Chapter - 3

TWO-TIER ARCHITECTURE FOR NIDS
Chapter 3
TWO-TIER ARCHITECTURE FOR NIDS

3.1 Two-tier Architecture Description

The two tier architecture for Network Intrusion Detection System (NIDS) is shown in figure 3.1. This framework is simple and easy to understand. The framework includes the pre-processes the data in detecting the intruder attacks, starting from reading data from a file which is the data obtained online and stored in a file, and preprocessing the data if data contains any missing values or noise, and reduction in data size by sampling, and selecting relevant features. This preprocessing is done in the first layer. After preprocessing the data the classification model is trained using any classification data mining technique.

The Induction Decision Tree (IDT) model has been used for classification. The training of the classification of IDT is done in first tier, and classification is done in second tier. That is tier-1 is for the training phase, and the tier-2 is the testing phase. The description of each processing step of processing in each tier is explained below:

3.1.1 Tier One Task Description

The work done to achieve the intrusions detection in the first layer is reading the training data, selecting feature subset, and training new rule-based method for classification. The KDDCUP99 DARPA dataset from MIT, U. S. A is chosen for evaluation of the model. This dataset is divided into two parts. One part of the samples is called training samples and the other part of samples is the test samples. From the training dataset the features subset is selected using infogain statistical measure.
3.1.2 Feature Selection

Feature selection from the available data is vital to the effectiveness of the methods employed. A set of features whose values in normal audit records differ significantly from the values in intrusion records is essential for having good detection performance. Data mining algorithms work more effectively if they have some amount of domain information available containing information on attributes that have higher priority than others, attributes that are not important at all, or with established relationships that are already known.
Here the entropy (information gain) method has been used for features selection. The method of selecting an attribute follow a recursive divide and conquer strategy. The kddcup'99 dataset features were listed in the table 3.1. The dataset has 41 features and one class labeled attribute. The attribute names along their data types shown in the table 3.1. The procedure for features selection is explained below.

Let given a training data \( D = \{t_1, \ldots, t_n\} \) where \( t_i = \{t_{i1}, \ldots, t_{ih}\} \) and the training data contains the following attributes \( \{A_1, A_2, \ldots, A_n\} \) and each attribute \( A_i \) contains the following attribute values \( \{A_{i1}, A_{i2}, \ldots, A_{ih}\} \). The attribute values can be discrete or continuous. Also the training data \( D \) contains a set of classes \( C = \{C_1, C_2, \ldots, C_m\} \). Each tuple in the training data \( D \) has a particular class \( C_j \). The algorithm calculates the information gain for each attribute \( \{A_1, A_2, \ldots, A_n\} \) from the training data \( D \). The attributes are sorted in descending order according the information gain.

The formulae for finding entropy (info of an attribute) is given by eq. 3.1.

\[
\text{Info} \ (D) = - \sum_{i=1}^{m} p_i \log_2 (p_i), \rightarrow \ (3.1)
\]

where \( p_i \) is the probability that an arbitrary tuple in \( D \) belongs to class \( C_i \).

The algorithm chooses one of the best attributes \( A_i \) among the attributes \( \{A_1, A_2, \ldots, A_n\} \) from the training data \( D \) with highest information gain value, and split the training data \( D \) into sub-datasets \( \{D_1, D_2, \ldots, D_n\} \) depending on the chosen attribute values of \( A_i \).

Using the equation 3.1, the attributes along their entropy is listed in table 3.2. The flowchart for features selection is shown in the figure 3.2. The procedure for the features selection explained above is shown in the flowchart.
Table 3.1 KDDCUP99 Features (Attributes) List

<table>
<thead>
<tr>
<th>S. No</th>
<th>feature name</th>
<th>Type</th>
<th>S. No</th>
<th>feature name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>duration</td>
<td>continuous</td>
<td>21</td>
<td>is_hot_login</td>
<td>discrete</td>
</tr>
<tr>
<td>2</td>
<td>protocol_type</td>
<td>discrete</td>
<td>22</td>
<td>is_guest_login</td>
<td>discrete</td>
</tr>
<tr>
<td>3</td>
<td>service</td>
<td>discrete</td>
<td>23</td>
<td>count</td>
<td>continuous</td>
</tr>
<tr>
<td>4</td>
<td>src_bytes</td>
<td>continuous</td>
<td>24</td>
<td>serror_rate</td>
<td>Continuous</td>
</tr>
<tr>
<td>5</td>
<td>dst_bytes</td>
<td>continuous</td>
<td>25</td>
<td>rerror_rate</td>
<td>Continuous</td>
</tr>
<tr>
<td>6</td>
<td>flag</td>
<td>discrete</td>
<td>26</td>
<td>same_srv_rate</td>
<td>Continuous</td>
</tr>
<tr>
<td>7</td>
<td>land</td>
<td>discrete</td>
<td>27</td>
<td>diff_srv_rate</td>
<td>Continuous</td>
</tr>
<tr>
<td>8</td>
<td>wrong_fragment</td>
<td>continuous</td>
<td>28</td>
<td>srv_count</td>
<td>Continuous</td>
</tr>
<tr>
<td>9</td>
<td>urgent</td>
<td>continuous</td>
<td>29</td>
<td>srv_serror_rate</td>
<td>Continuous</td>
</tr>
<tr>
<td>10</td>
<td>hot</td>
<td>continuous</td>
<td>30</td>
<td>srv_rerror_rate</td>
<td>Continuous</td>
</tr>
<tr>
<td>11</td>
<td>num_failed_logins</td>
<td>continuous</td>
<td>31</td>
<td>srv_diff_host_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>12</td>
<td>logged_in</td>
<td>discrete</td>
<td>32</td>
<td>dst_host_count</td>
<td>Continuous</td>
</tr>
<tr>
<td>13</td>
<td>num_compromised</td>
<td>continuous</td>
<td>33</td>
<td>dst_host_srv_count</td>
<td>Continuous</td>
</tr>
<tr>
<td>14</td>
<td>root_shell</td>
<td>discrete</td>
<td>34</td>
<td>dst_host_same_srv_rate</td>
<td>Continuous</td>
</tr>
<tr>
<td>15</td>
<td>su_attempted</td>
<td>discrete</td>
<td>35</td>
<td>dst_host_diff_srv_rate</td>
<td>Continuous</td>
</tr>
<tr>
<td>16</td>
<td>num_root</td>
<td>continuous</td>
<td>36</td>
<td>dst_host_same_src_port_rate</td>
<td>continuous</td>
</tr>
<tr>
<td>17</td>
<td>num_file_creations</td>
<td>continuous</td>
<td>37</td>
<td>dst_host_srv_diff_host_rate</td>
<td>Continuous</td>
</tr>
<tr>
<td>18</td>
<td>num_shells</td>
<td>continuous</td>
<td>38</td>
<td>dst_host_serror_rate</td>
<td>Continuous</td>
</tr>
<tr>
<td>19</td>
<td>num_access_files</td>
<td>continuous</td>
<td>39</td>
<td>dst_host_srv_serror_rate</td>
<td>Continuous</td>
</tr>
<tr>
<td>20</td>
<td>num_outbound_cmds</td>
<td>continuous</td>
<td>40</td>
<td>dst_host_rerror_rate</td>
<td>Continuous</td>
</tr>
<tr>
<td>21</td>
<td>is_hot_login</td>
<td>discrete</td>
<td>41</td>
<td>dst_host_srv_rerror_rate</td>
<td>Continuous</td>
</tr>
</tbody>
</table>

Most previous methods for feature selection emphasized only the reduction of high dimensionality of the feature space. But in cases where many features are highly redundant with each other, we must utilize other means, for example, more complex dependence models such as Bayesian network classifiers.
In my work, information gain and divergence-based feature selection methods have been used for statistical machine rule-based learning method without relying on more complex dependence models.

This feature selection method strives to reduce redundancy between features while maintaining information gain in selecting appropriate features for classification. Entropy is one of greedy feature selection methods, and conventional information gain which is commonly used in feature selection for classification models.

Moreover, our feature selection method sometimes produces more improvements of conventional machine learning algorithms over support vector machines which are known to give the best classification accuracy. The features with highest entropy have been selected. The selected features were shown in table 3.3.

3.2 Tier Two Task Description

This layer includes the testing phase of the classifier. The classifier is the new rule based classifier which is described here.

3.2.1 New Rule Based Classifier (NRBC) Model

Main idea is building a classification model using Induction Decision Tree (IDT) for normal records, and attacking records based on labelled training data, and using it to classify each new unseen record. The rules are generated from the IDT for classification. Instead of generating rule from training data, the rules can be generated from decision tree.
Figure 3.2 Flow chart for Features Selection using Entropy Method
<table>
<thead>
<tr>
<th>Rank</th>
<th>Feature Name</th>
<th>Entropy</th>
<th>Rank</th>
<th>Feature Name</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>src_bytes</td>
<td>1.73443</td>
<td>21</td>
<td>dst_host_srv_rerror_rate</td>
<td>0.29025</td>
</tr>
<tr>
<td>2</td>
<td>service</td>
<td>1.60489</td>
<td>22</td>
<td>rerror_rate</td>
<td>0.27679</td>
</tr>
<tr>
<td>3</td>
<td>count</td>
<td>1.53779</td>
<td>23</td>
<td>srv_diff_host_rate</td>
<td>0.26933</td>
</tr>
<tr>
<td>4</td>
<td>srv_count</td>
<td>1.17949</td>
<td>24</td>
<td>srv_rerror_rate</td>
<td>0.1916</td>
</tr>
<tr>
<td>5</td>
<td>dst_host_same_src_port_rate</td>
<td>1.10496</td>
<td>25</td>
<td>wrong_fragment</td>
<td>0.13855</td>
</tr>
<tr>
<td>6</td>
<td>protocol_type</td>
<td>0.98378</td>
<td>26</td>
<td>hot</td>
<td>0.13141</td>
</tr>
<tr>
<td>7</td>
<td>dst_host_diff_srv_rate</td>
<td>0.96876</td>
<td>27</td>
<td>num_compromised</td>
<td>0.08992</td>
</tr>
<tr>
<td>8</td>
<td>dst_host_srv_count</td>
<td>0.96519</td>
<td>28</td>
<td>duration</td>
<td>0.07216</td>
</tr>
<tr>
<td>9</td>
<td>dst_host_same_srv_rate</td>
<td>0.89407</td>
<td>29</td>
<td>num_failed_logins</td>
<td>0.02657</td>
</tr>
<tr>
<td>10</td>
<td>diff_srv_rate</td>
<td>0.86691</td>
<td>30</td>
<td>land</td>
<td>0.01799</td>
</tr>
<tr>
<td>11</td>
<td>dst_bytes</td>
<td>0.85391</td>
<td>31</td>
<td>root_shell</td>
<td>0.01208</td>
</tr>
<tr>
<td>12</td>
<td>same_srv_rate</td>
<td>0.83267</td>
<td>32</td>
<td>is_guest_login</td>
<td>0.00911</td>
</tr>
<tr>
<td>13</td>
<td>flag</td>
<td>0.82963</td>
<td>33</td>
<td>num_file_creations</td>
<td>0.00858</td>
</tr>
<tr>
<td>14</td>
<td>dst_host_serror_rate</td>
<td>0.64387</td>
<td>34</td>
<td>num_access_files</td>
<td>0.00734</td>
</tr>
<tr>
<td>15</td>
<td>logged_in</td>
<td>0.63707</td>
<td>35</td>
<td>num_root</td>
<td>0.00637</td>
</tr>
<tr>
<td>16</td>
<td>serror_rate</td>
<td>0.62279</td>
<td>36</td>
<td>num_outbound_cmds</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>dst_host_srv_serror_rate</td>
<td>0.51385</td>
<td>37</td>
<td>is_host_login</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>dst_host_count</td>
<td>0.5064</td>
<td>38</td>
<td>urgent</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>srv_serror_rate</td>
<td>0.46685</td>
<td>39</td>
<td>num_shells</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>dst_host_srv_diff_host_rate</td>
<td>0.45478</td>
<td>40</td>
<td>su_attempted</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>dst_host_rerror_rate</td>
<td>0.35237</td>
<td>41</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
Table 3.3 Selected features

<table>
<thead>
<tr>
<th>S.No</th>
<th>Selected Feature</th>
<th>S.No</th>
<th>Selected Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>src_bytes</td>
<td>9</td>
<td>dst_host_same_srv_rate</td>
</tr>
<tr>
<td>2</td>
<td>service</td>
<td>10</td>
<td>diff_srv_rate</td>
</tr>
<tr>
<td>3</td>
<td>count</td>
<td>11</td>
<td>dst_bytes</td>
</tr>
<tr>
<td>4</td>
<td>srv_count</td>
<td>12</td>
<td>same_srv_rate</td>
</tr>
<tr>
<td>5</td>
<td>dst_host_same_src_port_rate</td>
<td>13</td>
<td>flag</td>
</tr>
<tr>
<td>6</td>
<td>protocol_type</td>
<td>14</td>
<td>dst_host_serror_rate</td>
</tr>
<tr>
<td>7</td>
<td>dst_host_diff_srv_rate</td>
<td>15</td>
<td>logged_in</td>
</tr>
<tr>
<td>8</td>
<td>dst_host_srv_count</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The time can be reduced for classification instead of using IDT Model for classification. Rule based Classification Model [63,64] must be able to handle skewed (imbalanced) class distributions. This is also called \textit{supervised classification technique} [65] because it requires knowledge of both \textit{normal} and \textit{attack} classes.

Information produced by data mining techniques can be represented in many different ways. Decision tree structures are a common way to organize classification schemes. In classifying tasks, decision trees visualize what steps are taken to arrive at a classification. Every decision tree begins with what is termed a root node, considered to be the "parent" of every other node. Each node in the tree evaluates an attribute in the data and determines which path it should follow.

Rules are a good way of representing information or bits of knowledge. A new rule-based classifier [83] uses a set of IF-THEN rules for classification. An IF-THEN rule is an expression of the form
IF condition THEN conclusion.

An example is rule R1,

R1: IF age = youth AND student = yes THEN buys_computer = yes.

To build a rule-based classifier, it needs to extract a set of rules that show the relationships between the attributes of a dataset and the class label. Each classification rule is of the following form:

\[ R : (\text{condition}) \Rightarrow y. \]

Here the condition is called the rule antecedent, which is a conjunction of the attribute test condition, and \( y \) is called the rule consequent, and it is the class label. A rule set can consist of multiple rules

\[ RS = \{ R_1, R_2, \ldots, R_n \} \]

A rule \( R \) can be assessed by its coverage and accuracy. Given a tuple, \( X \), from a class labeled data set, \( D \).

Let \( n_{\text{cover}} \) be the number of tuples covered by \( R \); \( n_{\text{correct}} \) is the number of tuples correctly classified by \( R \), and \( |D| \) be the number of tuples in \( D \). The coverage and accuracy of \( R \) can be defined as

\[
\text{Coverage (R)} = \frac{n_{\text{cover}}}{|D|} \quad (3.2)
\]

\[
\text{Accuracy (R)} = \frac{n_{\text{correct}}}{n_{\text{cover}}} \quad (3.3)
\]

That is, a rule’s coverage is the percentage of tuples that are covered by the rule. For a rule’s accuracy, we look at the tuples that it covers and see what percentage of them the rule can correctly classify.
The New Rule Based algorithm is shown in Algorithm 1. It uses the sequential covering approach to extract rules from the data. The algorithm extracts the rules one class at a time for a data set. Let \((y_1, y_2, \ldots, y_n)\) be the ordered classes according to their frequencies, where \(y_1\) is the least frequent class and \(y_n\) is the most frequent class.

During the \(i\)'th iteration, instances that belong to \(y_1\) are labeled as positive examples, while those that belong to other classes are labeled as negative examples. The procedure for New Rule-Based classification algorithm is shown in figure 3.3.

A rule is desirable if it covers most of the positive examples and none of the negative examples. Our Rule-based learning algorithm is based on the RIPPER algorithm [69], which was introduced by Cohen and considered to be one of the most commonly used rule based algorithm in practice, which is given in figure 3.4.

The Learn_One_Rule() procedure, shown in figure 3.5, is the key function of the Rule-based algorithm. It generates the best rule for the current class, given the current set of uncertain training tuples. The Learn_One_Rule() includes two phases:

- Growing rules, and
- Pruning rules.

In the first phase, growing rules, in more detail, while the other pruning rules is similar to regular rule-based classifier [70], thus will not be elaborated. After generating a rule, all the positive and negative examples covered by the rule are eliminated. The rule is then added into the rule set as long as it does not violate the stopping condition, which is based on the minimum description length (DL) principle.
Rule also performs additional optimization steps to determine whether some of the existing rules in the rule set can be replaced by better alternative rules.

 NRBC Algorithm 1

Input: Training Dataset D
Output: Intrusion Detection Model
Procedure:

1. Calculate the information gain for each attributes $A_i = \{A_1, A_2, \ldots, A_n\}$ from the training data $D$ using equation (3).
2. Choose an attribute $A_i$ from the training data $D$ with the maximum information gain value.
3. Split the training data $D$ into sub-datasets $\{D_1, D_2, \ldots, D_n\}$ depending on the attribute values of $A_i$.
4. Calculate the prior $P(C_j)$ and conditional probabilities $P(A_{ij} | C_j)$ of each sub-dataset $D_i$.
5. Classify the examples of each sub-dataset $D_i$ with their respective prior and conditional probabilities.
6. If any example of sub-dataset $D_i$ is misclassified then again calculate the information gain of attributes of sub-dataset $D_i$, choose the best attribute $A_i$ with maximum information gain value from sub-dataset $D_i$, split the sub-dataset $D_i$ into sub-sub-datasets $D_{ij}$ and again calculate the prior and conditional probabilities for each sub-sub-dataset $D_{ij}$. Finally, classify the examples of sub-sub-datasets using their respective prior and conditional probabilities.
7. Continue this process until all the examples of sub/sub-sub-datasets are correctly classified.

Figure 3.3 New Rule-Based Classification (NRBC) Algorithm

Rule also performs additional optimization steps to determine whether some of the existing rules in the rule set can be replaced by better alternative rules.
The process of growing rules, Grow(), is shown in figure 3.6. The basic strategy is as follows:

1. It starts with an initial rule: $\emptyset \rightarrow y$, where the left hand side is an empty set and the right hand side contains the target class. The rule has poor quality because it covers all the examples in the training set. New conjuncts will subsequently be added to improve the rule’s quality.

2. The probabilistic information gain is used as a measure to an antecedent of the rule (steps 3-4). The details of how to compute probabilistic information gain for uncertain data will be shown in the next section.

**RIPPER RULE BASED Algorithm 2**

Input : Dataset D, and ClassSet C

begin

1) Rule_set = $\emptyset$; //initial set of rules learned is empty
2) for each class C do
3) repeat
4) Rule = Learn_One_Rule(D, C);
5) Remove tuples covered by Rule from DataSet D;
6) until terminating condition;
7) Rule_set += Rule;
8) end for;
9) return Rule_set;
10) end

Figure 3.4 New Rule-Based Classification Algorithm Based on RIPPER
3. If an instance is covered by the rule, Function splitUncertain(), is invoked (steps 5-9). Function splitUncertain() returns part of the instance that is covered by the rule. Then, the part of the instance that is covered by the rule is removed from the dataset, and the rule growing process continues, until either all the data are covered or all the attributes have been used as antecedents.

Function splitUncertain() is shown in figure 3.7. As the data is uncertain, a rule can partly cover an instance.
Function splitUncertain() computes what proportion of the instances is covered by a rule based on the uncertain attribute interval and probabilistic distribution function.

Rule employs a general-to-specific strategy to grow a rule and the probabilistic information gain measure to choose the best conjunct to be added into the rule antecedent. The new rule is then pruned based on its performance on the validation set. The following metric has been used for rule pruning.

Growing phase Algorithm 4

Input: Instances growData

begin
1: coverData = { };
2: while (growData.size() > 0) (numUnusedAttributes > 0) do
3: Find the attribute Ai and the split point sp, which has the highest probabilistic information gain;
4: Antecedent += RuleAntecedent(Ai, sp);
5: for (each instance Ij) do
6: if (covers(Ij)) then
7: inst = splitUncertain(Ij, Ai, sp);
8: coverData += inst;
9: end if;
10: end for;
11: growData -= coverData;
12: end while; End

Figure 3.6 Grow Algorithm
The **probabilistic prune** for a rule $R$ is

$$\text{ProbPrune}(R, p, n) = \frac{\text{PC}(p) - \text{PC}(n)}{\text{PC}(p) + \text{PC}(n)}$$
Here PC(p) and PC(n) is the probabilistic cardinality of positive and negative instances covered by the rule. This metric is monotonically related to the rule’s accuracy on the validation set. If the metric improves after pruning, then the conjunct is removed. Like RIPPER, Rule starts with the most recently added conjunct when considering pruning. Conjuncts are pruned one at a time as long as this results in an improvement.

Please note that for uncertain data, a rule may partly cover instances, therefore, the number of positive and negative instances covered by a rule are no longer integers but real values. The second rule is a default rule. Like traditional rule-based classifier, NRBC also generate a default rule, which can be applied to instances which do not match any rules in the rule set. This default rule has accuracy around 91%.

To classify a test sample in the dataset, the algorithm estimates the likelihood that $e_i$ is in each class. The probability that $e_i$ is in a class is the product of the conditional probabilities for each attribute value with prior probability for that class. The posterior probability $P(C_j | e_i)$ is then found for each class and the example classifies with the highest posterior probability for that example.

The algorithm will continue this process until all the examples of sub-datasets or sub-sub-datasets are correctly classified. When the algorithm correctly classifies all the examples of all sub/sub-sub-datasets, then the algorithm terminates and the prior and conditional probabilities for each sub/sub-sub-datasets are preserved for future classification of unseen examples. The main procedure of proposed algorithm is described as follows.
3.2.2 Pruning Unnecessary Conditions

If there are conditions of that rule that are inconsequential to the outcome, discard them thus simplifying the rule (and thus improving efficiency). This is accomplished by proving that the outcome is independent of the given condition. Events A and B are independent if the probability of event B does not change given that event A occurs. Using Bayes Rule:

\[ P(B|A) = P(B) \]

This states that the probability of event B given that event A occurs is equal to the probability that event B occurs by itself. If this holds true, then event A does not effect whether or not event B occurs. If A is a condition and B is a result, then A can be discarded without affecting the rule.

3.2.3 Pruning Unnecessary Rules

If two or more rules share the same end result, you may be able to replace them with a rule that fires in the event that no other rule is fired:

If (no other rule fires) Then (execute these common actions)

If there is more than one such group of rules, replace only one group. Which one is determined by some heuristic tiebreaker. Two such tiebreakers follow:

Replace the larger of the two groups. If group A has six rules which share a common result and group B only has five, replace the larger group A with will eliminate more rules and simplify the rule base the most.
Replace the group with the highest average number of rule conditions. While more rules may remain, the rules that remain will be simpler as they have fewer conditions.

### 3.2.4 Rule Prediction

Once the rules are learned from a dataset, they can be used for predicting class types of unseen data. Like a traditional rule classifier, each rule of Rule_set is in the form of, "IF Conditions THEN Class = C_i".

Because each instance I_i can be covered by several rules, a vector can be generated for each instance \( P(I_i, C) = (P(I_i, C_1), P(I_i, C_2), \ldots, P(I_i, C_n)) \) in which \( P(I_i, C_j) \) denotes the probability for an instance to be in class \( C_j \). It can be called as Class Probability Vector (CPV). As an uncertain data instance can be partially covered by a rule, and denoted as the degree an instance \( I \) covered by a rule \( R_j \) by \( P(I, R_j) \).

When \( P(I, R_j) = 1 \), \( R_j \) fully covers instance \( I \); when \( P(I, R_j) = 0 \), \( R_j \) does not cover \( I \); and when \( 0 < P(I, R_j) < 1 \), \( R_j \) partially covers \( I \).

An uncertain instance may be covered or partially covered by more than one rule. I allow a test instance to trigger all relevant rules. The \( w(I, R_k) \) can be used to denote the weight of an instance \( I \) covered by the \( k \)th rule \( R_k \). The weight of an instance \( I \) covered by different rules is as follows:

\[
\begin{align*}
    w(I_i, R_1) &= I_i.w \times P(I_i, R_1) \\
    w(I_i, R_2) &= (I_i.w - w(I_i, R_1)) \times P(I_i, R_2) \\
    \vdots
    w(I_i, R_n) &= (I_i.w - \sum_{k=1}^{n-1} w(I_i, R_k)) \times P(I_i, R_n)
\end{align*}
\]
For the first rule $R_1$ in the rule set, $w(I_i, R_1)$ should be the weight of the instance, $I_i.w$, times the degree the instance covered by the rule, $P(I_i, R_1)$. For the second rule $R_2$, $w(I_i, R_2)$ is the remained probability cardinality of instance $I_i$, which is $(I_i.w - w(I_i, R_1))$, times the rule coverage, $P(I_i, R_2)$. Similarly, $w(I_i, R_n)$ should be the remained probability cardinality of instance $I_i$, which is $I_i.w - \sum_{k=1}^{n-1} w(I_i, R_k)$, times $P(I_i, R_n)$.

Suppose an instance $I_i$ is covered by $m$ rules, then it class probability vector $P(I_i, C)$ is computed as follows:

$$P(I_i, C) = \sum_{k=1}^{m} P(R_k, C) \times w(I_i, R_k),$$

where $P(R_k, C)$ is a vector $P(R_k, C) = (P(R_k, C_1), P(R_k, C_2),..., P(R_k, C_n))^T$ and denotes the class distribution of the instances covered by rule $R_k$. $P(R_k, C_i)$ is computed as the fraction of the probabilistic cardinality of instances in class $C_i$ covered by the rule over the overall probabilistic cardinality of instances covered by the rule.

After computing the CPV for instance $I_i$, the instance will be predicted to be of class $C_j$, which has the largest probability in the class probability vector. This prediction procedure is different from a traditional rule based classifier.

When predicting the class type for an instance, a traditional rule based classifier such as RIPPER usually predicts with the first rule in the rule set that covers the instance. As an uncertain data instance can be fully or partially covered by multiple rules, the first rule in the rule set may not be the rule that covers it best. New Rule will use all the relevant rules to compute the probability for the instance to be in each class and predict the instance to be the class with the highest probability.
3.3 NRBC Model Ruleset

Using the above procedures the rule set is generated for each class type. The DARPA KDDCUP99 dataset has been used for evaluation of the RBC model. The rules for each class are given below.

3.3.1 U2R Class Ruleset

(service = telnet) and (src_bytes >= 135) THEN class=u2r
(service = ftp_data) and (dst_bytes >= 2072) and (dst_bytes <= 5928) THEN class=u2r
(service = telnet) and (dst_bytes >= 183) and (dst_bytes <= 233) THEN class=u2r

3.3.2 R2L Class Ruleset

(service = ftp_data) and (flag = SF) THEN class=r2l
(service = telnet) and (src_bytes >= 112) THEN class=r2l
(service = ftp) and (src_bytes <= 119) and (src_bytes >= 36) THEN class=r2l
(count <= 1) and (service = login) THEN class=r2l
(count <= 1) and (dst_bytes >= 2551) and (src_bytes <= 51) THEN class=r2l

3.3.3 Probe Class Ruleset

(count >= 309) and (src_bytes <= 6) THEN class=probe
(service = eco_i) and (src_bytes <= 20) THEN class=probe
(flag = REJ) and (count >= 3) THEN class=probe
(flag = SH) THEN class=probe
(src_bytes >= 1) and (src_bytes <= 6) and (count >= 3) THEN class=probe.
(flag = RSTR) and (src_bytes <= 0) THEN class=probe
3.3.4 Normal Class Ruleset

(dst_bytes >= 32) and (src_bytes <= 12804) THEN class=normal.
(count <= 1) and (service = smtp) THEN class=normal.
(count <= 2) and (src_bytes <= 1476) and (src_bytes >= 6) THEN class=normal.
(flag = REJ) THEN class=normal.

3.4 Performance Evaluation of NRBC

The performance results for RBC model are shown in Table 3.4. The model is tested on DARPA kddcup99 dataset on 22,213 test records. The ten-fold cross fold validation is used to classify the records.

3.4.1 Purpose of Cross Validation

Suppose we have a model with one or more unknown parameters, and a data set to which the model can be fit (the training data set). The fitting process optimizes the model parameters to make the model fit the training data as well as possible.

If we then take an independent sample of validation data from the same population as the training data, it will generally turn out that the model does not fit the validation data as well as it fits the training data. This is called overfitting, and is particularly likely to happen when the size of the training data set is small, or when the number of parameters in the model is large.

Cross-validation is a way to predict the fit of a model to a hypothetical validation set when an explicit validation set is not available.
Sometimes it is called rotation estimation. It is a technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the *training set*), and validating the analysis on the other subset (called the *validation set* or *testing set*).

To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds. Here I have implemented 10-fold Cross-validation method. The algorithm for 10-fold cross validation is figure 3.8.

In the experiment, ten-fold cross validation has been used. Data is split into 10 approximately equal partitions; each one is used in turn for testing while the rest is used for training, that is, 9/10 of data is used for training and 1/10 for testing.

The whole procedure is repeated 10 times, and the overall accuracy rate is counted as the average of accuracy rates on each partition. The procedure for cross-validation is shown in figure 3.8.

The accuracy of the RBC is 99.7933% that is 22213 records correctly classified out of 22259 records and error rate is 0.2067% that is only 46 records is incorrectly classified against the same number of records.
3.4.2 Receiver Operating Characteristic (ROC) Curves

The ROC approach [144,145] to evaluating predictive ability of classifiers provides an intuitive and convenient way of dealing with asymmetric costs of the two types of errors. ROCs are plotted in coordinates spanned by the rates of false positive and true positive classifications.

True positive (TP) refers to the positive tuples that were correctly labeled by the classifier, True negatives (TN): refers to the negative tuples that were correctly labeled by the classifier, False positives (FP): are the negative tuples that were incorrectly labeled.

There are various parameters to evaluate the performance of the classification model. Some of the parameters are defined below.
True positive rate (TPR) = \[ TPR = \frac{TP}{TP + FN} \]

False positive rate (FPR) = \[ FPR = \frac{FP}{FP + TN} \]

### 3.4.3 Confusion Matrix

The other parameter is the confusion matrix [142,143] is more commonly named contingency table. In DARPA KDDCUP there are four class types, and therefore we need a 4x4 confusion matrix, the matrix could be arbitrarily large. The number of correctly classified instances is the sum of diagonals in the matrix; all others are incorrectly classified.

The True Positive (TP) rate (TPR) is the proportion of examples which were classified as class x, among all examples which truly have class x, i.e. how much part of the class was captured. It is equivalent to recall or sensitivity.

The False Positive (FP) rate (FPR) is the proportion of examples which were classified as class x, but belong to a different class, among all examples which are not of class x. In the matrix, this is the column sum of class x minus the diagonal element, divided by the rows sums of all other classes. The Precision for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class.

\[ Pr = \frac{TP}{TP + FP} \]

*F-Measure* \((F_v)\) [141] combines the true positive rate (recall) and precision \(Pr\) into a single utility function which is defined as \(\nu\)-weighted harmonic mean.

\[ F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \]
Using above all, the results of the RBC model is shown in Tables 3.4 and 3.5.

Table 3.4  RBC Classification results in Confusion Matrix

<table>
<thead>
<tr>
<th>Class</th>
<th>Normal</th>
<th>Dos</th>
<th>Probe</th>
<th>R2l</th>
<th>U2r</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>13437</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dos</td>
<td>3</td>
<td>7759</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Probe</td>
<td>5</td>
<td>7</td>
<td>858</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R2l</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>116</td>
<td>6</td>
</tr>
<tr>
<td>U2r</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>43</td>
</tr>
</tbody>
</table>

Table 3.5 RBC Classification Detailed Accuracy by Class

<table>
<thead>
<tr>
<th>Class</th>
<th>TP_Rate</th>
<th>FP_Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F_Measure</th>
<th>ROCArea</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>1</td>
<td>0.001</td>
<td>0.999</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DoS</td>
<td>0.999</td>
<td>0.002</td>
<td>0.997</td>
<td>0.999</td>
<td>0.998</td>
<td>0.999</td>
</tr>
<tr>
<td>Probe</td>
<td>0.986</td>
<td>0</td>
<td>0.998</td>
<td>0.986</td>
<td>0.992</td>
<td>0.992</td>
</tr>
<tr>
<td>R2l</td>
<td>0.906</td>
<td>0</td>
<td>0.959</td>
<td>0.906</td>
<td>0.932</td>
<td>0.986</td>
</tr>
<tr>
<td>U2r</td>
<td>0.811</td>
<td>0</td>
<td>0.827</td>
<td>0.811</td>
<td>0.819</td>
<td>0.933</td>
</tr>
</tbody>
</table>
Figure 3.9  ROC Curve for R2L Class

Figure 3.10  ROC Curve for U2R Class
This model has been applied on KDDCUP’99 Dataset with sample size of 22,259 records. The steps as per the framework have been followed and applied and intermediate results were not shown. The final results for new rule based model for accuracy shown in confusion matrix tables 3.4 and 3.5 as summary, and graphs of ROC curves shown in figures 3.9 and 3.10.

These results show that the accuracy for normal classes normal and DoS type records around 0.99 close to one, where as for other two classes it drops to 0.98 and 0.90 because records are only a few. The area under ROC is one for ideal model where the model predicts with 100% accuracy. The results in the confusion matrices are shown in tables 3.4 and 3.5, the diagonal column indicates the number of records correctly predicted for each class type, and the remaining entries in that matrix are wrongly predicted records.

3.5 Chapter Summary

The model performance is around 99% with reduced dataset features. The reduction in features results in reduced model learning time, execution time in prediction, and reduces the space requirement.

The model, I proposed is a new rule-based algorithm for classifying and predicting kddcup’99, dataset records for classification. I propose new approaches for deriving optimal rules from data, pruning and optimizing rules, and class prediction for data records. This model follows the new paradigm of directly mining kddcup dataset. The avenues of future work include developing uncertain data mining techniques for various applications, including sequential pattern mining, association mining, spatial data mining and web mining, where data is usually uncertain.