Interactive Rule Interestingness Learning*

Tolga Aydın and H.Altay Güvenir

Bilkent University
Department of Computer Engineering

Bilkent, 06800 Ankara, TURKEY
{atolga, guvenir}@cs.bilkent.edu.tr

Abstract. *Data mining is the efficient discovery of patterns in large databases, and classification rules are perhaps the most important type of patterns in data mining applications. However, the number of such classification rules is generally too huge that selection of interesting ones among all discovered rules becomes an important task. In this paper, factors related to the interestingness of a rule are investigated and some new factors are proposed. Following this, an interactive rule interestingness-learning algorithm (IRIL) is developed to automatically label the classification rules either as "interesting" or "uninteresting" with limited user participation. In our study, VFPI (Voting Feature Point Intervals), a feature projection based concept description learning and classification algorithm, is also developed in the framework of IRIL. Being specific to our concerns, VFPI takes the rule interestingness factors as features and is used to learn the rule interestingness concept and to classify the unlabeled classification rules. The success of the previously developed feature projection based learning and classification techniques encouraged us to develop VFPI. The empirical results based on TCMB (Central Bank of Turkish Republic) data set give promising results.*
Keywords: *rule interestingness, voting, feature projection based classification*

1. Introduction

Data mining is the efficient discovery of patterns, as opposed to data itself, in large databases [5]. Patterns in the data can be represented in many different forms, including classification rules, association rules, clusters, sequential patterns, time series, contingency tables, and others [6]. However, the number of discovered patterns is usually too big such that the user analyzing the patterns is generally interested in a subset of them. Therefore, selection of interesting patterns seems to be an important research topic.

In this paper, we concentrate on the patterns represented by the classification rules and develop an interactive rule interestingness-learning algorithm (IRIL) to automatically classify these rules or to select the interesting ones, with limited user participation. In our study, VFPI (Voting Feature Point Intervals), a feature projection based concept description learning and classification algorithm, was also developed in the framework of IRIL. Being specific to our concerns, VFPI takes the rule interestingness factors as features and is used to learn the rule interestingness concept and to classify the unlabeled classification rules. On the other hand, BCFP (Benefit Maximizing Classification using Feature Projections) classification rules learning algorithm [8] was employed to obtain the rules whose interestingness labels will be determined. In our previous related study [9], Quinlan's C4.5/C5.0 [7], an industrial quality decision tree and classification rule

* This project is supported by TUBITAK (Scientific and Technical Research Council of Turkey) under Grant 101E044.

induction technique, had been used. BCFP learns probabilistic classification rules, whereas C4.5/C5.0 learns single-class predicting classification rules. Actually, C4.5/C5.0, BCFP and VFPI are all concept description learning and classification algorithms.

The interestingness issue has been an important problem ever since the beginning of data mining research [2]. There are many factors contributing to the interestingness of a discovered pattern [2, 3, 4]. Some of them are coverage, confidence, support, completeness, unexpectedness and actionability. The first four factors are objective and the last two factors are subjective. Objective interestingness factors can be measured independently of the user. However, subjective interestingness factors are not user-independent. The value of a subjective interestingness factor may vary among users and even for the same user analyzing the discovered pattern at different times.

An objective interestingness measure is constructed by combining a proper subset of the objective interestingness factors in a suitable way. For example, objective interestingness factor x can be multiplied by the square of the objective interestingness factor y to obtain an objective interestingness measure of the form xy^2. It is also possible to use an objective interestingness factor x alone as an objective interestingness measure. (e.g. *Confidence*) Discovered patterns having *Confidence* $\geq$ *threshold* may be regarded as “interesting”. Although the user determines the threshold, this is regarded as limited user participation and the interestingness measure is still assumed to be objective.

The existing subjective interestingness measures in the literature are constructed upon unexpectedness and actionability factors. Assuming the discovered pattern to be a classification rule induced from a data set (domain), the user figures out the unexpected, expected, actionable and/or inactionable rule templates (a rule template consists of the specific features of the domain under study and their associated specific ranges of values) at the beginning. The rules are then compared with these templates. The rules having high matching values with unexpected and/or actionable templates are usually selected as the subjectively interesting rules.

Both types of interestingness measures have some drawbacks. A particular objective interestingness measure that is suitable for one domain may not be suitable for another domain. On the other hand, in the case of subjective interestingness measures, rule templates may be difficult to express. There are two reasons for this: As the number of features of a data set (domain) increases, the rule templates may become quite complex to express. Furthermore, even if the rule templates are not complex, the user may not be well in expressing his expectations/knowledge in computer languages. It would be better if the rule templates were determined automatically. The second drawback of a subjective interestingness measure is the necessity of expressing the rule templates in terms of the features of the domain. Therefore, for each domain, we need to deal with different features. It would be better to have rule templates that are expressed independent of the features of the domain. The third drawback of a subjective interestingness measure is that rules are compared with the rule templates that address the unexpectedness and/or actionability issues. Interestingness is assumed to depend on these two issues. However, it would be better if we had rule templates that dealt with the interestingness issue directly and if we benefited

from unexpectedness and actionability as two of the factors used to express the rule templates. That is, interestingness of a pattern may depend on different factors and is not fixed to just unexpectedness and actionability issues.

The idea of rule templates that are automatically determined, independent of the features of the domain and directly related with the interestingness issue motivated us to design IRIL (Interactive Rule Interestingness Learning) algorithm that learns a concept description (here concept description takes the form of rule templates), a subjective interestingness measure. To ensure that rule templates are independent of the features of the domain, factors related with the rule interestingness issue were used to construct the templates. For the time being, some existing and newly developed objective interestingness factors that have the capability to determine the interestingness of rules were used to construct these templates. That is, these objective interestingness factors took the role of the features, on which the rule templates were previously defined, of the domain. However, the interestingness factors are not just limited to the objective ones. We can also use the subjective interestingness factors.

Furthermore, the rule templates used in IRIL are directly related with the rule interestingness issue. That is, a rule template gives characteristics of either interesting or uninteresting rules.

Lastly, rule templates used in the literature are figured out at the beginning, and input rules to be labeled are compared with these templates. However, in IRIL, this is not the case. The rule templates are determined or learned automatically rather than expressing them manually at the beginning. Input rules are still compared with these templates. But, an input rule is labeled if the rule templates make labeling with high certainty. If the labeling or classification certainty factor is not sufficient, user is asked to classify the rule manually. The user looks at the interestingness factors (current version of IRIL includes only some existing and newly developed objective interestingness factors) and labels the rule accordingly. In IRIL, rule templates are learned or updated incrementally by using the interestingness labels of the rules that are on demand given either as “interesting” or “uninteresting” by the user. That is, our rule interestingness-learning algorithm is interactive.

In this paper, we deal with classification rules that were obtained using BCFP classification rules learning algorithm on TCMB data set. This data set includes information about 25632 companies that were given credit by the central bank. These classification rules are classified (labeled) either as *interesting* or *uninteresting* by the IRIL algorithm that is the focus of this paper.

TCMB Data set is described in section 2. Section 3 is devoted to the classification rules induced by the BCFP algorithm on TCMB data set. This section also explains how a rule set is constructed from the classification rules, including the knowledge representation, interestingness labels and the rule interestingness factors of this rule set. Section 4 deals with the new IRIL algorithm, explaining the VFPI (Voting Feature Point Intervals) feature projection based concept description learning and classification algorithm developed in the framework of IRIL, vote evaluation strategies, the general execution of IRIL and experimental results on TCMB data set. We conclude the paper with some remarks and suggestions for future study.

2. Central Bank of Republic of Turkey (TCMB) Data Set

TCMB data set includes information related to the companies that were given credit by the central bank. The data set includes 164 determining features, 159 of which take linear and 5 of which take nominal values, and one target feature. Some of the determining features are shown in Table 1. The target feature takes two possible values: “SUCCEED” (The company paid back the credit to the central bank) and “FAIL” (The company could not pay back the credit). TCMB data set is a comprehensive set consisting of 25632 instances.

3. Rules belonging to the TCMB Data Set

We used BCFP benefit maximizing, feature projection based classification rules learning algorithm on TCMB data set and obtained 184 classification rules. These rules were used to evaluate the success of IRIL interactive rule interestingness-learning algorithm. Any classification rule learned by BCFP distributes its votes among possible target feature values such that the sum of votes is one. Therefore, rules of BCFP are probabilistic rather than concrete.

The aim of the study presented in this paper is to select the interesting rules from the set of learned classification rules. This selection problem is modeled as a new classification problem and a *rule set* is produced for the given rules. There are a total of 184 instances in this rule set, where an instance is constructed for each corresponding rule. Each instance is represented by a vector whose components are the interestingness label of the corresponding rule and the interestingness factor values having the potential to determine the interestingness of the corresponding rule. The knowledge representation, the interestingness label concept and the interestingness factors used in the representation of the instances are explained in detail in the following section.

3.1 Knowledge Representation

IRIL tries to classify the corresponding rules of the rule set automatically according to their interestingness. Therefore, it performs a second level of data mining and makes learning of learning.

Each such classification rule produced by BCFP is in the form of ***If*** *(A_j op value)* ***(then)***→ (*SUCCEED: $vote_{SUCCEED}$*, FAIL*: $vote_{FAIL}$*) and produces just one instance. Here, A_i is any feature of the TCMB data set (such as Cari Oran, Likidite Oranı …), *op* $\in$ {=, $\neq$, <, $\leq$, >, $\geq$}, *value* is either a real number or string. The target feature value that takes the highest vote becomes the major class of that classification rule. The instance produced by the corresponding rule is shown by a vector consisting of linear or nominal determining feature values and a target feature value, just as every instance of any data set. However, as opposed to an ordinary data set, features of an instance in our rule set are in fact rule interestingness factors some of which were developed throughout the study presented in this paper. Each feature carries information about a specific property of the corresponding rule. On the other hand, the target feature value of the instance denotes the interestingness label of the associated rule.

Table 1. Some features of TCMB data set.

`	Name of Feature	Type	Code	Name of Feature	Type
F1	Borsa	Nominal	F35	CalismaSermayesiDevirHizi	Linear
F2	YilKurtar	Linear	F36	NetCalismaSermayesiDevirHizi	Linear
F3	Sector	Nominal	F37	MaddiDuranVarlikDevirHizi	Linear
F4	OzKaynak	Nominal	F38	DuranVarlikDevirHizi	Linear
F5	NetSermaye	Nominal	F39	OzKaynaklarDevirHizi	Linear
F6	DonemKar	Nominal	F40	AktifDevirHizi	Linear
F7	Yil	Linear	F41	NetKarOzKaynaklarOrani	Linear
F8	CariOran	Linear	F42	VergiOncesiKarOzKaynaklarOrani	Linear
F9	LikiditeOrani	Linear	F43	FaizVeVergiOncesiKarPasifTorani	Linear
F10	NakitOrani,Stoklar	Linear	F44	NetKarAktifToplamiOrani	Linear
F11	StoklarDönenVarliklarOrani	Linear	F45	FaaliyetKariFaaliyetin GerceklestirilmesindeKulvarlikOrani	Linear
F12	AktifToplamiOrani	Linear	F46	BirikmeliKarlilikOrani	Linear
F13	StokBagimlilikOrani	Linear	F47	FaaliyetKariNetSatislarOrani	Linear
F14	KisaVadeliAlacaklarDönen VarliklarOrani	Linear	F48	BrütSatisKariNetSatislarOrani	Linear
F15	KisaVadeliAlacaklarAktif ToplamiOrani	Linear	F49	NetKarNetSatislarOrani	Linear
F16	YabanciKaynaklarToplamiAktif ToplamiOrani	Linear	F50	SatilanMalinMaliyetiNetSatislarOrani	Linear
F17	OzKaynaklarAktifToplamiOrani	Linear	F51	FaaliyetGiderleriNetSatislarOrani	Linear
F18	OzKaynaklarYabanciKaynaklar ToplamiOrani	Linear	F52	FaizGiderleriNetSatislarOrani	Linear
F19	KisaVadeliYabanciKaynaklarPasif ToplamiOrani	Linear	F53	FaizVeVergiOncesiKarFaiz GiderleriOrani	Linear
F20	UzunVadeliYabanciKaynaklarPasif ToplamiOrani	Linear	F54	NetKarVeFaizGiderleriFaiz GiderleriOrani	Linear
F21	UzunVadeliYabanciKaynaklarDevamli SermayeOrani	Linear	F55	BankaKredileriNetSatislar	Linear
F22	MaddiDuranVarliklarOzKaynaklarOrani	Linear	F56	BrütSatisKariAktifToplami	Linear
F23	MaddiDuranVarliklarUzunVadeliYabanci KaynaklarOrani	Linear	F57	CabukDegerlerAktifToplami	Linear
F24	DuranVarliklarYabanciKaynaklar ToplamiOrani	Linear	F58	CabukDegerlerNetSatislar	Linear
F25	DuranVarliklarOzKaynaklarOrani	Linear	F59	CabukDegerlerStoklar	Linear
F26	DuranVarliklarDevamliSermayeOrani	Linear	F60	DönenVarliklarYabanciKaynakToplami	Linear
F27	KisaVadeliYabanciKaynaklarYabanci KaynaklarToplamiOrani	Linear	F61	DuranVarliklarAktifToplami	Linear
F28	BankaKredileriAktifToplamiOrani	Linear	F62	FaaliyetKariAktifToplami	Linear
F29	KisaVadeliBankaKredileriKisaVadeli YabanciKaynaklaOrani	Linear	F63	FaaliyetKariNetIsletmeSermayesi	Linear
F30	BankaKredileriYabanciKaynaklar ToplamiOrani	Linear	F64	FaaliyetKariOzKaynaklar	Linear
F31	DönenVarliklarAktifToplamiOrani	Linear	F65	FaizGiderleriYabanciKaynakToplami	Linear
F32	MaddiDuranVarliklarAktifToplamiOrani	Linear	F66	FaizGiderleriHazirDegerler MenkulKiymetler	Linear
F33	StokDevirHizi	Linear	F67	FaizVeVergiOncesiKarDuranVarliklar	Linear
F34	AlacakDevirHizi	Linear	F68	FaizVeVergiOncesiKarNetSatislar	Linear

3.2 Interestingness Labels

There are two possible interestingness labels for each rule: “interesting” and “uninteresting”. While running and evaluating the accuracy performance of the IRIL

algorithm for the classification rules of the TCMB data set, the user classified each rule either as “interesting” or “uninteresting”.

3.3 Rule Interestingness Factors

Rule set, like all data sets, has some determining features. And each feature corresponds to one of the rule interestingness factors some of which were developed in the framework of this study. There are 13 features (factors) used and Table 2 shows the names and the possible values taken by these features for TCMB data set. Among these features, ten are of linear and three are of nominal type. The rule set does not include any missing feature value.

Any rule learned by the BCFP algorithm distributes its vote, 1, among “SUCCEED” and “FAIL”. However, the target feature value that takes the highest vote becomes the major class of that classification rule and this constitutes the instance’s first feature’s value in the rule set. For instance, if a rule learned from the TCMB data set has “FAIL” as the major class, the value of the first feature of the instance associated with that rule becomes “FAIL”.

Table 2. Features of the rule set.

Feature	Values
Major Class	SUCCEED, FAIL
Major Class Frequency	[0, 1]
Rule Size	1
Confidence with respect to Major Class	[0, 1]
Coverage	[0, 1]
Support with respect to Major Class	[0, 1]
Completeness with respect to Major Class	[0, 1]
Zero Voted Class Count	0, 1
Standard Deviation of Class Votes	≥ 0
Major Class Vote	(0.5, 1]
Minor Class Vote	[0, 0.5)
Decisive	TRUE, FALSE
Strong Decisive	TRUE, FALSE

Below one of the rules produced by BCFP algorithm for the TCMB data set is given:

Rule 79:

If FaizGiderleriHazırDeğerlerMenkulKıymetler ≥ 27.25

Then (→) Major Class = “FAIL” (“SUCCEED”: 0.28, “FAIL”: 0.72)

The user labeled this rule as “uninteresting”. And the associated instance of this rule is constructed as follows:

Major Class = “FAIL”

Major Class Frequency = 0.022784

Rule Size = 1

Confidence with respect to Major Class = 0.038069

Coverage = 0.057389

Support with respect to Major Class = 0.002185

Completeness with respect to Major Class = 0.095890

Zero Voted Class Count = 0

Standard Deviation of Class Votes = 0.185522

Major Class Vote = 0.718633

Minor Class Vote = 0.281367

Decisive = “FALSE”

Strong Decisive = “TRUE”

Interestingness Label = “uninteresting”

Among the features shown above, *Major Class Frequency* gives the ratio of the instances of the data set that have “FAIL”, the value of the first feature of the instance associated with Rule 79, as the target feature value. *Rule Size* feature shows the number of conditions in the antecedent part of the rule. For a rule shown as ***If*** *(A op value)* ***(then)***→ *Major Class* = *B*:

n: total number of instances in the data set
|*A*|: number of instances satisfying the rule antecedent in the data set
|*B*|: number of instances having *B* as the target feature value in the data set
|*A*&*B*|: number of instances satisfying both by the rule antecedent and consequent in the data set

According to this information, other feature values of an instance of the rule set are determined as follows:

Confidence with respect to Major Class = |A&B| / |A|

Coverage = |A| / n

Support with respect to Major Class = |A&B| / n

Completeness with respect to Major Class = |A&B| / |B|

Zero Voted Class Count feature gives the number of target feature values that were given a zero vote by the rule. For the case of TCMB data set, this feature will either be 0 or 1 since we have two values for the target feature, namely “SUCCEED” and “FAIL”. *Standard Deviation of Class Votes* is the standard deviation of the votes of the all target feature values. Furthermore, *Major Class* and *Minor Class* features give the vote amount of the target feature value that received the highest and lowest vote, respectively.

The last two features take boolean values. A rule is decisive (*Decisive* = "TRUE") if the standard deviation of the votes is greater than s_{min}, whose definition is given below:

$$s_{min} = \frac{1}{(T\arg etFeatureValueCount - 1)\sqrt{T\arg etFeatureValueCount}}$$

If a rule distributes its vote, 1, evenly among all possible target feature values, then the standard deviation of the votes becomes zero and the rule becomes extremely undecisive. This is the worst vote distribution that can happen. The next worst vote distribution is obtained if exactly one target feature value takes a zero vote, and the all vote is distributed evenly among the remaining target feature values. The standard deviation of the votes that will occur in such a scenario is called s_{min}.

Lastly, if the difference of the two highest votes is greater than $\frac{1}{T\arg etFeatureValueCount}$, then the rule is said to be strongly decisive (*Decisive* = "TRUE").

4. IRIL Algorithm

IRIL (Interactive Rule Interestingness Learning) is an interactive, feature projection based rule interestingness-learning algorithm. Rules produced by any rule-learning algorithm (In our study, BCFP was chosen as the rule learning algorithm) are taken as input rules, and IRIL tries to classify these rules automatically according to their interestingness.

4.1 Learning Concept Description by the VFPI Algorithm

VFPI (Voting Feature Point Intervals) is a feature projection based concept description learning and classification algorithm developed in our study. It is used to learn the rule interestingness concept and to classify the unlabeled rules.

The learning phase of VFPI, given in Figure 1, is achieved incrementally. On a nominal feature, concept description is shown as the set of point intervals and the number of instances from each possible target feature value (In our TCMB case, from "interesting" and "uninteresting" interestingness labels) at each point interval. On the other hand, on a linear feature, concept description is shown as the set of infinite point intervals and the normal (gaussian) probability density functions for all target feature values (In our TCMB case, for both interestingness labels). Concept descriptions on linear features differ from our previous study presented in [9]. We were using VFI (Voting Feature Intervals) concept description learning and classification algorithm, in which concept description on a linear feature was shown as the set of range intervals and the number of instances from each possible target feature value at each range interval. However, in this paper, it is assumed that the instance space of each target feature value has a normal probability density function.

```
VFPI_train ( t )                    /* t: training instance added into the training set */
begin
    let c be the class of t
    let others be the remaining classes

    if training set = {t}           /* if t is the 1st training instance, initialize class counts */
        for each class a
            class_count [ a ] = 0

    class_count [ c ] = class_count [ c ] + 1

    for each feature f

        if f is nominal

            i = find_point_interval(f, t_f)

            if such an i exists     /* if t_f value is observed in the training set */
                point_interval_class_count [ f, i, c ] = point_interval_class_count [ f, i, c ] + 1

            else                    /* add new point interval for the nominal feature f */
                let k be the point_interval_count [ f ]
                k = k + 1
                point_interval class_count [ f, k, c ] = 1
                point_interval_class_count [ f, k, others ] = 0

        else if f is linear         /* update normal density function parameters for c */

            if training set = {t}
                μ_{f, c} = t_f
                μ_{f, others} = 0
                μ²_{f, c} = t_f²
                μ²_{f, others} = 0

            else
                μ_{f, c} = (μ_{f, c} * (class_count[c]-1) + t_f) / class_count[c]      /* update μ_{f, c} */
                μ²_{f, c} = (μ²_{f, c} * (class_count[c]-1) + t_f²) / class_count[c]   /* update μ²_{f, c} */
                update σ_{f, c}     /* the details are omitted here */

    return normal density functions (for linear features) and point intervals (for nominal features)
end.
```

Figure 1. Incremental train in VFPI.

For a nominal feature *f*, *find_point_interval* (*f*, t_f) procedure tries to find the training instance's value at feature *f* (t_f) in the concept description belonging to *f*. If t_f is found in a point interval *i*, then *point_interval_class_count* [*f*, *i*, *c*] is incremented by 1, assuming that the training instance has target feature value *c*. If t_f is not found in any point intervals, then a new point interval *n* is constructed and *point_interval_class_count* [*f*, *n*, *class*] is initialized to 1 for *class* = *c*, and to 0 for all other possible target feature values (for all *class* $\neq$ *c*). In our study, feature values used in VFPI are the interestingness factor values computed for the

classification rules, and target feature takes only two values. That is, using the above terminology, *class* = "interesting" or *class* = "uninteresting".

For a linear feature *f*, if a training instance *t* having target feature value *c* is examined, we let the previous training instances having *class* = *c* to construct a set *P* and let $\mu_{f,\ c}$ and $\sigma_{f,\ c}$ to be the mean and the standard deviation of the values of the instances in *P* on feature projection *f*, respectively. Then, $\mu_{f,\ c}$ and $\sigma_{f,\ c}$ are updated incrementally so that the previous training instances' values on *f* need not be stored anywhere. Being able to update $\sigma_{f,\ c}$ incrementally requires $\mu^2_{f,\ c}$ to be updated incrementally, too. This can easily be understood by looking at the below formula, which is a rearranged form of standard deviation calculation to be suitable for incremental update:

$$\sigma_{f,\ c} = \sqrt{\frac{\text{class\_count}[\,c\,]}{\text{class\_count}[\,c\,]-1}(\mu^2_{f,\,c} - (\mu_{f,\ c})^2)}$$

When a training instance *t* of class *c* comes in hand, the below three computations are made incrementally, also leading to the incremental update of standard deviation.

class_count [c] = *class_count* [c] + 1

$$\mu_{f,\ c} = \frac{\mu_{f,c} * (\text{class\_count}[\,c\,]-1) + t_f}{\text{class\_count}[\,c\,]}$$

$$\mu^2_{f,\ c} = \frac{\mu^2{}_{f,c} * (\text{class\_count}[\,c\,]-1) + t_f{}^2}{\text{class\_count}[\,c\,]}$$

If the training instance *t* is the first training instance, concept description parameters are initialized as follows:

$\mu_{f,\ class} = t_f$ (if *class* = *c*)

$\mu_{f,\ class} = 0$ (if *class* ≠ *c*)

$\mu^2_{f,\ class} = t_f^2$ (if *class* = *c*)

$\mu^2_{f,\ class} = 0$ (if *class* ≠ *c*)

For just one training instance, it is not possible to compute standard deviation.

Following these required updates, normal probability density functions for all target feature values are computed. Since we make interestingness analysis and just have two interestingness labels, two such functions are obtained as follows:

$$\frac{1}{\sigma_{f,\,\text{interesting}}\sqrt{2\pi}}\, e^{-\frac{(q_f - \mu_{f,\,\text{interesting}})^2}{2\sigma^2_{f,\,\text{interesting}}}} \qquad (q_f \in (-\infty, +\infty))$$

$$\frac{1}{\sigma_{f,\,\text{uninteresting}}\sqrt{2\pi}}\, e^{-\frac{(q_f - \mu_{f,\,\text{uninteresting}})^2}{2\sigma^2_{f,\,\text{uninteresting}}}} \qquad (q_f \in (-\infty, +\infty))$$

For a better understanding of concept description learning, let us look at the sample data set in Figure 2.

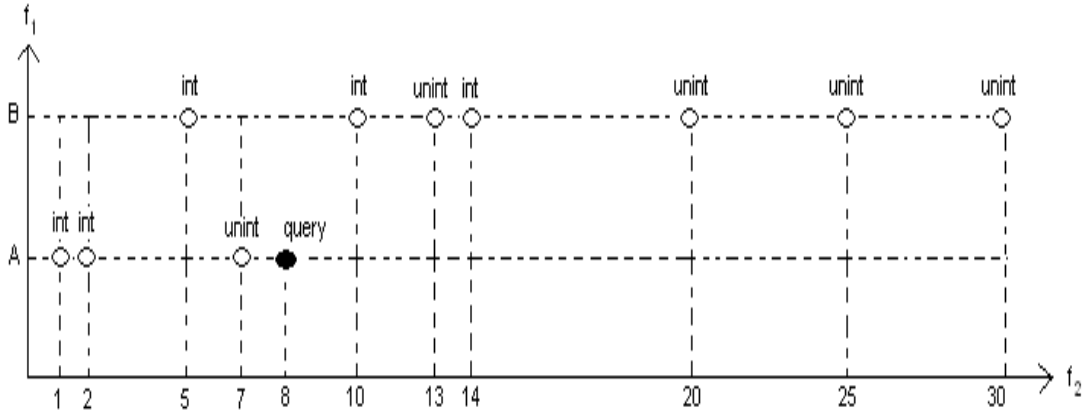

Figure 2. Sample data set.

This data set consists of 10 training instances, having one nominal (f_1) and one linear (f_2) feature. Nominal feature takes just two values “A” and “B”, whereas linear feature takes some integer values. Furthermore, target feature takes two possible values: “interesting” and “uninteresting”. In Figure 3, concept descriptions learned at both features are shown.

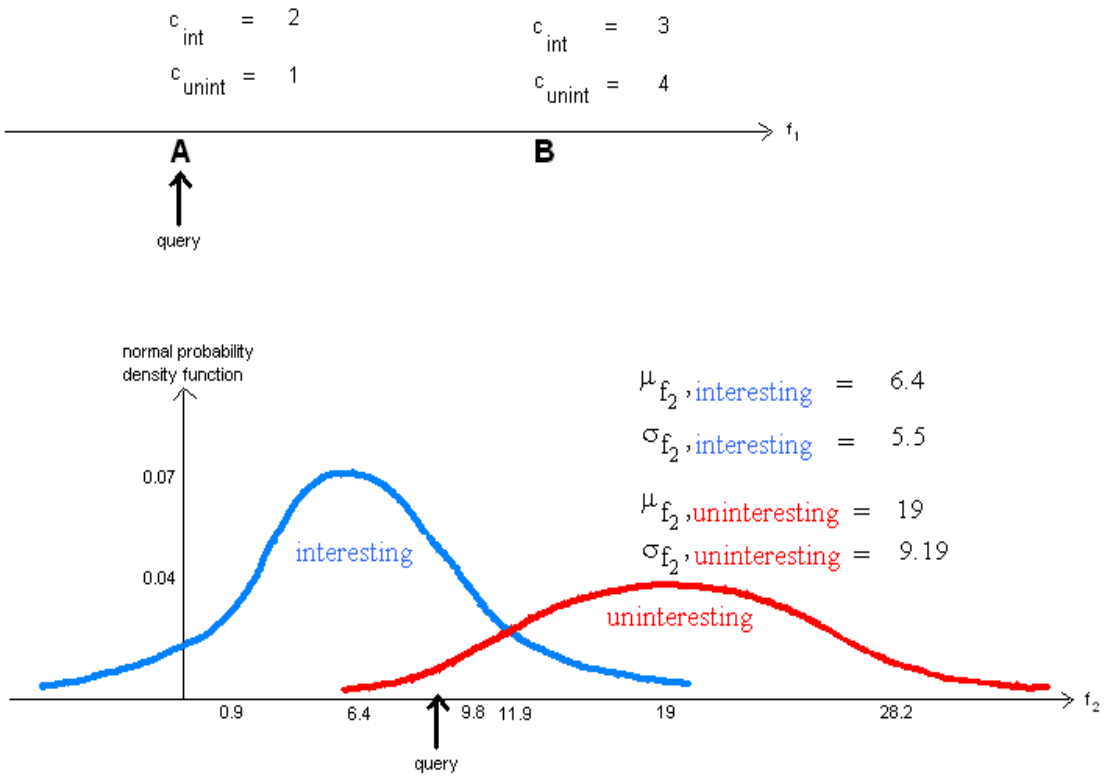

Figure 3. Concept description learned for the sample data set.

For the nominal feature, there are two point intervals. The point interval “B” includes 3 instances of class “interesting” and 4 instances of class “uninteresting”. That is:

point_interval classcount [f_1, *B*, *interesting*] = 3

point_interval classcount [f_1, *B*, *uninteresting*] = 4

For the linear feature, normal probability density functions are constructed.

4.2 Classification in the VFPI Algorithm

Classification phase of VFPI algorithm is shown in Figure 4. The query instance is projected on all features and each feature gives votes for each possible target feature values. If a feature is not ready to classification process, it gives zero, otherwise gives normalized votes. Normalization ensures that each feature has the same weight in classifying the query instances. However, if a feature is not ready, it does not involve in the classification process, therefore need not give normalized votes.

The criterion for a feature to be ready or not to be ready for the classification process was developed to ensure that we do not need any warm-up training instances, used in our previous study [9], anymore. For a feature to be ready for the classification process, it should have at least two different values for each target feature value. For example, if we look at the data set in Figure 2, f_2 linear feature has five different values for “interesting” (1, 2, 5, 10, 14), and again five different values for “uninteresting” (7, 13, 20, 25, 30), so this feature is ready. On the other hand, f_1 nominal feature has two different values for “interesting” (A, B), and again two different values for “uninteresting” (A, B), so this feature is ready, too. In the given example, it is just by chance to have same number of different values for each target feature value (Like five different values for “interesting” and “uninteresting”, or two different values for “interesting” and “uninteresting”).

The classification in the VFPI starts by giving a zero vote for each target feature value (for each class) on each feature dimension. The features that are not ready do not participate in the classification process. The participating features are handled according to their type. For a nominal feature *f*, *find_point_interval* (*f*, q_f) procedure is used to search whether q_f (query instance *q*’s value on feature *f*) exists in the set of point intervals. If q_f is found in a point interval *i*, feature *f* gives votes for each target feature value as shown below, and then these votes are normalized to ensure equal voting power among features.

$$feature\_vote\ [f, c] = \frac{point\_interval\_class\_count\ [\ f, i, c\ ]}{class\_count\ [\ c\ ]}$$

In the above equation, we divide the number of class *c* instances on point interval *i* of feature *f* by the number of total class *c* training instances to increase the vote given to minor class.

VFPI$_{query}$(*q*) /* *q*: query instance*/
begin
for each feature *f*
for each class *c*
feature_vote [*f*, *c*] = 0 /* vote of feature *f* for class *c* */

if *feature_ready_for_query_process* (*f*)

if *f* is nominal
i = *find_point_interval* (*f*, q_f)
if such an *i* exists /* if q_f value is observed in the *training set* */
for each class *c*

$$feature\_vote\ [f, c] = \frac{point\_interval\_class\_count[f,i,c]}{class\_count[c]}$$

normalize_feature_votes (*f*) /* such that $\sum_c feature\_vote[f, c] = 1$ */

else if *f* is linear
for each class *c*

$$feature\_vote\ [f, c] = \lim_{\Delta x \to 0} \frac{1}{\sigma_{f,c}\sqrt{2\pi}} e^{-\frac{(q_f - \mu_{f,c})^2}{2\sigma_{f,c}^2}} \Delta x$$

normalize_feature_votes (*f*) /* such that $\sum_c feature\_vote[f, c] = 1$ */

return *feature_vote* [*f*, *c*] for each *f*, *c* pair
end.

Figure 4. Classification in the VFPI.

For a linear feature *f*, each target feature value (class) gets the vote given in the below equation. These votes are then normalized, too.

$$feature\_vote\ [f, c] = \lim_{\Delta x \to 0} \frac{1}{\sigma_{f,c}\sqrt{2\pi}} e^{-\frac{(q_f - \mu_{f,c})^2}{2\sigma_{f,c}^2}} \Delta x$$

Assuming that a random variable *x* has the normal probability density function

$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, let f (a) = b, where a, b are real numbers. Then in the distribution of the random variable *x*, the ratio of the *x* = "a" values is not "b", but "b $\lim_{\Delta x \to 0} \Delta x$" which is actually 0. This can lead us to think that *feature_vote* [*f*, *c*] will be zero for all classes and there will be a $\frac{0}{0}$ situation in the vote normalization process. However, since "$\lim_{\Delta x \to 0} \Delta x$" term appears both in the numerator and the denominator, they cancel each other and no problem occurs.

Classification process can be better explained by an example. We use the data set in Figure 2, learned concept description in Figure 3 and the query instance shown by the <A, 8> vector. Query instance's positions on both features are shown in Figure 3.

feature_vote [f_1, *interesting*] = $\frac{2}{5}$ =0.4

feature_vote [f_1, un*interesting*] = $\frac{1}{5}$ =0.2

normalized_feature_vote [f_1, *interesting*] = $\frac{0.4}{0.4+0.2}$ = 0.67

normalized_feature_vote [f_1, un*interesting*] = $\frac{0.2}{0.4+0.2}$ = 0.33

feature_vote [f_2, interesting] = $\lim_{\Delta x \to 0} \frac{1}{5.5\sqrt{2\pi}} e^{-\frac{(8-6.4)^2}{2*5.5^2}} \Delta x$

= 0.07 $\lim_{\Delta x \to 0} \Delta x$

feature_vote [f_2, un*interesting*] = $\lim_{\Delta x \to 0} \frac{1}{9.19\sqrt{2\pi}} e^{-\frac{(8-19)^2}{2*9.19^2}} \Delta x$

= 0.02 $\lim_{\Delta x \to 0} \Delta x$

normalized_feature_vote [f_2, *interesting*] = $\frac{0.07 lim_{\Delta x \to 0} \Delta x}{0.07 lim_{\Delta x \to 0} \Delta x + 0.02 lim_{\Delta x \to 0} \Delta x}$ = 0.78

normalized_feature_vote [f_2, un*interesting*] = $\frac{0.02 lim_{\Delta x \to 0} \Delta x}{0.07 lim_{\Delta x \to 0} \Delta x + 0.02 lim_{\Delta x \to 0} \Delta x}$

= 0.22

The example consists of two classes “interesting” and “uninteresting”. Class “interesting” takes the set of votes {0.67, 0.78}, and class “uninteresting” takes the set of votes {0.33, 0.22}. The way these votes are used to predict a class for the query instance depends on the vote evaluation strategy, which will be explained in the next section. The classification phase of VFPI just gives the votes of all possible feature-class pairs.

4.3 Vote Evaluation Strategies

The classification phase of VFPI concludes by presenting votes of each feature for all possible classes. In choosing the vote evaluation strategies, we were inspired by Kittler J., and Hatef M.’s study on combining classifiers [10] and used *Select_Best_Feature_Votes*, *Select_Median_Feature_Votes* and *Use_Majority_Voting* that were studied in their paper along with *Sum_Feature_Votes* and *Use_Given_Feature* strategies that we developed ourselves. The study presented in [10] investigates the ways the predictions of different classifiers are combined to classify a query instance. The authors point out the weak points of Naïve Bayesian approach, so they make some assumptions to get rid of these weaknesses and explain the prediction-combining techniques that result from these assumptions. Our previous study presented in [9] had only used the *Sum_Feature_Votes* vote evaluation strategy.

Figure 5 shows the *Classify* procedure that makes use of vote evaluation strategies and the classification phase of VFPI algorithm ($VFPI_{query}$). Other feature

projection based classification algorithms, rather than VFPI, in which each feature distributes votes among possible classes, can also be used here. For a target feature value (class) *c*, the strategies work as follows:

Classify (*q, vote evaluation strategy*) /* *q*: query instance*/
begin

VFPI$_{query}$ (*q*)

for each class (target feature value) *c*
final_vote [c] = 0

C_f = -1

if ***vote evaluation strategy*** is *Sum_Feature_Votes*
for each class (target feature value) *c*

$$final\_vote\ [c] = \sum_{f=1}^{\#Features} feature\_vote\ [f, c]$$

else if ***vote evaluation strategy*** is *Select_Best_Feature_Votes*
for each class (target feature value) *c*

$$final\_vote\ [c] = \max_{f=1}^{\#Features} feature\_vote[f, c]$$

else if ***vote evaluation strategy*** is *Select_Median_Feature_Votes*
for each class (target feature value) *c*

$$final\_vote\ [c] = median_{f=1}^{\#Features}\ feature\_vote[f, c]$$

else if ***vote evaluation strategy*** is *Use_Majority_Voting*
for each class (target feature value) *c*
for each feature *f*

$$\text{if } feature\_vote\ [f, c] = \max_{c=1}^{\#Classes} feature\_vote[f, c]$$

final_vote [c] = *final_vote* [*c*] + 1

else if ***vote evaluation strategy*** is *Use_Given_Feature*
for each class (target feature value) *c*
final_vote [*c*] = *feature_vote* [*given_feature*, *c*]

for each class (target feature value) *c*

$$\text{if } \min_{i=1}^{\#Classes} final\_vote[i] < final\_vote\ [c] = \max_{i=1}^{\#Classes} final\_vote[i]$$

label (classify) q as "class c" with a certainty factor C_f

return C_f
end.

Figure 5. Classification with vote evaluation techniques.

Sum_Feature_Votes sums all features' votes for class *c*.

Select_Best_Feature_Votes finds the highest vote for class *c* among all features.

Select_Median_Feature_Votes sorts the votes given for class *c* by all features, and takes the median vote for class *c*.

Use_Majority_Voting employs a second level voting. It initializes the final vote for class *c* to 0, and then examines the features one by one. If class *c* takes the highest vote on a feature, final vote for class *c* is incremented by 1. Otherwise, it remains the same.

Use_Given_Feature takes the vote of the given feature for class *c*. In our TCMB rule set (also a data set, but different from TCMB data set), *Support with respect to Major Class* was chosen as the given feature (actually a rule interestingness factor).

The above processes are repeated for all class values. If there exists a class *c* that gets the highest vote and there also exists at least one other class that gets a lower vote than *c*, then class *c* is predicted to be the class of the query instance. The certainty factor of the classification (C_f) is computed as follows:

$$C_f = \frac{finalvote[c]}{\sum_{i=1}^{\#Classes} finalvote[i]}$$

If no prediction is made, certainty factor is taken as "–1" to indicate this situation. In the previous section's example, "interesting" class had {0.67, 0.78}, and "uninteresting" class had {0.33, 0.22} as the sets of votes from two determining features. Now, using *Sum_Feature_Votes* vote evaluation strategy, query instance is labeled as "interesting".

final vote [interesting] = 0.67 + 0.78 = 1.45

final vote [uninteresting]= 0.33 + 0.22 = 0.55

The certainty factor of the classification is $\frac{1.45}{1.45+0.55}$ = 72%

4.4 General Execution of IRIL Algorithm

IRIL algorithm, shown in Figure 6, needs three input parameters:

a) *R* (The set of classification rules induced by the BCFP benefit maximizing, feature projection based rule induction algorithm)

b) $MinC_t$ (Minimum certainty threshold)

c) *vote evaluation strategy*

IRIL algorithm tries to classify (in our case, tries to determine the interestingness label) the rules in *R*. If the certainty factor of the classification (C_f) is bigger than the minimum certainty threshold ($MinC_t$) for a query rule *r*, this rule is inserted into the successfully classified rules set (R_s). Otherwise, two situations are possible: either the concept description is not able to classify *r* (C_f = -1), or the concept description's classification (prediction of *r*'s interestingness label) is not of sufficient strength. If $C_f < MinC_t$, rule *r* is presented, along with its computed thirteen interestingness factor values (such as *Coverage*, *Rule Size*, *Decisive* …), to the user for classification. The core point is that user is expected to classify, or label, the rule *r* by analyzing only the given interestingness factor values of the

corresponding rule. Any other criterion in labeling the rules would degrade the accuracy performance of IRIL. Also if the user himself labels a rule, the certainty factor of this labeling will be 100% by nature. Such a rule (actually the instance holding the newly determined interestingness label and the interestingness factor values of this rule) is then inserted into the training rule set R_t and the concept description is reconstructed incrementally. In the implementation of IRIL, VFPI was used as the concept-learning algorithm. That is, *ReconstructConceptDescription* procedure, in Figure 6, is in fact the training phase of the VFPI algorithm, shown in Figure 1. Other feature projection based concept description learning algorithms, rather than VFPI, can also be used to reconstruct (update) the concept description.

```
IRIL ( R, MinC_t , vote evaluation strategy)
begin
    R_t ←∅
    R_s ← ∅

    repeat
        for each rule r ∈ R

            C_f ←Classify (r, vote evaluation strategy)

            if C_f < MinC_t
                ask the user to classify r
                set C_f of this classification to 1
                insert r into R_t
                ReconstructConceptDescription (r)

            else
                add r into R_s

            remove r from R

        for each rule r ∈ R_s

            C_f ←Classify (r, vote evaluation strategy)

            if C_f < MinC_t
                remove r from R_s
                add r into R

    until R is empty

    output rules in R_s
end.
```

Figure 6. IRIL algorithm.

All of the rules of the set R are labeled either automatically by the classification algorithm (In this paper, we developed and used VFPI), or manually by the user. User participation leads rule interestingness learning process to be an interactive one. When the number of instances in the training set increases, the concept description learned tends to be more powerful and reliable. When the labeling of

the rules ends, the rules in R_s are relabeled by the latest version of the concept description. Because, for instance, any rule *r* that was previously classified as "interesting" with a sufficient certainty factor by a weak version of the concept description may now be labeled as "interesting" with an insufficient certainty factor or possibly be labeled as "uninteresting" by the latest and the most reliable version of the concept description. Such rules are excluded from R_s and inserted into *R*. The cycle is repeated until *R* gets empty and IRIL concludes by presenting the labeled rules in R_s. The rules in R_s are labeled either as "interesting" or "uninteresting" with a certainty factor greater than the minimum certainty threshold. If the user is only interested in the interesting rules, then the rules of R_s labeled as "interesting" may be presented in a sorted order by the certainty factor to the user alone.

The concept description in the framework of IRIL is constructed incrementally by the training phase of the VFPI algorithm. The previous training instances are not reused to update the concept description when a new training instance comes. Generally, the incremental and the batch versions of an algorithm lead to different results. However, VFPI that we developed in this study does not suffer from this and constructs exactly the same concept descriptions for both incremental and batch versions.

In our previous study in [9], warm-up rules were being used to initialize the concept description. However, the number of the warm-up rules was hard to determine. Also, even if we determined the warm-up rule count, there must have been a balance between the number of rules labeled by the user as "interesting" and the number of rules labeled by the user as "uninteresting". Ensuring this balance was not enough, either. It was possible for an interestingness factor (a feature) always to predict the same interestingness label for the query rules. Therefore, we gave up using such rules and developed interestingness factors' readiness criterion for the classification process in this study. Any interestingness factor does not involve in the voting process until it is ready. It becomes ready when the user classifies sufficient number of rules manually and as a consequence; the resulting concept description becomes more reliable and powerful.

4.5 Experimental Results

IRIL (Interactive Rule Interestingness Learning) algorithm was developed and used to label the 184 rules obtained by employing BCFP (Benefit Maximizing Classifier Using Feature Projections) classification rules learning algorithm on TCMB data set. Since IRIL is an interactive algorithm, it certainly needed some user participation throughout the classification (labeling) process. However, after the execution of IRIL, we forced the user to label the rules that were successfully classified by IRIL to measure the accuracy of the algorithm on TCMB data set. The user, who is also an expert in The Central Bank, labeled 67 rules (36.41%) as "interesting" and 117 rules (63.59%) as "uninteresting". Forcing the user to classify the induced rules may not be feasible all the time. Because, many data mining applications will result in a huge number of classification rules and for these many classification rules, accuracy measurement will not be possible. However, accuracy values of the rule interestingness-learning algorithm on small sets, and on small portions of the large sets are assumed to reflect the power and reliability of this algorithm. Besides, development of an interactive rule interestingness

algorithm aims to keep the user participation low and to label the huge number of classification rules automatically.

In our experiments, we used three different minimum certainty threshold ($MinC_t$) values and gave the corresponding results in Table 3, Table 4 and Table 5.

Table 3. Results for IRIL ($MinC_t$: 51%).

	Sum Feature Votes	Select Best Feature Votes	Select Median Feature Votes	Use Majority Voting	Use "Support" Feature
Number of rules	184	184	184	184	184
Number of rules classified automatically with high certainty	154	163	172	158	172
Number of rules predicted as "INTERESTING" with high certainty	50	58	28	50	72
Number of rules classified by user	30	21	12	26	12
User participation	16%	11%	6%	14%	6%
User participation among interesting rules	21%	9%	3%	7%	3%
User participation among uninteresting rules	14%	13%	8%	18%	8%
Overall Accuracy	98%	97%	71%	92%	92%
Accuracy among interesting rules	94%	93%	34%	81%	95%
Accuracy among uninteresting rules	100%	99%	94%	100%	91%
Recall of "INTERESTING"	94%	93%	34%	81%	95%
Precision of "INTERESTING"	100%	98%	78%	100%	86%

Table 4. Results for IRIL ($MinC_t$: 53%).

	Sum Feature Votes	Select Best Feature Votes	Select Median Feature Votes	Use Majority Voting	Use "Support" Feature
Number of rules	184	184	184	184	184
Number of rules classified automatically with high certainty	128	156	171	158	158
Number of rules predicted as "INTERESTING" with high certainty	36	54	56	50	57
Number of rules classified by user	56	28	13	26	26
User participation	30%	15%	7%	14%	14%
User participation among interesting rules	42%	15%	4%	7%	6%
User participation among uninteresting rules	24%	15%	8%	18%	19%
Overall Accuracy	98%	98%	73%	92%	96%
Accuracy among interesting rules	92%	95%	58%	81%	90%
Accuracy among uninteresting rules	100%	100%	82%	100%	100%
Recall of "INTERESTING"	92%	95%	58%	81%	90%
Precision of "INTERESTING"	100%	100%	66%	100%	100%

Table 5. Results for IRIL ($MinC_t$: 55%).

	Sum Feature Votes	Select Best Feature Votes	Select Median Feature Votes	Use Majority Voting	Use "Support" Feature
Number of rules	184	184	184	184	184
Number of rules classified automatically with high certainty	112	135	171	158	157
Number of rules predicted as "INTERESTING" with high certainty	28	36	56	50	57
Number of rules classified by user	72	49	13	26	27
User participation	39%	27%	7%	14%	15%
User participation among interesting rules	54%	42%	4%	7%	6%
User participation among uninteresting rules	31%	18%	8%	18%	20%
Overall Accuracy	97%	98%	73%	92%	96%
Accuracy among interesting rules	90%	92%	58%	81%	90%
Accuracy among uninteresting rules	100%	100%	82%	100%	100%
Recall of "INTERESTING"	90%	92%	58%	81%	90%
Precision of "INTERESTING"	100%	100%	66%	100%	100%

For example, let us analyze the 2nd column of Table 3 that has the results for the *Sum_Feature_Votes* vote evaluation strategy. The user classifies 30 rules with 100% certainty, and 154 rules are classified automatically with certainty factors greater than the minimum certainty threshold. User participation, which is the ratio of the rules classified by the user, is 16% in the classification process. This ratio is a general one, however, it is also possible to compute the user participation among actually interesting and actually uninteresting rules. According to the results of the 2nd column of Table 3, the user classifies 21% of the 67 interesting and 14% of the 117 uninteresting rules. In the classification process, it is always desired that rules are generally classified automatically, and user participation among interesting rules, uninteresting rules and as a whole is low.

However, accuracy values of the automatic classifications of IRIL also play an important role. If we look at the results of the 2nd column of Table 3, accuracy

values are measured as 98%, 94% and 100% for the whole rules in R_s (overall accuracy), for the actually interesting rules in R_s (accuracy among interesting rules) and for the actually uninteresting rules in R_s (accuracy among uninteresting rules), respectively. It is important for the three accuracy values to be close to each other. For instance, if the above three accuracy values were 65%, 20% and 75%, respectively, we would easily claim that IRIL made biased classifications in favor of "uninteresting" class. Because, accuracy among uninteresting rules is too high, whereas accuracy among interesting rules is too low. Furthermore, 117 of the rules (63.59%) are uninteresting, and we could label all the rules as "uninteresting" without using IRIL that would result in an accuracy value of 63.59%, which is very close to the overall accuracy of 65%. In the experiments, all the vote evaluation strategies are generally successful, except for the *Select_Median_Feature_Votes* strategy especially for $MinC_t$ = 51%.

As mentioned in the previous sections, IRIL need not present all the rules in R_s, but only the rules automatically labeled as "interesting". This causes IRIL to be an "interesting rules learning" algorithm. In this case, the following criteria may need to be evaluated:

$$\textit{Recall}\ (\text{“interesting”}) = \frac{\#\ \text{rules that are classified as "interesting" and are actually interesting}}{\#\ \text{rules that are classified automatically and are actually interesting}}$$

$$\textit{Precision}\ (\text{“interesting”}) = \frac{\#\ \text{rules that are classified as "interesting" and are actually interesting}}{\#\ \text{rules that are classified as "interesting"}}$$

In our experiments, all the vote evaluation strategies except *Select_Median_Feature_Votes* achieved high *Recall* and *Precision* values and became successful. *Select_Median_Feature_Votes* gave low *Recall* value for $MinC_t$ = 51% and showed an average performance in terms of *Recall* and *Precision* criteria for the other minimum certainty thresholds.

If we analyze Table 3, Table 4 and Table 5, classification accuracies are quite high for all strategies except for the *Select_Median_Feature_Votes* strategy even for low $MinC_t$ values. For this strategy, average accuracy performance is obtained.

IRIL algorithm, whose learned concept description is a subjective interestingness measure, is also compared with an objective interestingness measure, namely *Confidence*. It is assumed that rules having confidence value greater than 75% are interesting, whereas the remaining ones are uninteresting. The accuracy of this objective measure is found to be 65%, a quite low accuracy value when compared to IRIL's results.

In the process of labeling the rules, user participation increases in proportion to the $MinC_t$. *Select_Median_Feature_Votes*, *Use_Majority_Voting*, and *Use_Given_Feature*, which uses *Support_with_respect_to_Major_Class* interestingness factor as the given feature, rule evaluation strategies have the smallest user participation ratios. Also in these mentioned strategies, user participation increases the least in proportion to the $MinC_t$, when compared to the other strategies.

If we evaluate the user participation, accuracy of prediction of the interestingness labels, *Recall* and the *Precision* criteria altogether, the best two strategies are found to be *Use_Majority_Voting*, and *Use_Given_Feature*, which uses *Support_with_respect_to_Major_Class* interestingness factor as the given feature.

However, requiring the user to select a feature (in our study, an interestingness factor) in classification process is not desired, leading *Use_Majority_Voting* to be the best rule evaluation strategy in IRIL algorithm.

5. Conclusion and Future Work

IRIL feature projection based, rule interestingness learning algorithm was developed and tested on TCMB data set, giving promising results. It is an interactive algorithm whose constructed concept description is actually a new subjective interestingness measure in the literature. The concept description differs among the users analyzing the same domain. That is, IRIL determines the important rule interestingness factors for a given domain subjectively.

In the framework of IRIL, a new concept description learning and classification algorithm, namely VFPI (Voting Feature Point Intervals), was also developed. It is inspired by VFI (Voting Feature Intervals) algorithm [1]. However, there are major differences such as incremental training, concept descriptions learned on linear features and the readiness criterion of features to participate in the classification process. The usage of feature projection based approaches is preferred because of their success in the previous studies and their ease of interpretability.

As future work, other concept description learning and classification algorithms other than VFPI can be developed. These need not be feature projection based algorithms, which would lead IRIL to be an interestingness-learning algorithm that isn't feature projection based. On the other hand, other objective and subjective rule interestingness factors may be investigated and used as the features of the rule sets. Finally, different and bigger data sets can be used to obtain more reliable experimental results, assuming that those data sets yield sufficient number of classification rules. If the number of classification rules gets too much, the developed rule interestingness factors may be insufficient and may require to use some rule pruning techniques, which may also be considered as a research topic.

References:

[1] Güvenir, H.A., and Demiröz, G., "**Classification by voting feature intervals**" *Proceedings of 9th European Conference on Machine Learning*, 1997, 85-92.

[2] Frawely, W.J., Piatetsky-Shapiro, G., and Matheus, C.J., "**Knowledge discovery in databases: an overview**" *Knowledge Discovery in Databases*, AAAI/MIT Press, 1991, 1-27.

[3] Major, J.A., and Mangano, J.J., "**Selecting among rules induced from a hurricane database**" *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, 1993, 30-31.

[4] Piatetsky-Shapiro, G., and Matheus, C.J., "**The interestingness of deviations**" *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, 1994, 25-36.

[5] 5- Fayyad, U., Shapiro, G., and Smyth, P., "**From data mining to knowledge discovery in databases**" *AI Magazine 17(3)*, 1996, 37-54.

[6] 6- Hilderman, R.J., and Hamilton, H.J., "**Knowledge discovery and interestingness measures: a survey**" *Technical Report*, Department of Computer Science, University of Regina, 1999.

[7] Quinlan, J.R., "**C4.5: program for machine learning**" Morgan Kaufmann, 1992.

[8] Güvenir, H.A., "**Benefit Maximization in Classification on Feature Projections**" *Proceedings of the 3rd IASTED International Conference on Artificial Intelligence and Applications* (AIA'03), Malaga, Spain (Sept. 8-10, 2003), 424-429.

[9] Güvenir, H.A., and Aydın, T., "**Feature Projection Based Rule Classification**" *Proceedings of the 12th Turkish Symposium on Artificial Intelligence and Neural Networks* (TAINN'2003), Çanakkale, Turkey (July 2-4, 2003), 652-661.

[10] Kittler, J., Hatef, M., Duin, R.P.W., and Matas, J., "**On Combining Classifiers**", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, 1998, 226-239.