Another approach to generating rules without first having a DT is called PRISM. PRISM generates rules for each class by looking at the training data and adding rules that completely describe all tuples in that class. Its accuracy is 100%. Example 0.1 illustrates the use of PRISM. Algorithm 0.1 adapted from [?] shows the process. Note that the algorithm refers to attribute-value pairs. As with earlier classification techniques, this needs to be modified to handle continuous attributes. In the example, we have again used the ranges of height values used in earlier examples.

Example 0.1 Using the data in Table ?? and the Output1 classification, the following shows the basic probability of putting a tuple in the Tall class based on the given attribute-value pair:

\[
\begin{align*}
\text{Gender} = F & \quad 0/9 \\
\text{Gender} = M & \quad 3/6 \\
\text{Height} \leq 1.6 & \quad 0/2 \\
1.6 < \text{Height} \leq 1.7 & \quad 0/2 \\
1.7 < \text{Height} \leq 1.8 & \quad 0/3 \\
1.8 < \text{Height} \leq 1.9 & \quad 0/4 \\
1.9 < \text{Height} \leq 2.0 & \quad 1/2 \\
2.0 < \text{Height} & \quad 2/2
\end{align*}
\]

Based on this analysis we would generate the rule:

If 2.0 < Height then Class = Tall.

Since all tuples which satisfy this predicate are tall, we do not add any additional predicates to this rule. We now need to generate additional rules for the Tall class. We thus look at the remaining 13 tuples in the training set and recalculate the accuracy of the corresponding predicates:

\[
\begin{align*}
\text{Gender} = F & \quad 0/9 \\
\text{Gender} = M & \quad 1/4 \\
\text{Height} \leq 1.6 & \quad 0/2 \\
1.6 < \text{Height} \leq 1.7 & \quad 0/2 \\
1.7 < \text{Height} \leq 1.8 & \quad 0/3 \\
1.8 < \text{Height} \leq 1.9 & \quad 0/4 \\
1.9 < \text{Height} \leq 2.0 & \quad 1/2
\end{align*}
\]

Based on the analysis we see that the last height range is the most accurate and thus generate the rule:

If 2.0 < Height then Class = Tall.

However only one of the tuples that satisfies this is actually tall, so we need to add another predicate to it. We then look only at the other predicates affecting these two tuples. We now see a problem in that both of these are males. The problem is actually caused by our “arbitrary” range divisions. We now divide the range into two subranges:

\[
\begin{align*}
1.9 < \text{Height} \leq 1.95 & \quad 0/1 \\
1.95 < \text{Height} \leq 2.0 & \quad 1/1
\end{align*}
\]

We thus add this second predicate to the rule to obtain:

If 2.0 < Height and 1.95 < Height <= 2.0 then Class = Tall

or

If 1.95 < Height then Class = Tall

Notice that this problem doesn’t exist if we look at tuples individually using the attribute-value pairs.
However, in that case we wouldn't generate the needed ranges for classifying the actual data. At this point we have classified all tall tuples. The algorithm would then proceed by classifying the short and medium classes. This is left as an exercise.

Algorithm 0.1

Input:
\[ D \] //Training data
\[ C \] //Classes

Output:
\[ R \] //Rules

PRISM Algorithm:
//PRISM algorithm generates rules based on best attribute-value pairs
\[ R = \emptyset; \]
for each \( C_j \in C \) do
repeat
\[ T = D; \] // All instances of class \( C_j \) will be systematically removed from \( T \)
\[ p = \text{true}; \] // Create new rule with empty left hand side
\[ r = \{ \text{If } p \text{ then } C_j \}; \]
repeat
for each attribute \( A \) value \( v \) pair found in \( T \) do
calculate \( \left( \frac{|\{ \text{tuples in } T \text{ with } A=v \land p \land \in C_j \}|}{|\{ \text{tuples in } T \text{ with } A=v \land p \}|} \right) \);
find \( A = v \) that maximizes this value;
\[ p = p \land (A = v); \]
until all tuples in \( T \) that satisfy \( A = v \);
\[ T = \{ \text{tuples in } T \text{ that belong to } C_j \}; \]
\[ D = D - T; \]
\[ R = R \cup r; \]
until there are no tuples in \( D \) which belong to \( C_j \);