ABSTRACT
In this paper, we report a deployed data mining application system for Motorola. Originally, its intended use was for identifying causes of cellular phone failures, but it has been found to be useful for many other engineering data sets as well. For this report, the case study is a data set containing cellular phone call records. This data set is like any data set used in classification applications, i.e., with a set of attributes which can be continuous or discrete, and a discrete class attribute. In our application, the classes are normally ended calls, calls which failed to setup, and calls which failed while in progress. However, the task is not to predict any failure, but to identify possible causes that resulted in failures. Then, engineering efforts may focus on improvements that can be made to the phones. In the course of the project, various classification techniques, e.g., decision trees, naive Bayesian classification and SVM were tried. However, the results were unsatisfactory. After several demonstrations and interaction with domain experts, we finally designed and implemented an effective approach to perform the task. The final system is based on class association rules, general impressions and visualization. The system has been deployed and is in regular use at Motorola. In this paper, we first describe our experiences with some existing classification systems and discuss why they are not suitable for the task. We then present our techniques. As an illustration, we show several visualization screens in the case study, which reveal some important knowledge. Due to confidentiality, we will not give specifics but only present a general discussion about the results.

Categories and Subject Descriptors
H.2.8 [Information Systems]: Data Management – Data Mining. 1.3.m [Computer Graphics]: Miscellaneous – Visualization.

General Terms
Human Factors, Management, Design.

Keywords
General impressions, visual data mining, class association rules.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.


1. INTRODUCTION
In many engineering applications, the task is to find causes of problems so that improvements can be made to the design or the manufacturing process in order to produce better products in the future. Our application belongs to this type. In the case study, Motorola’s goal was to discover possible causes of cellular phone failures, i.e., to identify situations in which all phones or even a particular model of phones are more likely to fail. Engineers can use this information to focus their efforts to determine what may be the root causes of the failures in the phone designs through further laboratory simulation and investigation.

We began this project in Aug 2004 when we were asked to build a system to solve the problem. The data set we obtained is like any classification data set, except that it is very large. It has more than 600 attributes and more than 200GB of data every month. Some of the attributes are continuous and some are categorical. One attribute indicates the final disposition of the call such as failed during setup, dropped while in progress, and ended successfully. This attribute is the usual class (target) attribute in classification, and it takes categorical/discrete values. The classes are highly skewed in the data because successfully ended calls represent a very large proportion of the data and the failure cases are rare. However, it is the rate of incidence of the failure classes that is interesting to our users. Unbalanced sampling is used but it still results in millions of data points.

Since the data set is a typical classification data set, we first tried several classification algorithms after sampling. However, none of them were satisfactory. We also tried class association rules generated by CBA [11] and displayed them using a sophisticated visualization tool which employed several rule ranking and rule query techniques. We had some success [25][26]. However, the approach still took extensive human effort to interpret the results. It was not considered satisfactory because it was not well suited for systematic exploration in order to find interesting knowledge. By working closely with a Motorola subject matter experts, we finally developed a new and very successful approach. Our system, which is called Opportunity Map, has been deployed and is in regular use at Motorola.

Our new approach is based on class association rules, general impressions and effective visualization. Class association rule mining [11] [1] produces all the rules in data to give a complete picture of the domain. This turns out to be extremely important because it allows the user to find everything that may be of interest and to see each rule in context. General impressions, which are summarized knowledge of the discovered rules, enable
the user to quickly focus on interesting and important knowledge in context. As our application shows, a rule is only interesting when it is compared with its peers or neighbors and shows a significant deviation from them. Effective visualization facilitates the whole discovery process and enables convenient exploration of the discovered knowledge.

In the rest of the paper, we will discuss why existing methods, such as classification techniques and interestingness techniques (which help the user find useful rules from a large set of rules) were not successful for our applications. We then present our new approach. After that, we use our application to demonstrate the proposed system, and describe the visualization sub-system.

2. EXPERIENCES WITH EXISTING METHODS

In this section, we discuss our experiences with several existing techniques, which were only helpful in limited ways, but were found not ultimately suitable for our applications.

2.1 Why Classification Algorithms are Unsuitable?

Since the data set is a typical classification data set, classification techniques [17][9][11] were tried first. We tried decision trees, and naive Bayesian classification (SVM [23] was also tried in Motorola, but with little success). These techniques were not successful due to the following reasons:

1. A typical classification algorithm, e.g., decision trees, rule induction, or the naive Bayesian method, only find a very small subset of the rules that exist in data. Most of the rules or patterns in data are not discovered because the objective of these algorithms is to find only enough rules for classification. Furthermore, the subset of discovered rules may not be useful for an application. Our experience was that after some rules were found the domain expert would say that certain rules did not make any sense and that some attributes should not be used at all in the rules. Then, the excluded attributes were removed, and the system was run again. When different rules were found. Then the expert said that the new rules did not make sense either. After many runs, the expert gave up, frustrated with the lack of meaningful results. We call this problem with existing classification algorithms the completeness problem.

2. Since the rules are generated for classification purposes, they usually contain many conditions to improve accuracy (except naive Bayesian). However, long rules are of limited use as the engineers do not believe that they can take any action based on a rule with many conditions. It is not possible to simulate many conditions in the laboratory to find the real causes. Furthermore, the data coverage of long rules is often so small that it is not even worth doing anything about them. We call this problem the long rules problem.

3. Individual rules have no context information. An important discovery of our interactions with domain experts is that a single rule is seldom interesting by itself no matter what its support and confidence values are. It is only interesting if it deviates significantly from the norm or from its neighbors. That is, a piece of knowledge is only interesting in a particular context. Since classification algorithms do not find all rules, there is no information about the norm or the context. We call this problem the context problem.

The key issue is the completeness problem. If the completeness problem is solved, we will be able to find short rules and give each rule a context. We tried class association rules generated using our CBA system [11] since class association rule mining aims to find all rules. Our initial system [25][26] based on such rules met some success, but it was still not sufficient.

2.2 Using Class Association Rules

Class association rules (CAR) [11] are a special kind of association rules [1] with only a class on the right-hand-side of each rule. It is of the form: \( X \rightarrow y \), where \( X \) is a set of conditions (attribute-value pairs), and \( y \) is a class. Mining of CAR rules means to generate all such rules that satisfy the user specified minimum support and confidence constraints. Thus, CAR rules help deal with the completeness problem. However, there are two issues.

1. If the minimum support is greater than zero (0), some of the rules will not be found (those rules that do not meet the minimum support constraint). Then, the context information of many rules is lost. For example, an attribute \( B \) has three possible values, \( a, b, d \). Due to the minimum support we may only find the rule \( B = a \rightarrow c \), where \( c \) is a class. The other two possible rules, \( B = b \rightarrow c \) and \( B = d \rightarrow c \), which form the context for \( B = a \rightarrow c \), may not be found because they do not satisfy the minimum support. Then, rule \( B = x \rightarrow c \) does not have a context, and it is hard to decide how interesting it is. The minimum confidence causes the same problem.

2. The number of CAR rules generated can be huge, which makes it very difficult for the user to find useful rules. For our data set, we can only find short rules before the system runs out of memory due to the large number of attributes.

There are many methods that help the user find interesting knowledge from a large set of discovered rules. Next, we discuss our experiences with some of the existing methods.

2.3 Experiences with Some Interestingness Methods

We experimented with several methods with limited success.

**Unexpectedness:** In this method, the user is asked to give some existing knowledge and the system then finds those unexpected rules [7][13][15][16][18][20][24]. This did not work well because our application is huge and the users were not sure what to expect. They wanted our system to find interesting knowledge for them. [4] studied neighborhood unexpectedness of rules. However, the neighborhood (similar to our context) is a set of syntactically similar rules, which is not applicable to our application.

**Rule ranking:** Ranking rules according to some interestingness measures [2][6][19][27] did not work either because the top ranked rules may not be interesting. Moreover, given each individual rule without a context to compare with is not useful as discussed in Section 2.1.

We also tried a few other methods, e.g., organizing rules hierarchically, querying rules, and using some rule operators, etc,
All these methods had some limited success but were not sufficient. The first version of our system was based on ranking and visualizing rules. It also organized rules hierarchically and allowed some rule querying. This initial system kept our domain experts interested in our work, but was not ready as a final product for deployment.

3. THE PROPOSED APPROACH

This section presents the proposed approach for our deployed application. We first discuss the guiding principles for our system, which we discovered from many interactions with domain experts. We then discuss how to improve class association rule mining in order to meet our requirements. Finally, we discuss how to help users find interesting knowledge based on the idea of visualizing [25][26]. Although occasionally some interesting rules were found, it was not an approach that users can work with to find interesting knowledge systematically. A different approach was needed.

After numerous interactions with domain experts, we learned the following:

1. How domain experts describe knowledge: Domain experts look for general knowledge which is broadly applicable rather than fragmented rules with little coverage of the data. For example, in our meetings they often said such things as “when the values of attribute B increase, the phone is more likely to fail”, and “attribute B is an important attribute”. We call such knowledge general impressions. When they talked about rules, they were mostly interested in short rules, seldom more than two conditions. It is impractical for them to do laboratory tests to meet the conditions of longer rules. They even expressed doubt of the validity of longer rules. Furthermore, interesting rules must represent some exceptions (see below).

2. What constitutes a piece of interesting knowledge: A piece of knowledge, e.g., a rule, is only interesting if it behaves quite differently from its peers or neighbors, or it represents an exception in a particular context. In other words, a rule is only interesting in comparison with other related rules. One cannot decide the interestingness of a rule in isolation. Thus, ranking of individual rules was a wrong approach for our application.

3. What kinds of user interactions with the system are needed: The users do not want the system to dictate what is interesting. Rather, they want to explore the data themselves and to discover knowledge in the process based on their engineering background. This means that the system should aim to find everything rather than only a subset of regularities as traditional classification systems do. The system should have a very convenient visual exploration tool. As we will see later, visualization helps users see interesting rules in context.

These critical findings form the guiding principles for us to design the new approach. The second and the third principles clearly point to class association rules, which give all the regularities in the data. However, some improvements to the existing system are needed to meet the requirements. The first point, in fact, coincides with the idea of general impressions as a form of summarized knowledge given in [10]. However, in [10], the general impressions have to be provided by the user. In this project, the system discovers them automatically from CAR rules, which constitutes meta-mining or second order data mining, i.e., mining knowledge from discovered rules (first order mining is to mine rules from the data). The general impression idea in [10] also needs to be extended. In the next subsection, we first discuss some extensions to the class association rule miner CBA.

3.2 Improving Class Association Rule Mining

To deal with the problem of finding all class association rules (CAR) to provide a context for every rule, we have to set both the minimum support and minimum confidence to 0. This causes combinatorial explosion. So a compromise is needed:

- We set the minimum support and minimum confidence to 0 in mining rules with 1 and 2 conditions, and set non-zero minimum support and minimum confidence for rules with more conditions. This ensures that the complete context information is discovered for interesting short rules. This can be done easily in a mining system such as CBA [11]. Using the minimum support and minimum confidence of 0 will not cause any memory problems for short rules. In practice, this was largely sufficient as longer rules are seldom useful. For longer rules, a high minimum support prevents too many rules from being generated.

It is also important to note that a single minimum support for the whole data set is not sufficient for our application in generating longer rules because the data is highly skewed. The most interesting classes are rare. Thus, for different classes, we need to use different minimum supports [12]. This ensures that rules are generated for rare (failure) classes without causing the frequent class to produce too many meaningless rules.

We note that CAR mining requires every attribute in the data to be discrete. This is not a problem as there are many existing discretization algorithms that can be used to discretize a continuous attribute into intervals. Manual and special data dependent discretization are supported through a simple GUI.

This flexible CAR mining framework is used in our system, which is an extension of the CBA system [11]. Now, the problem is how to present the rules to the user so that interesting knowledge can be found easily. This leads us to the mining of general expressions.

4. MINING GENERAL IMPRESSIONS

As we discussed in Section 3.1, we found that when domain experts exchange knowledge, they typically use summarized knowledge, which we call general impressions. The question is how to let a data mining system find such knowledge in the first place. Below we define some general impressions and describe how to mine them.

4.1 Preliminaries

Human users generally express their knowledge in a hierarchical manner from summaries to details. The general impressions form the summarized knowledge and the specific rules are the details. In this work, we use the following hierarchy (Figure 1).
This hierarchy has three levels representing knowledge of different granularities. The first level is called the Summarized Knowledge level. This level provides summarized information about the data. It aims to help the user locate the most interesting attributes within the context of the overall data set. It provides general impressions in the forms of discriminative attributes and trends attributes. We will discuss them in next section.

The second level is the Intermediate Knowledge level, which provides the supporting information (e.g., data counts, percentages and other numbers) to compute the summarized knowledge. All the supporting information is designed to be provided with the context to facilitate knowledge understanding. It includes the general impressions in the forms of: class distribution (CD) rules, unit trends, and exceptions. The details will be given in the next sub-section.

The lowest level contains unorganized data mining rules. Rules are basic units of knowledge providing information for the computation of general impressions at the higher levels.

General impression (GI) mining is an incremental process with respect to the length of GI. The length of a GI corresponds to the number of different attributes used in the GI. Length k GIs are mined from length k rules. In effect, a length k+1 GI can be considered as a length k GI plus one additional condition to restrict the data to a finer granularity. The first k+1 conditions are called data constraints:

Definition (data constraint): A data constraint (DC) is a set of conditions. Each condition is an attribute-value pair $A_i = a_{ij}$, where $A_i$ is an attribute and $a_{ij}$ is one of its values.

A data constraint is used to narrow down the data to be analyzed. That is, it specifies a subset of the data that satisfies the constraint. A data point satisfies a data constraint if its corresponding attributes have the corresponding specified values. DC may be empty.

For simplicity, in the following sections, we assume that the data in question is a subset of data after the constraint is applied. That is, GI can have data constraints added in front of them.

4.2 General Impression Mining

We now define and compute various types of general impressions. Each sub-section focuses on one type.

4.2.1 Exceptions and Discriminative Attributes

Our experiences show that domain experts are always interested in knowing the main attributes that affect the classes most. We call such attributes discriminative attributes. We will compute them based on exceptions of rule confidences.

Let the set of classes in the data be $C = \{c_1, c_2, ..., c_n\}$. Let the set of attributes be $A = \{A_1, A_2, ..., A_m\}$. Let the original data set be $D$, and the subset of the data be $D_{DC} \subseteq D$ after a data constraint is applied. For simplicity, we do not explicitly mention the data constraint, we only use $D_{DC}$.

Definition (expected confidence): Given the data $D_{DC}$, the expected confidence of a rule on class $c_k$, denoted by $Conf_{Exp}(c_k)$, is the class $c_k$'s prior probability in $D_{DC}$.

Let the values of attribute $A_i$ be $a_{i1}, a_{i2}, ..., a_{ip}$. We use $Conf_{Exp}(A_i = a_{ij}, c_k)$ to denote the actual or observed confidence of the rule, $A_i = a_{ij} \rightarrow c_k$. If the values of attribute $A_i$ are completely random or not related to class $c_k$, then $Conf_{Exp}(A_i = a_{ij}, c_k)$ will be to the same as $Conf_{Obs}(c_k)$. However, in reality $Conf_{Exp}(A_i = a_{ij}, c_k)$ and $Conf_{Obs}(c_k)$ can be very different, which indicates whether $A_i = a_{ij}$ is positively or negatively correlated to class $c_k$. We call such a rule $(A_i = a_{ij} \rightarrow c_k)$ an exception. We use the degree of exception to measure the difference.

Definition (degree of exception): Given the data $D_{DC}$, the degree of exception, denoted by $DE(A_i = a_{ij}, c_k)$, of the rule $A_i = a_{ij} \rightarrow c_k$ is measured by how the observed confidence is different from the expected confidence:

$$DE(A_i = a_{ij}, c_k) = |Conf_{Obs}(A_i = a_{ij}, c_k) - Conf_{Exp}(c_k)|$$

If one class has a large degree of exception for an attribute value, it is very likely that other classes will have large degrees of exceptions as well (because the observed confidence values add up to 1 for all classes) for each attribute value. So we can combine them together to produce the discriminative power of $A_i = a_{ij}$, denoted by $DP(A_i = a_{ij})$.

Definition (discriminative power of $A_i = a_{ij}$): Given the data $D_{DC}$, the discriminative power of $A_i = a_{ij}$ is defined as:

$$DP(A_i = a_{ij}) = \sum_{c_k} DE(A_i = a_{ij}, c_k) \times w_{ck}$$

where $w_{ck}$ is a parameter reflecting the importance of class $c_k$.

The discriminative power of $A_i = a_{ij}$ shows how discriminative $A_i = a_{ij}$ is with respect to all classes (weighted by $w_{ck}$). Weighting classes is important because of the skewed distribution of our data. Likewise, we can also define the discriminative power of an attribute, which shows how useful this attribute is to the classification of classes.

Definition (discriminative power of an attribute on class $c_k$): Given the data $D_{DC}$, the discriminative power of an attribute $A_i$ on class $c_k$ is the ability of the attribute to distinguish data of class $c_k$ and other classes. It is computed with:

$$DP(A_i, c_k) = \sum_{j=1}^{c_k} sup(A_i = a_{ij}, c_k) \times DE(A_i = a_{ij}, c_k)$$

where $r$ is the total number values of attribute $A_i$ and $sup(A_i = a_{ij}, c_k)$ is the support count of the rule $A_i = a_{ij} \rightarrow c_k$.

Definition (discriminative power of an attribute): Given the data $D_{DC}$, the discriminative power of attribute $A_i$ is the ability of this attribute to distinguish data of all classes:

$$DP(A_i) = \sum_{c_k} sup(A_i = a_{ij}, c_k) \times DE(A_i = a_{ij}, c_k)$$
\[ DP(A_i) = \sum_{t=1}^{n} w_{c_t} \times DP(A_i, c_t) \]

Ranking can be performed based on the results. For convenience, in applications, we call an attribute a discriminative attribute if it is ranked high by its discriminative power. We should note that these are not the only ways to compute different discriminative powers. Other methods are possible. Our current methods work quite well in our applications.

4.2.2 Trends and Trend Attributes

Another piece of summarized knowledge that domain experts often talk about is that as the values of an attribute increase/decrease, the phone is more/less likely to fail. We call such knowledge trend general impression. Note that trend is only applicable to continuous attributes and ordinal attributes.

Definition (unit trend of \( A_i \) on class \( c_j \)): Given the data \( D_{DC} \), a unit trend of attribute \( A_i \) on class \( c_j \), denoted by \( UT(A_i, c_j) = [a_{i_0}, ..., a_{i_t}, b_i, c_i] \), is a trend of the rule confidences of a set of consecutive values \([a_{i_0}, ..., a_{i_t}]\) of attribute \( A_i \) on class \( c_i \), \( b_i \) (called a trend type) is decided by a statistical test, and takes one of the types from the set \{increasing-trend, decreasing-trend, stable-trend\}. The consecutive values from \( a_{i_0} \) to \( a_{i_t} \) are such that any larger range will not have the trend \( b_i \). That is, the trend is maximal.

For an attribute \( A_i \) and class \( c_j \), there can be multiple unit trends over the values of the attribute. For example, it can have an increasing trend from value1 to value5, and a decreasing trend from value6 to value10. In applications, we require the value range of a unit trend to cover at least 3 consecutive values (or intervals). An example trend is:

In a particular salary range, as the salary of the borrower goes higher, a loan is more likely to be approved.

Here, "loan approved" is the class \( c_i \) that we are interested in. The attribute "salary" is the trend attribute \( A_i \), which shows an increasing trend \( b_i \) as its values go from small \( a_{i_0} \) to large \( a_{i_t} \).

A unit trend is derived from a set of rules based on their confidences. For example, the above simple trend may be derived from the following rules:

Rule 1: salary = 20K → loan_approved, Confidence: 20%
Rule 2: salary = 30K → loan_approved, Confidence: 35%
Rule 3: salary = 40K → loan_approved, Confidence: 40%
Rule 4: salary = 50K → loan_approved, Confidence: 45%
Rule 5: salary = 60K → loan_approved, Confidence: 56%

By using the trend relationship, all the five rules (or even more) can be summarized into a single piece of knowledge. This unit trend has the value range from 20K to 60K. Our domain experts really like this notion because that is what they are interested in and is exactly the format that they are familiar with.

The type of a trend is calculated using a statistical test called reverse arrangement test [3]. For example, to test whether it is an increasing trend, we first calculate how many times that a later value is strictly greater than an earlier value. Each time that happens, we call it a reversal. If there are a lot of reversals (more than are likely from pure chance with no trend), we have significant evidence of an increasing trend. If there are too few reversals we have significant evidence of decreasing trend. Formally, it works as follows:

1. Given \( r \) ordered values as \( v_1, v_2, ..., v_r \) for all the possible pairs, we count a reversal each time when \( i < j \) and \( v_i < v_j \). We keep the total count as reversal count \( R \).
2. For \( r \) values, the maximum possible number of reversals is \( r(r-1)/2 \). For random data, on average we would expect to have \( r(r-1)/4 \) reversals.
3. With value \( r \) and \( R \), we can look up the statistic tables [3] to decide the count whether we can conclude a significant trend in the data.

For each unit trend, we also compute two statistical properties.

Support: It is the sum of the data points this trend covers (can be the raw count). It can also be expressed as a percentage of the total number of data points in \( D_{DC} \).

Confidence: It is used to indicate how likely the trend is reliable. A value of 1.0 indicates a perfect trend of that type without exception. Otherwise, it is calculated based on how many data values and their covered data points satisfy the trend.

Confidence \( = 1.0 - \frac{Abnormal\_data\_support}{Support} \)

where Abnormal_data is the data count which does not follow the trend. We use an example to illustrate this below.

The data set \( D_{DC} \) has 1200 points, and a number of classes. The trend attribute \( A_i \) has 6 possible values. The data counts as well as the confidence values for each class are given in Table 1.

<table>
<thead>
<tr>
<th>Class</th>
<th>Data Count</th>
<th>Support</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_j )</td>
<td>200</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>200</td>
<td>200</td>
</tr>
</tbody>
</table>

For class \( c_i \), it displays an increasing trend having confidence values of 10%, 15%, 20%, 25% and 30% over the first five values. It shows an important property of the data: as the values increase, it is more likely to result in class \( c_i \). For this trend, the support count is the sum of the data counts for class \( c_i \) in its first five values, i.e., \( 20+30+40+50+60 = 200 \). The confidence is 1.0 as the increasing trend is on all five values with no exception.

Definition (trend value of an attribute on class \( c_j \)): Given the data \( D_{DC} \), the trend value of attribute \( A_i \), denoted as \( TV(A_i, b_i, c_j) \), on class \( c_j \) for trend type \( b_i \) is defined by:

\[ TV(A_i, b_i, c_j) = \sum_{t=1}^{n} \frac{Support_{UT_j} \times Confidence_{UT_j}}{Confidence_{UT_j}} \]

where \( UT_j \) is an unit trend of attribute \( A_i \) and there are \( t \) such unit trends.

We can also define the trend value for an attribute with respect to all classes.

Definition (trend value of an attribute): Given the data \( D_{DC} \), the trend value of attribute \( A_i \), denoted as \( Trend(A_i, b_i) \), for trend type \( b_i \) is defined by:
Trend($A_i, b_j$) = \sum_{i=1}^{m} TV(A_i, b_j, c_k)

where $m$ is the number of classes.

Finally, we can use the trend values (TV and Trend) to rank attributes according to different types of trends (increasing-trend, decreasing-trend or stable-trend). For convenience, in our application, we call the top ranked attributes the trend attributes.

4.2.3 Rules in Context

In the preceding sub-sections, we discussed general impressions as summarized knowledge. We now discuss general impressions that are directly related rules. We note that a rule is only interesting in comparison with its neighbors.

**Definition (class distribution rule):** Given the data $D_{DC}$, a class distribution rule (CD Rule) is a rule with all the classes on the right-hand-side (RHS) of the rule. It is expressed as

$A_i = a_{ij} \rightarrow \{c_k\} \subseteq \mathbb{C}$

A CD rule extends the concept of class association rule (CAR) [11] by showing the class distribution as the consequence. An example CD Rule is (the data set has only two classes):

$\text{salary} = 30K \rightarrow \text{loan Approved, Confidence: 35\%}
\text{not approved, Confidence: 65\%}$

The CD rule is easily displayed in a visualization screen as we will see in the next section. The user can visually compare the confidences for different classes to find interesting rules. In effect, a CD rule gives the user the necessary context information for how interesting knowledge is found in practice. The data mining is to discover possible causes of call failures and ways to further improve the performance of Motorola phones.

The user is sometimes interested in values of an attribute that behave similar to another value. This is mainly useful for an attribute with a large number of values. For example, in our application, the number of phone models is large. If the user finds a weak phone, he also wants to find other similar phones. Manually going through all the rules is difficult.

5. EVALUATION

It is difficult to have an objective measure of the effectiveness of our system. As evidence of its effectiveness, the system has been deployed in Motorola, and it is in daily use. Apart from the original application, the system has also been used by Motorola engineers to analyze other engineering data sets. This section first gives the architecture of the proposed system and then presents a case study based on the real-life application use of the system at Motorola’s Mobile Devices business unit.

5.1 System Architecture

The system architecture of Opportunity Map is given in Figure 2. Raw data are fed into the data preprocessing module. Besides data cleaning, an important task of this module is to perform attribute creation, which creates new attributes from existing attributes. For example, the user may be more interested in the ratio of two raw attributes $A_i$ and $A_j$. So a new attribute $A_i$ is created as $A_i/A_j$. Also, some attribute values are renamed from raw sensor input formats to more meaningful and user friendly numbers or strings.

Data hashing and structuring module converts and segments the data into small pieces to enable efficient scanning and retrieval. The rule miner uses a modified CBA program to mine class association rules. The extensions to CBA have been discussed in Section 3.2 (CBA has an attribute discretization sub-system). General impression miner performs all the tasks described in Section 4. The visual user interface provides a rich set of operations that enable the user to interact with the system very easily and to identify interesting knowledge quickly.

5.2 Case Study

This case study aims to show user interactions with the system and how interesting knowledge is found in practice. The data comes from Motorola mobile phone usage log. The goal of the data mining is to discover possible causes of call failures and ways to further improve the performance of Motorola phones. This is a large and high dimensional data set. It has seventy six million (76,049,178) data records. And it has about 600 attributes.

After the initial analysis by domain experts, the number of attributes was reduced to slightly more than 200, in which one attribute is the target (class) attribute. The class attribute has three values (classes), and it is highly unbalanced. The majority class covers a very large proportion of the data points. Due to confidentiality, we are unable to give the specific percentage. For
the same reason, all attributes, values, and classes in our discussion below are replaced by generic names and values. To save space, images are cropped and resized.

The visualization system supports automatic scaling among classes to address the class imbalance issue. The scaling basically increases relative proportions, and trends are preserved in the scaled visualization. Figure 3 shows the visualization of the data without scaling. Because of the highly skewed class distribution, nothing can be seen about the minority classes, which are of the greatest interest to our users. Figure 4 shows the visualization with scaling, which allows trends and distributions to be seen clearly. Scaling is a crucial step in the visualization system.

The visualization uses a matrix layout, and has two main modes, overall visualization mode, and drill down visualization mode. In the overall visualization mode (Figure 4), the X axis is associated with attributes in the data. The Y axis is associated with the classes. For each attribute (one column), each grid shows all one conditional rules of the corresponding class value. Each rule is visualized as a thumbnail bar. The height of the bar is the rule confidence value. We note that the first two levels of rules mining use the minimum support and minimum confidence of 0. Thus everything is found. Blue color is used by default. Some attributes may have many possible values (e.g., Att002, Att003, etc.) that the grid size may be inadequate to draw them all. Light blue is used to indicate this. In order to see all the values, the user can either increase the grid size, or use drill down visualization (see below). Various data counts and proportions (rule supports) are written out on the screen or on the Information Panel on the right when the user moves the mouse over the screen.

This overall visualization mode is able to summarize and show a number of important properties of the data immediately:
1. The data distribution of each attribute is illustrated by the distribution bars at the top of each column above the X axis. For the class attribute, they are on the left of the Y axis
2. For each attribute, the data distributions of its values with respect to all classes are drawn as thumbnail views in each column: each grid shows the data distribution of the attribute on one class (row).
3. One-condition class association rules are visualized for all attributes. Each rule is represented as a small bar in the visualization, with its context information: rules of all other values of the attribute (X direction in each cell) and all other classes (Y direction across rows) are visualized side by side.

4. Trends are easily detectable from the shape in each grid.

5. Strong trends are indicated using color arrows: red for decreasing, green for increasing and gray for stable trends.

6. Since the user can see the property of the data easily with a glance, any individual piece of information discovered here is shown with its context. This greatly helps understanding. After seeing the overall visualization, the user may be interested in trends and want to see which attributes are related to the
classes. For example, if the user is interested in seeing increasing trends with respect to the first class (first row, class Value4733), using a sorting command from the menu produces the visualization in Figure 5. Attributes are sorted so that those with strongest increasing trends on class Value4733 appear first on the X axis. With this screen, the user can easily confirm previous knowledge, or find interesting new knowledge quickly. For example, it clearly shows that as the values of attribute Att951 increase from small to large, the chance of failure increases (failure class Value4733). This may be a vital piece of knowledge which could be used in product design to satisfy the desired goals. Visualization of sorted trend attributes for other trend types and classes is easily viewable using similar commands.

Another sorting shows the discriminative attributes (Figure 6). Please note that the pink color is used to indicate that the scaling produced some very small values that can hardly be seen on current screen (for that grid).

The top ranked attributes all have strong discriminating powers on the classes. For example, the values of the first attribute (Att021) clearly discriminate data into different classes. Attribute Att020 also has strong discriminating power except for the middle 4 values. Our users confirmed that this kind of knowledge was much more useful than rules produced by classification systems. A classification system may only use one attribute (e.g., Att021) but miss the others which may be more useful to the user. Also crucial is the fact that the user can view all the details side-by-side. This provides contexts for visual knowledge discovery, which is shown to be crucial in practice.

The summarized information in the overall visualization is very useful in practice to get the users started and to pick the right attributes for further study using drill down visualization. In a drill down visualization, the X axis is associated with values of a given attribute. The Y axis is associated with another attribute (usually the class attribute).

Figure 7 shows a drill down visualization of one attribute (Att001, on the X axis) with all classes (the Y axis). This visualization reveals the following detailed pieces of knowledge:

1. All the class distribution (CD) rules of the given attribute. Each column is one such CD rule (corresponding to 3 class association rules in this case). For example, the CD rule corresponding to the last column looks like:

   \[
   \text{Att001}=\text{Value0232} \rightarrow \text{Att210}=\text{Value4733}, 0.5395\%
   \]

   \[
   \text{Att210}=\text{Value4734}, 0.6263\%
   \]

   \[
   \text{Att210}=\text{Value4735}, 98.8341\%
   \]

2. The exact trends of this attribute with respect to all classes (if the attribute is ordinal). It also shows the exact counts and percentages which are not shown in the overall visualization.

3. Exceptions of this attribute. The dotted dark blue horizontal lines show the expected confidences for the classes (scaling is applied). Those bars that deviate from their expected confidence represent important exceptions. Values (on X axis) can be sorted according to the degrees of exception easily from a menu command. Values can also be sorted based on their value similarity comparing to a certain value.

For each piece of knowledge discovered here, the user is able to see all the related classes (in the vertical direction) and all the related values (in the horizontal direction) from the visualization. They act as the context.

Now, suppose the user is interested in seeing what will happen if another condition is added. This can be easily visualized by adding a data constraint to the visualization so that the subset of data can be visualized and studied. Figure 8 shows the result of using "Att012 = Value0497" as the data constraint (Value0497 is a particular product model). Thus, all the information shown in this visualization has 2 attributes involved (plus the class attribute), i.e., two conditional rules. For example, for the last column, the corresponding CD rule now is:

\[
\text{Att012}=\text{Value0497}, \text{Att001}=\text{Value0232} \rightarrow
\]

\[
\text{Att210}=\text{Value4733}, 0.6612\%
\]

\[
\text{Att210}=\text{Value4734}, 1.3914\%
\]

\[
\text{Att210}=\text{Value4735}, 97.9472\%
\]

Comparing with Figure 7, we can see that adding the above data constraint affects the last column greatly. It changed the class Value4734 from below expected confidence (in Figure 7) to far above expected confidence (Figure 8). This piece of information could be used by engineers to modify product designs.

Another drill down visualization can be used for comparative study, as shown in Figure 9. It "stacks" two or more drill down visualizations by drawing the bars for each corresponding grid side by side into one grid, creating a clear contrast view in each grid. Since the Y axis is the class attribute, and the data constraints represent two different products, this visualization clearly show that the second product (Value0498) has a lower failure rate than the first one, by having consistently tall bars in the row of Value4735 (the class for “success calls”). Other rows and bars show how the products perform on different values of Att012 as well as the comparisons between them.

Please note that the Y axis can be assigned to any attribute, not only the class attribute. Due to space limitations, we are unable to show a screen-dump and an example.

The system also has another interesting feature. When it sorts the attributes using a criterion (e.g., trend value), the attributes are listed in the sorted order on the left of the screen. When the user moves the mouse over one of them, the corresponding drill down visualization is shown instantly on the right. This enables the user to quickly glance through all the attributes for details without issuing any command. Figure 10 demonstrates this idea by sorting attributes based on the increasing trend on class Value4734.

In summary, the overall visualization gives the user summarized knowledge. It shows various sorting results for different general impressions. This allows the user to pick down interesting attributes easily. The drill down visualization gives further information and enables more detailed analysis. What is crucial is that all the pieces of information and rules are shown in context, which enables fast discovery of interesting knowledge. Our users confirm that the system is able to give them significant help and enable them to easily and quickly analyze their data effectively.

Finally, note that we did not show any long rules in this section. Although they can be shown as well, since they are less useful in our application we omit them due to space limitations.
6. LESSONS LEARNED
Through this project, we learned valuable lessons. We summarize them below:

1. Working with domain experts closely. They not only provide valuable insight, but also show how knowledge in the domain is expressed. The idea of general impression mining is the result of many discussions with the domain experts. Their comments and feedback finally led us in the right direction.

2. Class association rules are very useful for finding causes of problems as such rules associate attributes directly to the target class and provide complete information. Mining with zero minimum support and zero minimum confidence for the first two levels is crucial for our application as it ensures that the complete context information is presented. This enables the user to see the full picture of knowledge in the domain.

3. Traditional classification algorithms such as decision trees, rule induction and naïve Bayesian algorithm are less effective as they only discover a very small subset of regularities in the data. Those that are not discovered may be what the user is looking for.

4. A piece of knowledge is only interesting in context. Without comparison with related rules, one cannot determine the interestingness of a rule.

5. User driving interactive knowledge discovery is the best approach. The system should only provide tools but not dictate the results that the user should see.

7. CONCLUSIONS AND FUTURE WORK
In this paper, we examined a deployed data mining system for identifying interesting knowledge from a large data set. The system is now in regular use at Motorola for finding causes of problems with mobile devices. Due to the success of the system, it is also being used for analyzing other types of data. The success of the Opportunity Map system indicates that GI mining combined with effective visualization represents a systematic approach to helping the user find interesting knowledge from a large number of rules. Our next step is to further improve the system to enhance its data pre-processing capability and to incorporate some more methods to help the user find useful knowledge quickly.

8. REFERENCES