Data Mining
Assignment 1

Ali Gholami

Department of Computer Engineering & Information Technology
Amirkabir University of Technology

http://ceit.aut.ac.ir/~aligholamee
aligholamee@aut.ac.ir

Abstract

In this assignment, several paramount concepts of *Data Analysis* will be explained. we'll discuss the importance of metrics in the first theoretical problem. A quick review on the *Apriori* algorithm for the *Association Rule Mining* will be explained also. We'll also show how *Weka* can be used for *Association Rule Mining.* Furthermore, The effectiveness of *Normalization* concept is proposed. Finally, an *Statistical* point of view will help us to demonstrate and rationalize the relationship between the *Performance* of the *Learning Algorithm* and the amount of *Data* available. A chief section of this assignment is dedicated to solve the *Titanic* problem, which is a great practice of data mining concepts in production. We'll use *Python* programming language and three main libraries; *Scikit-Learn*, *Pandas* and *Numpy* to tackle this problem. The Python implementation of the Titanic problem is provided on a *Jupyter Notebook* attached with this report.

Keywords. *Apriori, Association Rule Mining, Normalization, Generalization, Preprocessing, Feature Engineering, Scikit-Learn, Pandas, Numpy, Python 3.5.*

1 Performance Metrics Analysis

Given the following *Confusion Matrix* for a prediction about cancer.

		Predicted Class		
		Cancer = Yes	Cancer = No	Total
Actual Class	Cancer = Yes	60	290	350
	Cancer = No	150	9500	9650
	Total	210	9790	10000

Table 1.1: Confusion matrix of cancer prediction.

Compute each of these performance metrics.

(a) Accuracy

(b) Sensitivity

(c) Precision

(d) Specificity

(e) F-measure

Solution

Before getting into the computations, we'll review the *nicknames* and *formulas* to calculate each of these metrics. We have computed each of these metrics in front of them.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} = \frac{60 + 9500}{60 + 9500 + 290 + 150} = 0.956 \tag{1.1}$$

$$TPR = Recall = Sensitivity = \frac{TP}{P} = \frac{TP}{TP + FN} = \frac{60}{60 + 290} = 0.171 \tag{1.2}$$

$$PPV = Precision = \frac{TP}{TP + FP} = \frac{60}{60 + 150} = 0.285 \tag{1.3}$$

$$TNR = Specificity = \frac{TN}{N} = \frac{TN}{TN + FP} = \frac{9500}{9500 + 150} = 0.984 \tag{1.4}$$

$$F - measure = \frac{2 * TP}{2 * TP + FP + FN} = \frac{2 * 60}{2 * 60 + 150 + 290} = 0.214 \tag{1.5}$$

Our mission is done! Nevertheless, we continue the explanation for almost each of these metrics. We'll discuss why *Accuracy* itself would be a bad metric in most of the challenging cases.

Why Performance Metrics Are Important

Metrics are important because they allow us to judge about models ability in prediction task. Without metrics we won't be able to compare models; Thus we won't be able to improve each.

What Kind of Performance Metrics Are Useful

Not all of metrics describe this ability correctly in different conditions. Generally speaking, we need an *Objective* metric. A metric could exhibit a great number for a classifier that classifies data as *True* always. This can happen is *Imbalanced Datasets*. The important thing is that, we need to establish a *Baseline* before getting into these numbers. We need to measure the performance for a simple system before start tuning these numbers up. The absolute maximum performance that a machine learning system can achieve, is called *Ceiling*. The performance we get with respect to the numbers(*like numbers calculated above*), is bound between *Baseline* and *Ceiling* values.

$$Baseline < Performance < Ceiling \tag{1.6}$$

Possibility of Getting Complete Performance

No, its not possible! Even using 2 humans to classify some data, *they might not agree 100% of the times.*

Accuracy Paradox

Accuracy is the proportion of the correct results that a classifier achieved. Assume a classifier who classifies its inputs as *true* always. The denominator for the *Accuracy* formula is the size of the dataset, which is a constant. The numerator while, contains $TP + TN$. This classifier predicts a great number of *TP* and a small number of *TN*. If the assumption changes to be that classification always turns out to be *false*, we'll get a huge value for *TN* and a small value for *TP*. The addition is the same by the way. Thus the accuracy of a *dummy* model can be amazing in both criteria. Thus, *Accuracy* is not a reliable metric in machine learning problems. We call this *Accuracy Paradox*.

Recall

This metric describes that, out of all the positive examples there were, what fraction did the classifier pick up?

Precision

This metric states that, out of all the examples the classifier labeled as positive, what fraction were correct?

Combination of Recall & Precision

Combination of these metrics, causes the results to be *balanced*.

F_β Score

F_β combines *Precision* and *Recall*. We'll talk about its advantages later in this assignment.

2 The Association Rule Mining

Following are four random transactions. What association rules can be found using Apriori algoirhtm?

- Minimum Support: 60%
- Minimum Confidence: 80%

Trans_ID	*Item_List*
T1	K, A, D, B
T2	D, A, C, B, E
T3	C, A, B, E
T4	B, A, D

Table 2.1: Demonstration of Database Transactions to Mine Rules From.

Solution

Before getting into the details of the *Apriori* algorithm, it worths mentioning that the *Minimum Support* is used to find the appropriate *itemset* and the *Minimum Confidence* will be used to find the final *rules*. Firstly, we need to find all items with size 1 and their frequencies. Note that in each of these sections, the value of *Support* for an itemset X can

Itemset	*Frequency*	*Support*
{A}	4	100%
{B}	4	100%
{C}	2	50%
{D}	3	75%
{E}	2	50%
{K}	1	25%

Table 2.2: Unity sized itemsets and their corresponding frequencies and support.

be computed using the following formula:

$$Support(X) = \frac{|\{t \epsilon T;\ X \subseteq t\}|}{|T|} \tag{2.1}$$

Itemset	*Frequency*	*Support*
{AB}	4	100%
{AD}	3	75%
{BD}	3	75%

Table 2.3: Itemsets of size 2 and their support.

We'll continue computing the itemsets with size 2. Note that the unity sized itemsets with support less than 60% won't be considered in this algorithm. Thus we get: Finally, we reach the moment that there is only 1 terminal left. The following itemset with size 3 will be the stopping point of this algorithm. We can start deriving rules from the itemset in table 2.4.

Itemset	*Frequency*	*Support*
{ABD}	3	75%

Table 2.4: Itemset of size 3 and its support.

To obtain the association rules, we should derive all combinations of $X \Rightarrow Y$ in this itemset. Note that in this part, we use *Confidence*, which tells how often the rule found to be true. *Confidence* is defined as:

$$Confidence(X \Rightarrow Y) = P(Y|X) = \frac{support\_count(A \cup B)}{support\_count(A)} \tag{2.2}$$

Thus we'll have:

- $\{A,\ B\} \rightarrow \{D\} = P(D \mid \{A,\ B\}) = \frac{3}{3} = 1$
- $\{A,\ D\} \rightarrow \{B\} = P(B \mid \{A,\ D\}) = \frac{3}{3} = 1$
- $\{B,\ D\} \rightarrow \{A\} = P(A \mid \{B,\ D\}) = \frac{3}{4} = 0.75$
- $\{D\} \rightarrow \{A,\ B\} = P(\{A,\ B \mid \{D\}) = \frac{3}{3} = 1$
- $\{B\} \rightarrow \{A,\ D\} = P(\{A, D\}| \ \{B\}) = \frac{3}{3} = 1$
- $\{A,\ B\} \rightarrow \{D\} = P(D \mid \{A,\ B\}) = \frac{3}{3} = 1$

The rules with confidence above 0.8 are acceptable.

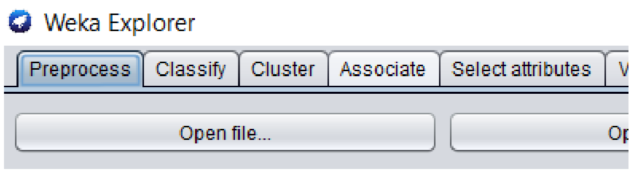


Figure 2.1: Open a random dataset.

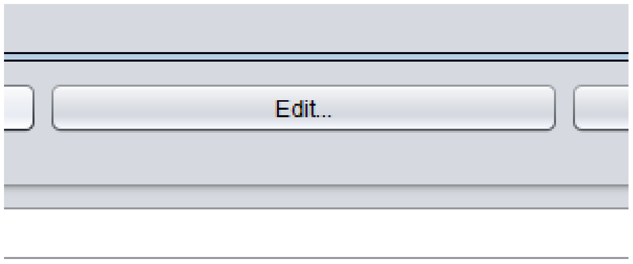
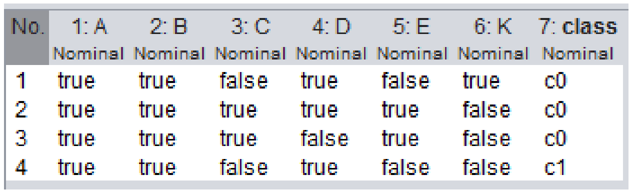


Figure 2.2: Customize the opened dataset using edit file.

Weka Tutorial on Association Rule Mining

In this section, we'll walk step by step to mine an association rule on an customized dataset.

1. Go to the *preprocess* tab. Hit the *Open File* button.
2. Edit the file by pressing the *Edit* button.
3. Add your dataset with respect to its attributes and class values.



No.	1: A Nominal	2: B Nominal	3: C Nominal	4: D Nominal	5: E Nominal	6: K Nominal	7: **class** Nominal
1	true	true	false	true	false	true	c0
2	true	true	true	true	true	false	c0
3	true	true	true	false	true	false	c0
4	true	true	false	true	false	false	c1

Figure 2.3: Add samples in the database for association.

4. Go to the associate tab by pressing the *associate* button.

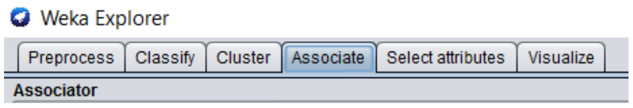
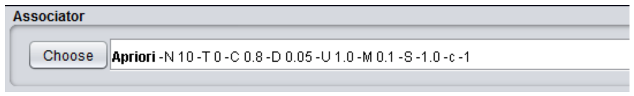


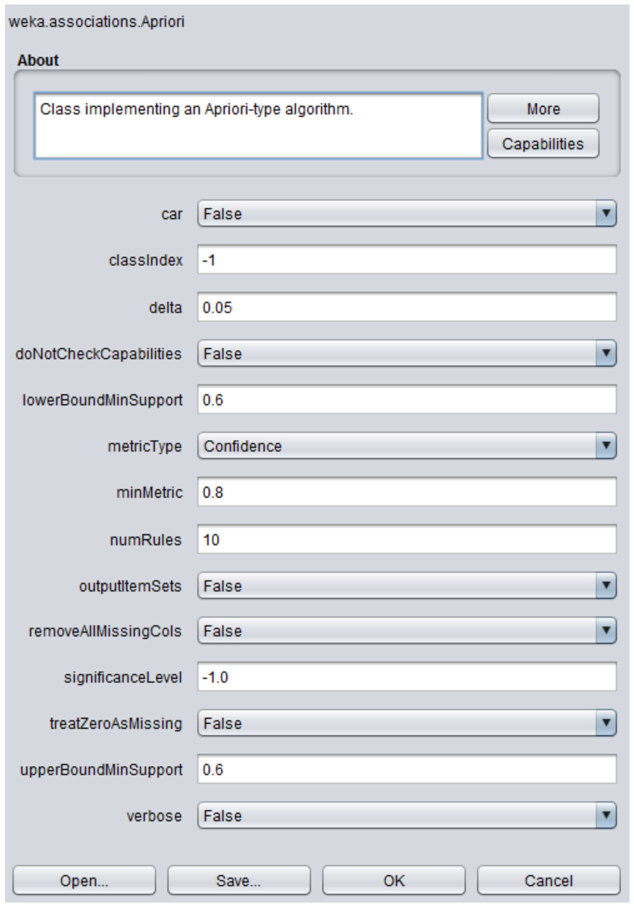
Figure 2.4: Open the association rules section.

5. Choose the algorithm and its parameters.


Associator

Choose **Apriori** -N 10 -T 0 -C 0.8 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

Figure 2.5: Tune the parameters of the algorithm.


weka.associations.Apriori

About

Class implementing an Apriori-type algorithm.

car	False
classIndex	-1
delta	0.05
doNotCheckCapabilities	False
lowerBoundMinSupport	0.6
metricType	Confidence
minMetric	0.8
numRules	10
outputItemSets	False
removeAllMissingCols	False
significanceLevel	-1.0
treatZeroAsMissing	False
upperBoundMinSupport	0.6
verbose	False

Figure 2.6: Set the support range.

6. Set minimum support by filling the lower bound and upper bound support range.
7. Set the minimum confidence using the confidence form.
8. Run the algorithm. The following is the result after hitting the *start* button.

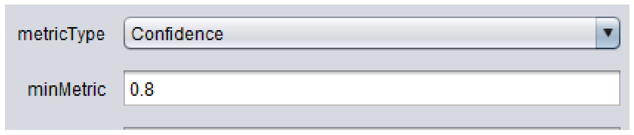
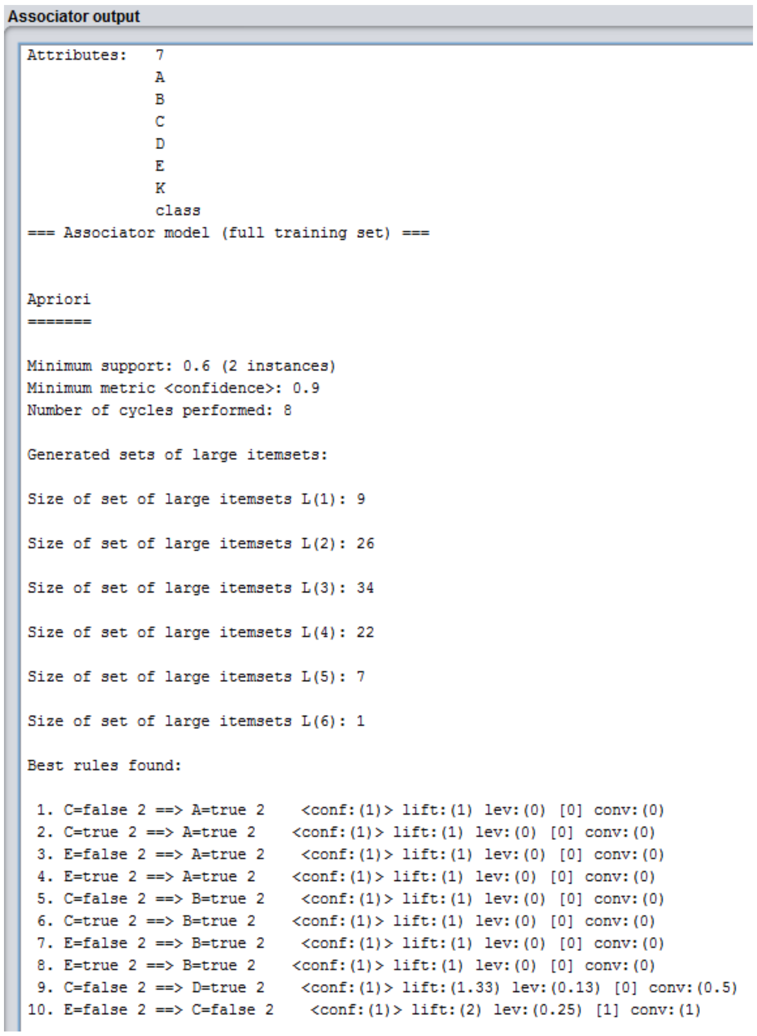

metricType	Confidence
minMetric	0.8

Figure 2.7: Set the minimum confidence.



```
Attributes:   7
              A
              B
              C
              D
              E
              K
              class
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.6 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 8

Generated sets of large itemsets:

Size of set of large itemsets L(1): 9

Size of set of large itemsets L(2): 26

Size of set of large itemsets L(3): 34

Size of set of large itemsets L(4): 22

Size of set of large itemsets L(5): 7

Size of set of large itemsets L(6): 1

Best rules found:

 1. C=false 2 ==> A=true 2    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 2. C=true 2 ==> A=true 2    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 3. E=false 2 ==> A=true 2    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 4. E=true 2 ==> A=true 2    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 5. C=false 2 ==> B=true 2    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 6. C=true 2 ==> B=true 2    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 7. E=false 2 ==> B=true 2    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 8. E=true 2 ==> B=true 2    <conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 9. C=false 2 ==> D=true 2    <conf:(1)> lift:(1.33) lev:(0.13) [0] conv:(0.5)
10. E=false 2 ==> C=false 2    <conf:(1)> lift:(2) lev:(0.25) [1] conv:(1)
```

Figure 2.8: Association results.

3 The Concept of Harmonic Mean and F_β Score

What is F_β score? Describe how the term β affects the final score.

Solution

We saw how the F_1 score is computed. Before getting into the details, let's recall something interesting; **Average**. The simplest form of average is what we describe as *sum of all values divided by their count*. There is another average(*mean*), which is called *Harmonic mean*. Harmonic mean is represented as:

$$\frac{1}{H} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{x_i} \tag{3.1}$$

Lets find the harmonic mean for two scores; Precision and Recall. The equation (2.1) turns out to be:

$$\frac{1}{H} = \frac{1}{2}\sum_{i=1}^{2}\frac{1}{x_i}$$

Replacing the terms, Recall and Precision:

$$H = \frac{1}{\frac{1}{2}(\frac{1}{Recall} + \frac{1}{Precision})} = \frac{2 * Recall * Precision}{Recall + Precision} \tag{3.2}$$

Generalizing this idea, we can multiply H by both sides of (2.1):

$$\frac{1}{n}\sum_{i=1}^{n}\frac{H}{x_i} = 1 \tag{3.3}$$

In other words, the average of ratios between the harmonic mean and the data points is unity. We call H the F_1 score. The important thing is that the effect of each score on the H is considered the same here($\frac{1}{2}$). We call this effectiveness as the *weight* of each score in forming the F_1 score. The summation of weights has to remain 1. Considering the the weight of $\frac{1}{\beta+1}$ as the weight for the *Precision*, we'll have $\frac{\beta}{\beta+1}$ as the weight for the *Recall*. Because we can get

$$\frac{\beta}{\beta+1} + \frac{1}{\beta+1} = \frac{\beta+1}{\beta+1} = 1$$

Rewriting the equation (2.2) by changing the weights of its parameters:

$$F_\beta = \frac{1}{\frac{\beta}{\beta+1}\frac{1}{Recall} + \frac{1}{\beta+1}\frac{1}{Precision}} = \frac{(\beta+1) * Recall * Precision}{Recall + \beta * Precision} \tag{3.4}$$

The Effect of β

In this subsection, we'll focus on the measurements of effectiveness; *Precision* and *Recall*. We'll discover the *effectiveness* of *Precision* and *Recall* on the F_β score. There are following possible conditions for the *Precision* and *Recall* weights.

$$W_{Precision} > W_{Recall}$$

$$W_{Precision} < W_{Recall}$$

$$W_{Precision} = W_{Recall}$$

According to the equation (2.3), the summation of these weights is equal to 1. Thus, in order to get $W_{Precision} = W_{Recall}$, weights are needed to be $\frac{1}{2}$ each. If $\beta > \frac{1}{2}$ the weight of *Precision* is higher than the *Recall* and vice versa. In the measurement process, the users can attach different relative importance to *Precision* and *Recall*. What we want is therefore a parameter(β) to characterize the measurement function in a way that *It measures the effectiveness of prediction with respect to a user who attaches β times as much importance to Recall as Precision.*

4 The Concept of Normalization

Describe the term *Normalization* in data engineering. What do we mean by *Normalizing* the data?

Solution

As the word implies, *Normalization* is the process of adjusting values into alignment. The *transformation* of values in order to represent them in a uniform range, is also called *Normalization*. This topic can have different meaning in different applications.

Normalization of Inputs in Neural Networks

The inputs must have same range of values, otherwise we'll be left with an *ill-conditioned* model after the training.

Convergence of Weights & Biases in Distance Based Classifiers

While using *Distance Based* classifiers, it is important not to get conditioned by features with wider range of possible values. *Normalization* is used to guarantee the *Convergence* of *Weights and Biases* in such conditions. We often call this, the *Convergence of Gradient Descent*, since the gradient descent is used most of the times to optimize the loss.

5 The Effect of Data

"More better data lets the model grasp a better intuition of an specific random event." This is simple expression of the *Law of Large Numbers* in probability theory. The parameters of our model is tuned to predict the real-world data. It could be the data that the model has never seen while being trained. Thus, we can infer that the input data distribution will comply with the *LLN* most of the times. We'll be reviewing two other effects of data in the upcoming subsections; *High Variance* and *High Bias* conditions.

High Variance Condition

Occurs when a model is too complicated and relatively, the amount of data is not enough. We often call this as *Overfitting* in machine learning. The result of the criterion will be

$$Error_{Training} << Error_{Testing}$$

This problem can be solved by reducing the number of features. As an example, assume a language model in which roughly every word in the vocabulary can be considered as a feature. *In this case, adding more data would help.*

High Bias Condition

Occurs when a model is too simple to explain our data. *Adding more data won't help in this case.*