Implementing Microsoft Decision Tree and Time Series algorithms:

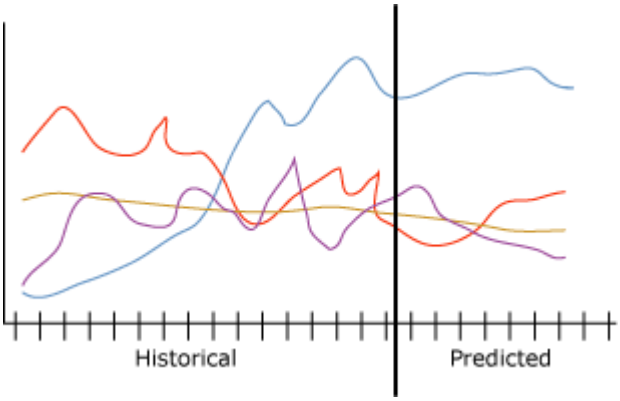
The Microsoft Decision Tree algorithm can be used to predict both discrete and continuous attributes. For discrete attributes, the algorithm makes predictions based on the input column or columns in a dataset. For example, if eight out of ten customers who bought chips were kids, the algorithm decides that age is a factor for the split. Here the predictable column would be a plot of the chip buyers against their age and the mining model would make a decision split for high or low age. When we keep track of this predictable column, the pattern and statistics such as count of the data, we make subsequent queries easier to execute. For continuous variables, the algorithm uses linear regression to determine where a decision tree splits.

The decision tree algorithm could predict discrete data. For continuous data, the decision tree algorithm builds a tree where each node contains a regression formula. At the point where a regression formula shows non-linearity, the decision tree algorithm adds a split. For example, if the scatter plot of the data could be fitted with trends represented by two lines joining at a vertex, the corresponding decision tree generated would have the equations for the two lines as the leaf nodes from a split of all the data.

We implement a decision tree as a recursive function:

```
tree CreateDecisionTree(data, attributes, target_attr, fitness_func)
  best_attribute = feature_select(attributes)
  tree = {best_attribute};
  for val in get_values(data, best)
      subtree = CreateDecisionTree(
                        get_examples(data, best, val),
                        attributes other than best,
                        target_attr,
                        fitness, func)
      tree[best][val] = subtree;
   return tree
```

The Microsoft time series algorithm is used to forecast the events given the historical information. From the MSDN:



We will specifically look at ARTXP and ARIMA algorithms mentioned here

These are for short term and long term predictions respectively and they use decision trees. In mixed mode, both algorithms are used and there is a parameter to bias it to one of the algorithms with a 0 and to the other with a 1 and intermediary in between.

The time series algorithm uses the random variables of a Markov chain to predict the immediate next given the history. It's called an auto regressive model because the probability for a value at a time has mean auto regressively dependent on the last p values in the series.

An autoregressive tree model has AR model at every leaf. This model represents the time periods in a series.

When we fit a linear regression to the scatter plot of the random variable, we can use it to describe the one step transition. However this is often too general. Instead if we split the data into regions and fit the regression, we get better approximation. This is what the splitting does. To learn the decision tree for the time series, we apply a split-leaf operator. This operator takes a predictor variable and a value. The resulting left and right nodes are for the two resulting regions.

To apply the split for a leaf, we use the seven values of each predictor variable that come from the boundaries of eight equi-probable contiguous regions of a normal distribution estimated from the restricted data set at the leaf for the predictor variable. We choose the split that result in the largest increase in model score. We stop splitting the tree when no split yields a higher score.

If we do not know the set of potential split variables, we iteratively apply the above method to get an ART for each and choose the one with the highest Bayesian score.

We will now see how to use this to predict the next step data. Note that the leaves of the decision tree for a piece wise trend predictor. To forecast, we can do a one step forecasting or a multi step forecasting. The former is more accurate than the latter. We are only interested in the former. To predict the distribution of the random variable for the next step, we only need one of the leaf nodes. Each leaf for the tree has a conditional time distribution for this variable because the path from the root to the leaf we are interested in has a conjunction of all the formulas. Remember that our generic auto regressive model of length p is a linear regression modeled using a normal distribution with model parameters as the mean, variance and the coefficients for each of the p observations. The autoregressive model tree is merely a piecewise linear AR model. This structure is chosen based on the highest posterior conditional probability $p(s|d)$ for the structure s given the data d. And this probability can be expanded as the combination of that for the structure and the marginal likelihood where the likelihood of the data is averaged over the uncertainty in the model parameters. To predict the data, we use this normal distribution with the model parameter of the one that is most likely

The likelihood could be measured with t-distribution but we use the normal distribution and for the leaf we are interested in, we find the transition probability to the next one-step and we find this given it's a stationary random Markov chain.

By that I mean we use the equation for the leaf that gives us the one step transition probability from the distribution we have. To find the equation, we find the coefficients, mean and variance from the last p observations using a normal distribution estimated from the dataset. We transform the dataset into the length p transformation of the time series data set where each of the transformation x = (x1, x2, ..., xp+1) and x is calculated for each i in 1 to T-p. The element within the bracket which is denoted by xj is the original event (i+j-1). For example, if we have events y = (1, 3, 2, 4) then the length-2 transformation is x1 = (1,3), x2 = (3,2) and x3 = (2,4) and the length-3 transformation is (1,3,2) and (3,2,4). This is called windowing and with the transformations that we get with this approach, we can now apply the decision tree technique mentioned earlier. We create a single leaf with the normal distribution from this transformation dataset which fits a line to the transformations with a target as the p+1. To fit the linear regression for a restricted data set, we determine the values of the random variable from the length p transformations of the time series data set.

For a given time series data set, a corresponding nine data sets for length p transformations is created. The p varies from zero to eight for the nine data sets. Each of these transformed datasets is centered and standardized before modeling; that is for each variable we subtract the mean value and divide by the standard deviation. Then we divided the data set into a training set used as input to the learning method and a holdout set to evaluate the model. The holdout set contains the cases corresponding to the last five observations in the sequence.

Then we decide to split it at one of the seven values corresponding to the eight equiprobable continuous regions (each of standard deviation length) of the normal distribution for the predictor variable. We continue to split until it does not improve the model score. The model score is a Bayesian score with the first term as the probability of the structure and the second term is the marginal likelihood. The second term balances the fit of the model to the complexity of the data. The second term can be approximated as a fit penalized by complexity to ease the calculation of the score for the model.

Courtesy:

Building decision trees in Python : OReilly Media.

Microsoft Decision Tree Algorithm Technical Reference

Microsoft Time Series Algorithm Technical Reference