Pass

2nd pass	C2	[{(A,e),(B,p)}, (3)],((C,n),1)	0.1	33
		[{(A,e),(B,p)}, (3)],((C,y),2)	0.2	67
		[{(A,e),(B,q)}, (1)],((C,y),1)	0.1	100
		[{(A,f),(B,q)}, (1)],((C,n),1)	0.1	100
		[{(A,g),(B,q)}, (3)],((C,n),1)	0.1	33
		[{(A,g),(B,q)}, (3)],((C,y),2)	0.2	67
		[{(A,g),(B,w)}, (2)],((C,n),2)	0.2	100
	L2	[{(A,e),(B,p)}, (3)],((C,y),2)	0.2	67
		[{(A,g),(B,q)}, (3)],((C,y),2)	0.2	67
		[{(A,g),(B,w)}, (2)],((C,n),2)	0.2	100

After those new frequent *classrules* have been identified, the algorithm produces the rules CAR_k using the *GenerateRules* function as shown in Table 4:

Table 4. Result of the GenerateRules function

CAR1	(A, e)→(C,y)
	(A, g)→(C,n)
	(B, p)→(C,y)
	(B, q)→(C,y)
	(B, w)→(C,n)
CAR2	{(A, e), (B, p)} → (C, y),
	{(A, g), (B, q)} → (C, y)
	{(A, g), (B, w)} → (C, n)
CARs	CAR1 ∪ CAR2

Finally, rule pruning is performed on these rules, and after that produces *pCARs* as shown in Table 5:

Table 5. The result after pruning

pCAR1	(A, e)→(C,y)
	(A, g)→(C,n),
	(B, p)→(C,y),
	(B, q)→(C,y),
	(B, w)→(C,n)
pCAR2	{(A, g), (B, q)} → (C, y)
pCARs	pCAR1 ∪ pCAR2

The RG algorithm is similar to Apriori algorithm [13], the difference is that we need to increment the support counts of the *rulecond* and the *classrule* separately whereas in algorithm Apriori only one count is updated. This is performed in order to be able to compute the confidence of the *classrule*.

2.2 The Classifier Builder (CB) Algorithm

The second step to perform associative classification is to use the algorithm called Classifier Builder (CB). All the possible subsets would involve evaluating the training data and selecting the subset with the right rule sequence that gives the least number of errors. The CB algorithm is a heuristic one. However, the classifier it builds performs very well as compared to that built by C4.5 [14]. In CB algorithm [13], the generated CARs are ordered based on the following definition:

Given two rules, *rule_i* and *rule_j*, *rule_i* > *rule_j* (*rule_i* is said to have higher precedence than *rule_j*) if:

1. the confidence of *rule_i* is higher than the confidence of *rule_j*, or
2. they both have the same confidence, but the support of *rule_i* is higher than that of *rule_j*, or
3. both rules have the same confidence value as well as the same supports value, but *rule_i* is generated earlier than *rule_j*.

Let *carSet* be the set of *CARs* and *D* be the training dataset. The basic idea of the model construction algorithm is to choose a subset of rules in *carSet* that have high precedence to cover the training data *D*. The built classifier has the following format:

$< r_1, r_2, \ldots, r_n,$ *defaultClass* $>$, where $r_i \in$ *carSet*, $r_x > r_y$ if $x < y$. if none of the rules can classify a tuple, then by default it is assumed to have *DefaultClass*. The CB algorithm is given in Figure 2.

In CB algorithm, the rules are sorted based to the ordering mentioned in the algorithm shown in Figure 2 (line 1). The algorithm then selects rules (line 2-3) for the classifier from *carSet*. The algorithm examines each of the selected rules and the rule under consideration is marked if it can correctly classify at least one tuple in the training set (line 5,6). If the rule is marked, all the tuples that the rule covers are removed from the training set and the majority class of the remaining tuples in the training set becomes the default class label (line 11,12). A rule that gets marked is appended to the end of the classifier.

Let C_r refer the list of rules ending in rule *r* that have been selected for inclusion in the classifier so far. The classifier C_r is used to classify the tuples of the training set, and assess the performance of the classifier. Since the classification values of the tuples in the training set are known, each classification attempt can be recorded as a correct classification or incorrect classification. When all the instances are classified, the classifier will be assigned an error rate which is the total number of incorrect classifications over the total number of classifications. The rule that has the least number of errors is identified and all rules added after this rule are removed. The default class label that is attached to that rule is designated as the default class label of the classifier (as in line 17).

```
1   carSet = sort (carSet);
2   for each rule r ∈ carSet in sequence do
3       temp = ∅;
4       for each tuple t ∈ D do
5           if t satisfies the conditions of r then
6               store t.id in temp and mark r if it correctly classifies t;
7       if r is marked then
8           insert r at the end of C;
9           delete all the tuples with the ids in temp from D;
10          selecting a default class for the current C;
11          compute the total number of errors of C;
12      end
13  end
14  Find the first rule p in C with the lowest total number
        of errors and drop all the rules after p in C;
15  Add the default class associated with p to end of C,
        and return C (our classifier).
```

Figure 2: The CB algorithm [13]

The CB algorithm adheres to the following two criteria:
Criteria 1. Every tuple in the training dataset is covered by the rule whose priority is the highest among the rules that can cover the tuple. This is due to the sorting performed in line 1.

Criteria 2. Every rule in *C* performs correct classification on at least one remaining training tuple when it is selected. This is because of the effect of lines 5-7 [13].

The problem of CB algorithm is its simplicity as it fits only for small datasets that can be accommodated in main memory and processed there. It is unsuitable for huge datasets that don't fit in main memory because then the CB needs to make may passes over data residing in hard disk.

The following example demonstrates how the model will be used to predict an unknown subject or tuple. Let us assume our model consist of four rules in the following order:

(1) *age = young* → *buys-computer = no* [Sup:0.05, Conf: 0.9]
(2) *age = young* ^ *income = high* → *buys-computer = no* [Sup:0.03, Conf: 0.8]
(3) *age = young* ^ *student = no* → *buys-computer = no* [Sup:0.02, Conf:0.78]
(4) *age = young* ^ *student = yes* → *buys-computer = yes* [Sup:0.015, Conf: 0.75]

Let us consider the data instance for which we want to predict the class:
(*age = young*) ^ (*income = high*) ^ (*student = yes*) → ??,

Assume we want to predict the class label for this instance, if the user will buy computer or not. All rules that cover the new instance are selected and confidence (or support) is used to weight the predictions made by each of these rules. In this case, we will use confidence to select the appropriate class label. Rules 1,2 and 4 are selected and choose the highest confidence for the new prediction. So, the first rule that has the highest confidence is the prediction of the new instance, *(buy_computer = yes)* for the test instance.

3. Prediction of Heart Disease

Heart disease is a common term that means that the heart is not functioning normally. Babies can be born with heart disease which is referred to as *congenital heart disease*. If people get heart disease later in life this is called *acquired heart disease* which is the most common. [16]. One of the most wide-spread types of acquired heart disease *is Coronary Artery Disease* (CAC). It is caused by a problem with the blood receptacles that transfers blood to the heart muscle. If these blood receptacles become very small, or if they get blocked, blood cannot flow through them in a normal way. Since less blood is provided to the heart muscle, the muscle cannot work at its regular capacity. The heart muscle becomes ill and weak and can even completely stop functioning if blood flow stops. Blocked blood receptacles in the heart are usually caused by rise cholesterol, smoking, diabetes, rise blood pressure, and inherited traits from ancestors. All these causes can damage the lining of the heart's blood receptacles and cause them to become narrowed or even blocked entirely [16, 17].

Table 6, which is adapted from [18] shows the factors that we've used along with a brief description of each factor. In our sample dataset only the first four factors are used: age (A), gender (G), diabetes (D) and obesity (O), in order improve heart disease (HD) prediction using associative classification.

Table 6. Factors contributing to Heart Disease

No.	Name	Description
1	Age (A)	is the most important risk factor that causes developing heart disease.
2	Gender (G)	male or female
3	Diabetes (D)	Indicates whether the person has diabetes or no
4	Obesity (O)	Captures the state of being overweight.
5	Heart disease (HD)	this is the class label. It can be YES nor NO.
6	No. of Identical Rows (NIR)	number of rows in the full table that have same values.

Table 7. Sample Health Dataset

A	G	D	O	HD	NIR
senior	m	yes	yes	yes	110
senior	f	yes	no	no	70
young	m	no	yes	yes	70
young	f	yes	no	no	60
senior	f	no	yes	yes	100
young	m	yes	yes	yes	130
young	f	yes	yes	yes	110
senior	f	yes	yes	yes	90
senior	m	no	no	no	140
young	f	no	no	no	120

The sample dataset that we've used consist of ten identical tuples (rows). Considering the duplicates, it is exactly one thousand tuples in the original database. The class label factor, heart disease (HD), has two distinct values, namely {yea, no}. The other factors contain various values depending on patients' data. Table 7 shows the dataset.

Let us consider a new row that is to be inserted with the following values: <***senior, m, no, yes***> for the attribute *A*,*G*,*D*, and *O*, respectively. We will use RG for generating association rules and CB for building a classifier to produce the best classifier out of the whole set of rules.

Phase 1, the RG algorithm is used to generate all class association rules by making multiple passes over the dataset. All rules that cover at least one tuple of the new instance are enclosed in a rectangle as shown in Table 8, to be used later in the second transaction of associative classification as a possible rule. In the first pass of RG, the *classrules* are generated by joining every attribute with the class label (HD), and calculate support and confidence for each *classrule* as shown in Table 8.

In the second pass, the class rules are generated by joining every two attributes with the HD, and calculate support and confidence for each *classrule* as shown in Table 9. In the third pass, the *classrules* are generated by joining every three attributes with the HD, and calculate support and confidence for each *classrule* as shown in Table 10. In the fourth pass which is the last round in this transaction, because we have only four factors in our sample dataset, the *classrules* are generated by joining the four attributes with the HD, and calculate support and confidence for each *classrule* as shown in Table 11.

Phase 2, the CB algorithm is used to produce the best classifier rule that gives the highest confidence and support with the least number of errors of the generated rules. All rules that are marked with the rectangles, which satisfy at least one rule of the new instance, are selected and ordered based on the order mentioned in the CB algorithm. In this work, only twenty rules have been selected, which got the highest priority among the rules that can cover the tuple as shown in Table 12. The column "No. of Errors" in Table 12 refers to the number of errors that are made by the current rule. By passing on the rules, we found that the first rule has the highest priority with less number of errors, so it has been selected as the prediction of the new instance. Accordingly, HD predication of the new instance is <*senior*, *m*, *no*, *yes* ⇒ *HD = ye*>.

Table 8. Result After 1st pass of the Algorithm

	Attribute → Label	Details	Sup	Conf
1st pass	A → HD	[{(A,senior)}, (510)], ((HD,no),210)	21	41.2
		[{(A,senior)}, (510)], ((HD,yes),300)	30	58.8
		[{(A,young)}, (490)], ((HD,no),180)	18	36.7
		[{(A,young)}, (490)], ((HD,yes),310)	31	63.3
	G → HD	[{(G,f)}, (550)], ((D,no),220)	22	40.0
		[{(G,f)}, (550)], ((D,yes),330)	33	60.0
		[{(G,m)}, (450)], ((D,no),210)	21	46.7
		[{(G,m)}, (450)], ((D,yes),240)	24	53.3
	D → HD	[{(D,no)}, (430)], ((HD,no),260)	26	60.5
		[{(D,no)}, (430)], ((HD,yes),170)	17	39.5
		[{(D,yes)}, (570)], ((HD,no),130)	13	22.8
		[{(D,yes)}, (570)], ((HD,yes),440)	44	77.2
	O → HD	[{(O,no)}, (390)], ((HD,no),390)	39	100.0
		[{(O,yes)}, (610)], ((HD,yes),610)	61	100.0

Table 9. Result After 2nd pass of the Algorithm

	Attribute → Label	Details	Sup	Conf
2nd pass	A ^ G → HD	[{(A,senior),(G,f)}, (260)],((HD,no),70)	7	26.9
		[{(A,senior),(G,f)}, (260)],((HD,yes),190)	19	73.1
		[{(A,senior),(G,m)}, (250)],((HD,no),140)	14	56.0
		[{(A,senior),(G,m)}, (250)],((HD,yes),110)	11	44.0
		[{(A,young),(G,f)}, (290)],((HD,no),180)	18	62.1
		[{(A,young),(G,f)}, (290)],((HD,yes),110)	11	37.9
		[{(A,young),(G,m)}, (200)],((HD,yes),200)	20	100.0
	A ^ D → HD	[{(A,senior),(D,no)}, (240)],((HD,no),140)	14	58.3
		[{(A,senior),(D,no)}, (240)],((HD,yes),100)	10	41.7
		[{(A,senior),(D,yes)}, (270)],((HD,no),70)	7	25.9
		[{(A,senior),(D,yes)}, (270)],((HD,yes),200)	20	74.1
		[{(A,young),(D,no)}, (190)],((HD,no),120)	12	63.2
		[{(A,young),(D,no)}, (190)],((HD,yes),70)	7	36.8
		[{(A,young),(D,yes)}, (300)],((HD,no),60)	6	20.0
		[{(A,young),(D,yes)}, (300)],((HD,yes),240)	24	80.0
	A ^ O → HD	[{(A,senior),(O,no)}, (210)],((HD,no),210)	21	100.0
		[{(A,senior),(O,yes)}, (300)],((HD,yes),300)	30	100.0
		[{(A,young),(O,no)}, (180)],((HD,no),180)	18	100.0
		[{(A,young),(O,yes)}, (310)],((HD,yes),310)	31	100.0
	G ^ D → HD	[{(G,f),(D,no)}, (220)],((HD,no),120)	12	54.5
		[{(G,f),(D,no)}, (220)],((HD,yes),100)	10	45.5
		[{(G,f),(D,yes)}, (330)],((HD,no),130)	13	39.4
		[{(G,f),(D,yes)}, (330)],((HD,yes),200)	20	60.6
		[{(G,m),(D,no)}, (210)],((HD,no),140)	14	66.7
		[{(G,m),(D,no)}, (210)],((HD,yes),70)	7	33.3
		[{(G,m),(D,yes)}, (240)],((HD,yes),240)	24	100.0
	G ^ O → HD	[{(G,f),(O,no)}, (250)],((HD,no),250)	25	100.0
		[{(G,f),(O,yes)}, (300)],((HD,yes),300)	30	100.0
		[{(G,m),(O,no)}, (140)],((HD,no),140)	14	100.0
		[{(G,m),(O,yes)}, (310)],((HD,yes),310)	31	100.0
	D ^ O → HD	[{(D,no),(O,no)}, (260)],((HD,no),260)	26	100.0
		[{(D,no),(O,yes)}, (170)],((HD,yes),170)	17	100.0
		[{(D,yes),(O,no)}, (130)],((HD,no),130)	13	100.0
		[{(D,yes),(O,yes)}, (440)],((HD,yes),440)	44	100.0

Table 10. Result After 3rd pass of the Algorithm

	Attribute → Label	Details	Sup	Conf
3rd pass	A ^ G ^ D → HD	[{(A,senior),(G,f),(D,no)}, (100)],((HD,yes),100)	10	100.0
		[{(A,senior),(G,f),(D,yes)}, (160)],((HD,no),70)	7	43.8
		[{(A,senior),(G,f),(D,yes)}, (160)],((HD,yes),90)	9	56.3
		[{(A,senior),(G,m),(D,no)}, (140)],((HD,no),140)	14	100.0
		[{(A,senior),(G,m),(D,yes)}, (110)],((HD,yes),110)	11	100.0
		[{(A,young),(G,f),(D,no)}, (120)],((HD,no),120)	12	100.0
		[{(A,young),(G,f),(D,yes)}, (170)],((HD,no),60)	6	35.3
		[{(A,young),(G,f),(D,yes)}, (170)],((HD,yes),110)	11	64.7
		[{(A,young),(G,m),(D,no)}, (70)],((HD,yes),70)	7	100.0
		[{(A,young),(G,m),(D,yes)}, (130)],((HD,yes),130)	13	100.0
	A ^ G ^ O → HD	[{(A,senior),(G,f),(O,no)}, (70)],((HD,no),70)	7	100.0
		[{(A,senior),(G,f),(O,yes)}, (190)],((HD,yes),190)	19	100.0
		[{(A,senior),(G,m),(O,no)}, (140)],((HD,no),140)	14	100.0

		[{(A,senior),(G,m),(O,yes)}, (110)],((HD,yes),110)	11	100.0
		[{(A,young),(G,f),(O,no)}, (180)],((HD,no),180)	18	100.0
		[{(A,young),(G,f),(O,yes)}, (110)],((HD,yes),110)	11	100.0
		[{(A,young),(G,m),(O,yes)}, (200)],((HD,yes),200)	20	100.0
	A ^ D ^ O → HD	[{(A,senior),(D,no),(O,no)}, (140)],((HD,no),140)	14	100.0
		[{(A,senior),(D,no),(O,yes)}, (100)],((HD,yes),100)	10	100.0
		[{(A,senior),(D,yes),(O,no)}, (70)],((HD,no),70)	7	100.0
		[{(A,senior),(D,yes),(O,yes)}, (200)],((HD,yes),200)	20	100.0
		[{(A,young),(D,no),(O,no)}, (120)],((HD,no),120)	12	100.0
		[{(A,young),(D,no),(O,yes)}, (70)],((HD,yes),70)	7	100.0
		[{(A,young),(D,yes),(O,no)}, (60)],((HD,no),60)	6	100.0
		[{(A,young),(D,yes),(O,yes)}, (240)],((HD,yes),240)	24	100.0
	G ^ D ^ O → HD	[{(G,f),(D,no),(O,no)}, (120)],((HD,no),120)	12	100.0
		[{(G,f),(D,no),(O,yes)}, (100)],((HD,yes),100)	10	100.0
		[{(G,f),(D,yes),(O,no)}, (130)],((HD,no),130)	13	100.0
		[{(G,f),(D,yes),(O,yes)}, (200)],((HD,yes),200)	20	100.0
		[{(G,m),(D,no),(O,no)}, (140)],((HD,no),140)	14	100.0
		[{(G,m),(D,no),(O,yes)}, (70)],((HD,yes),70)	7	100.0
		[{(G,m),(D,yes),(O,yes)}, (240)],((HD,yes),240)	24	100.0

Table 11. Result After 4th pass of the Algorithm

	Attribute → Label	Details	Sup	Conf
4th pass	A ^ G ^ D ^ O → HD	[{(A,senior),(G,f),(D,no),(O,yes)}, (100)],((HD,yes),100)	10	100.0
		[{(A,senior),(G,f),(D,yes),(O,no)}, (70)],((HD,no),70)	7	100.0
		[{(A,senior),(G,f),(D,yes),(O,yes)}, (90)],((HD,yes),90)	9	100.0
		[{(A,senior),(G,m),(D,no),(O,no)}, (140)],((HD,no),140)	14	100.0
		[{(A,senior),(G,m),(D,yes),(O,yes)}, (110)],((HD,yes),110)	11	100.0
		[{(A,young),(G,f),(D,no),(O,no)}, (120)],((HD,no),120)	12	100.0
		[{(A,young),(G,f),(D,yes),(O,no)}, (60)],((HD,no),60)	6	100.0
		[{(A,young),(G,f),(D,yes),(O,yes)}, (110)],((HD,yes),110)	11	100.0
		[{(A,young),(G,m),(D,no),(O,yes)}, (70)],((HD,yes),70)	7	100.0
		[{(A,young),(G,m),(D,yes),(O,yes)}, (130)],((HD,yes),130)	13	100.0

Table 12. Rules with the Highest Priority

ID	Details	Sup	Conf	No of Errors
1	[{(O,yes)}, (610)], ((HD,yes),610)	61	100.0	0
2	[{(D,yes),(O,yes)}, (440)],((HD,yes),440)	44	100.0	1
3	[{(G,m),(O,yes)}, (310)],((HD,yes),310)	31	100.0	0
4	[{(A,young),(O,yes)}, (310)],((HD,yes),310)	31	100.0	1
5	[{(A,senior),(O,yes)}, (300)],((HD,yes),300)	30	100.0	0
6	[{(G,f),(O,yes)}, (300)],((HD,yes),300)	30	100.0	1
7	[{(D,no),(O,no)}, (260)],((HD,no),260)	26	100.0	1
8	[{(G,m),(D,yes)}, (240)],((HD,yes),240)	24	100.0	1
9	[{(G,m),(D,yes),(O,yes)}, (240)],((HD,yes),240)	24	100.0	1
10	[{(A,young),(D,yes),(O,yes)}, (240)],((HD,yes),240)	24	100.0	2
11	[{(A,senior),(O,no)}, (210)],((HD,no),210)	21	100.0	1
12	[{(A,young),(G,m)}, (200)],((HD,yes),200)	20	100.0	1
13	[{(A,young),(G,m),(O,yes)}, (200)],((HD,yes),200)	20	100.0	1
14	[{(A,senior),(D,yes),(O,yes)}, (200)],((HD,yes),200)	20	100.0	1
15	[{(G,f),(D,yes),(O,yes)}, (200)],((HD,yes),200)	20	100.0	2
16	[{(A,senior),(G,f),(O,yes)}, (190)],((HD,yes),190)	19	100.0	1
17	[{(D,no),(O,yes)}, (170)],((HD,yes),170)	17	100.0	0
18	[{(A,senior),(G,m),(D,no)}, (140)],((HD,no),140)	14	100.0	0
19	[{(G,m),(O,no)}, (140)],((HD,no),140)	14	100.0	1
20	[{(A,senior),(G,m),(O,no)}, (140)],((HD,no),140)	14	100.0	1

4. Conclusion

In this paper, we have briefly summarized how Associative Classification works. Heart disease is a very common disease and is receives big attention from the medical community. The main purpose of this paper is to demonstrate how a classifier based on the associative classification technique can be built and used to classify health data pertaining to heart disease. The two phases of classification by association, namely the rule generation phase and the classifier building phase have been applied to the medical data in order to build the classification rules. Given the data of a new patient, the classifier can then be applied to predict whether the new patient may have or may not have a heart disease.

References

[1] Z. Fu , B. L. Golden , S. Lele , S. Raghavan and E. Wasil "Diversification for better classification trees", Comput. Oper. Res., vol. 33, no. 11, pp.3185 -3202 2006.

[2] A. Alashqur, "Representation Schemes Used by Various Classification Techniques – A Comparative Assessment" International Journal of Computer Science Issues (IJCSI), Volume 12, Issue 6, pages 55-63, November 2015. ISSN: 1694-0814.

[3] J. Gray and G. Fan "Classification tree analysis using TARGET", Comput. Statist. Data Anal., vol. 52, pp.1362 -1372, 2008.

[4] A. Alashqur, "A Novel Methodology for Constructing Rule-Based Naïve Bayesian Classifiers" International Journal of Computer Science & Information Technology (IJCSIT), Vol 7, No 1, Pages 139-151, February 2015. ISSN: 0975 – 4660.

[5] A. Freitas , D. C. Wieser and R. Apweiler "On the importance of comprehensible classification models for protein function prediction", IEEE/ACM Trans. Comput. Biol. Bioinformat, vol. 7, no. 1, pp.172-182, 2010.

[6] A. Alashqur, "Using a Lattice Intension Structure to Facilitate User-Guided Association Rule Mining." Journal of Computerand Information Science, Published by Canadian Center of Science and Education. Vol. 5, No.2, pages 11-21, March 2012. ISSN: 1913-8989.

[7] Han, J., M. Kamber and J. Pei Data Mining: Concepts and Techniques, 3rd Ed. 2011. Morgan Kaufmann.

[8] N. Hussein, A. Alashqur, and B. Sawan, "Using the Interestingness Measure Lift to Generate Association Rules" Journal of Advanced Computer Science & Technology, Vol 4, No 1, pages 156-162, June 2015. ISSN: 2227-4332.

[9] A. Alashqur, "RDB-MINER: A SQL-Based Algorithm for Mining True Relational Databases" Journal of Software, Vol. 5, No. 9, Pages: 998-1005, Sept. 2010.

[10] Tan, P-N, M. SteinBach, and V. Kumar, 2005. Introduction to Data Mining, Addison Wesley. ISBN-10: 0321321367.

[11] A. Alashqur, "Mining Association Rules: A Database Perspective" International Journal of Computer Science and Network Security (IJCSNS), pages 69-74, VOL.8 No.12, December 2008.

[12] S. Palanisamy, 2006, "Association Rule Based Classification" pp. 15–30.

[13] B. Liu, W . Hsu and Y. Ma: "Integrating Classification and Association Rule Mining" , KDD-98 Proceedings. pp.1-4.

[14] J.R Quinlan. 1993. C4.5: programs for machine learning. Morgan Kaufmann.

[15] R.Agrawal and R.Srikant, 1994, "Fast algorithms for mining association rule", Proceedings of the International Conference on Very Large Databases, pp.480—498.

[16] "https://simple.wikipedia.org/wiki/Heart_disease" as of August 12, 2016.

[17] M.Akhil jabbar, Priti Chandra, and B.L Deekshatulu, "Heart Disease Prediction System using Associative Classification and Genetic Algorithm," International Conference on Emerging Trends in Electrical, Electronics and Communication Technologies-ICECIT, pp. 3, 2012.

[18] D. AL-Dlaeen, A. Alashqur, "Using Decision Tree Classification to Assist in the Prediction of Alzheimer's Disease" Sixth International Conference on Computer Science and Information Technology (CSIT), Pages 122-126, March 26-27, 2014. Amman, Jordan. IEEE Xplore Digital Library, ISBN: 978-1-4799-3998-5.

Hisham Ogbah is currently working towards the MSc degree in Computer Science at the Applied Science University in Amman, Jordan. He received his B.Sc. degree in Computer Science from Sikkim Manipal University, India, in 2008. Between 2009 and 2013 he worked as a software developer in Yemen. His research interests include classification techniques and concept drift.

Abdallah Alashqur is an associate professor in the Faculty of IT at the Applied Science University (ASU) in Amman, Jordan. Dr. Alashqur holds a Master's and a Ph.D. degrees from the University of Florida, Gainesville. After obtaining his Ph.D. degree in 1989, he worked for around seventeen years in industry (in the USA). He joined ASU in 2006. His research interests include data mining and database systems.

Hazem Qattous is currently an assistant professor in the Faculty of Information Technology at the Applied Science University in Amman, Jordan. Dr. Qattous holds a Ph.D. degree in Computing from Glasgow University, UK. His research interests are in Human-Computer Interaction.