Weka Analyzer Visualizations Tutorial

# Summary

This tutorial will give you a hands on overview of using three custom built Weka Visualizations in Analyzer. You'll learn about three common data mining algorithms including classification, clustering and attribute selection. We'll also talk about how to use the models you build to score new customers.

This tutorial was based on the Pentaho World 2015 presentation "Extending the Pentaho Platform with a Decision Tree Visualization" presented by Benny Chow, Pedro Vale and Steve Szabo.

# Business Background

Data mining algorithms can be difficult to use for business users and so the goal of the Weka Analyzer visualizations is to leverage the easy to use interface of Analyzer, coupled with the advanced algorithms in Weka to enable a business user to quickly model, explore and run data mining models on their own data.

In this tutorial, we will work with a marketing manager Ben as he tries to design the next marketing campaign at a big bank. He is going to analyze recent telemarketing campaign response data to see what types of customers actually ended up purchasing a product. He will combine his understanding of the business with three different data mining algorithms to build a new model to predict what types of new customers are likely to purchase. He will then use his new model to score new customers and target those likely to purchase in the next marketing campaign.

His [data set](https://github.com/bennychow/weka-analyzer-viz/raw/master/tutorial/telemarketing.csv) includes the following important attributes about his customers:

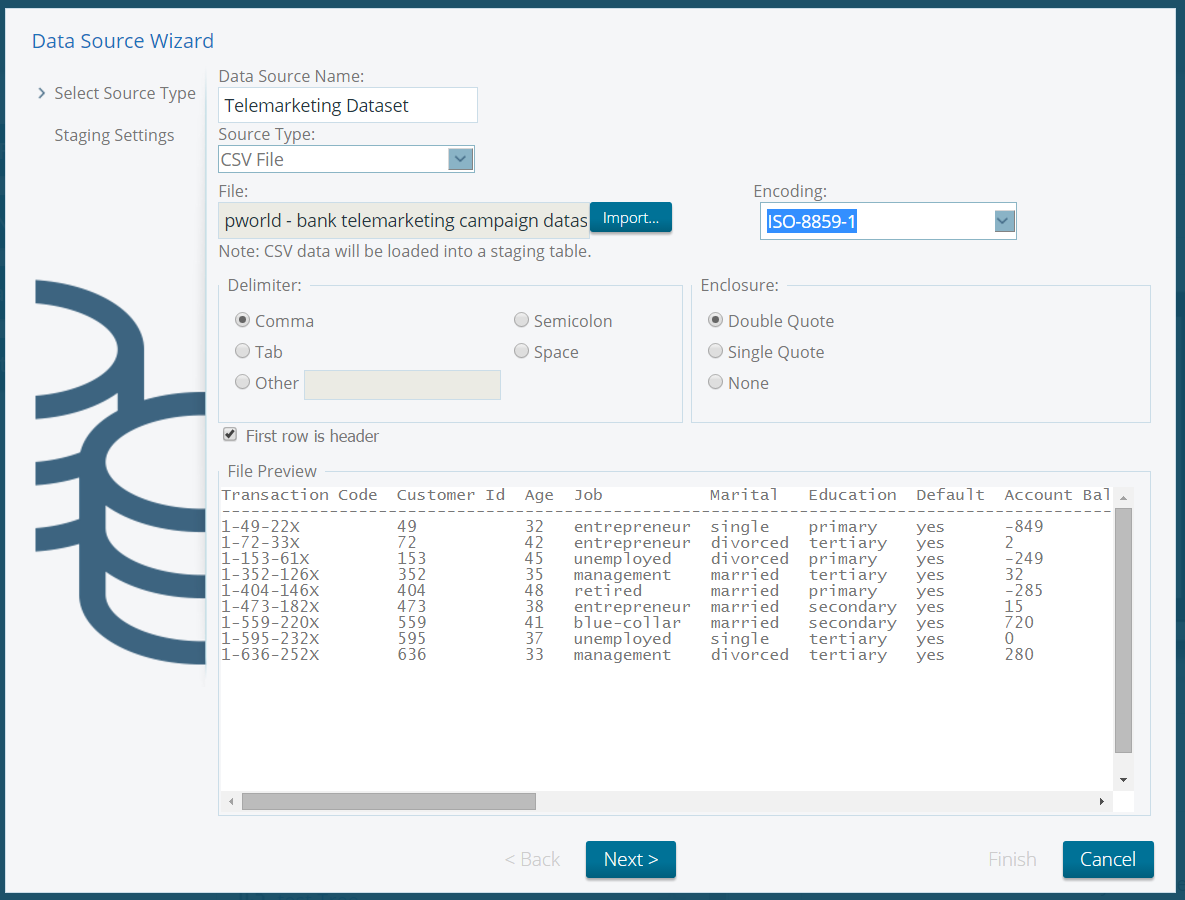
* Customer Age
* Job
* Marital Status
* Education Level
* Account Balance - Negative if they owe the bank money
* Housing - Whether they own a house
* Loan - Whether they have a loan with the bank
* Contact - How the user was contacted by the telemarketer
* Day/Month/Date - Date time of contact
* Call Duration - How long the call lasted
* Time Between Calls - Time between the current call and last call
* # of Calls
* Campaign Success - Outcome of the call. Interest level of customer in product
* Purchased Product - Whether the customer ultimately purchased the product

# Importing Data and Modeling

There are many ways to get data into Pentaho and the quick and dirty way is to upload a CSV file using the data source wizard. This wizard will inspect the CSV file and create a single DB table with column types matching the CSV headers.

You can download the telemarketing dataset [here](https://github.com/bennychow/weka-analyzer-viz/raw/master/tutorial/telemarketing.csv).

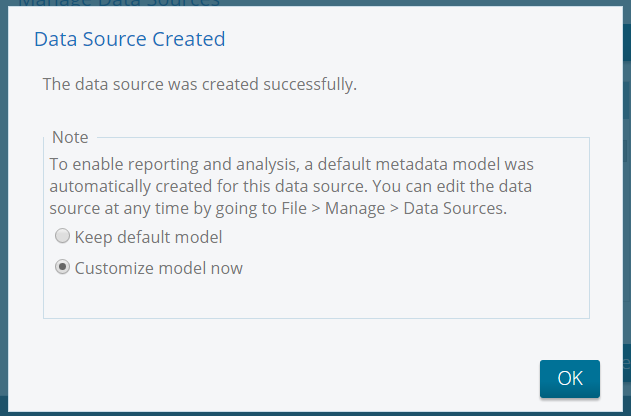
Fortunately, Ben has his dataset on his desktop and he loads it into Pentaho as shown below:



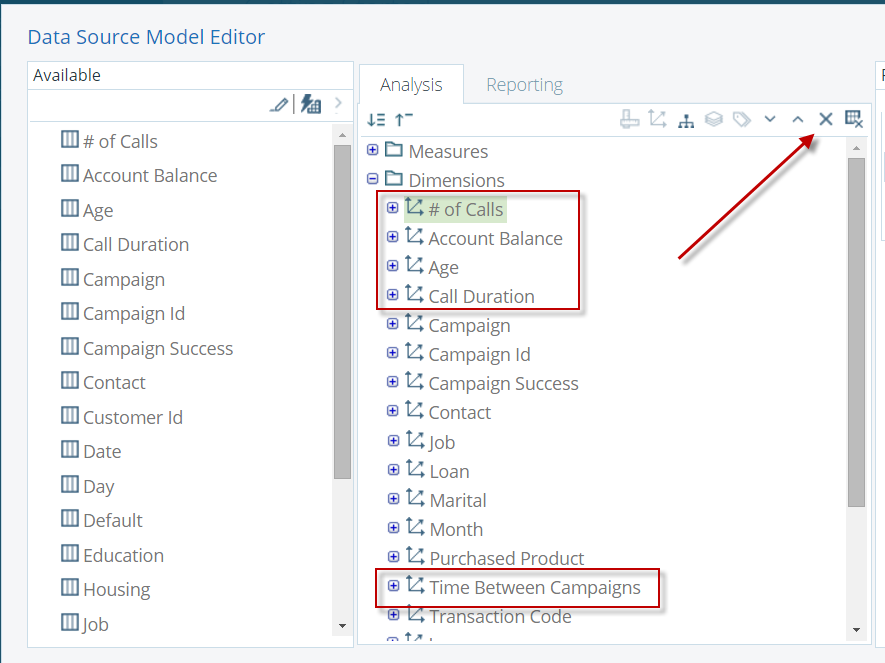
Take all defaults and click Next and then Finish. We'll explore this dataset later in Analyzer.

Datasource Wizard will expose every numeric column as both a single level hierarchy and a measure. For data mining purposes, it's better to leave numeric data as numeric and not turn them into strings. This allows the algorithms to find arbitrary split points such as Income between 21,588 and 53,328. So, let's clean this up by removing any numeric columns that have been added as levels.

Select Customize Model as shown below.



Select the following Dimensions and delete them from the model:



Click OK and now open your new data source in Analyzer...

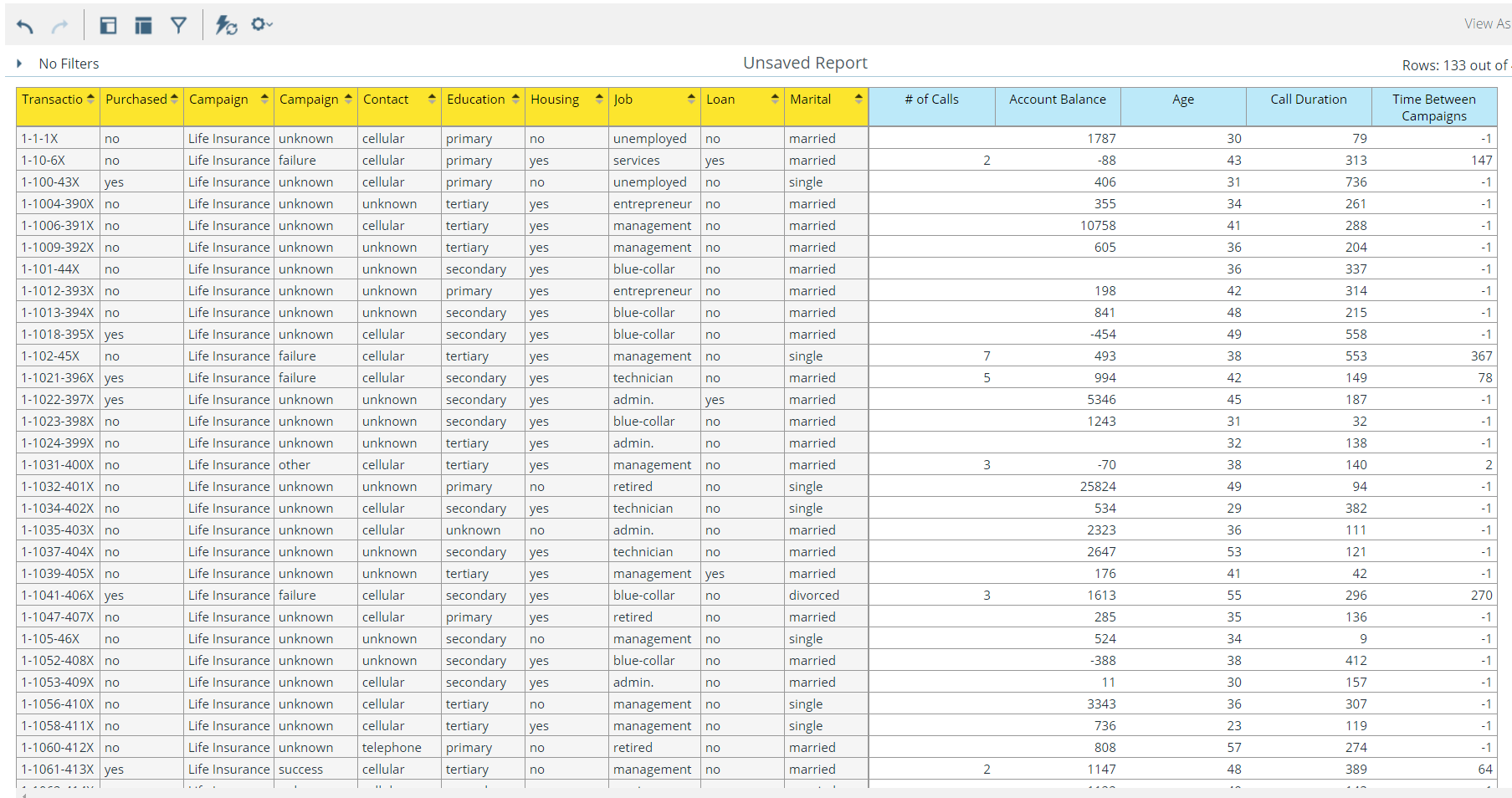
# Exploring the Telemarketing Dataset

The first thing Ben will need to do is to verify his dataset. He does this by dragging out all the fields which he thinks could be predictive towards whether a customer purchases or not. The fields he picks are based on his understanding of what makes a customer purchase. However, later on, we'll see that there are algorithms such as "Attribute Selection" that can automatically suggest the fields that could predict customer purchase.

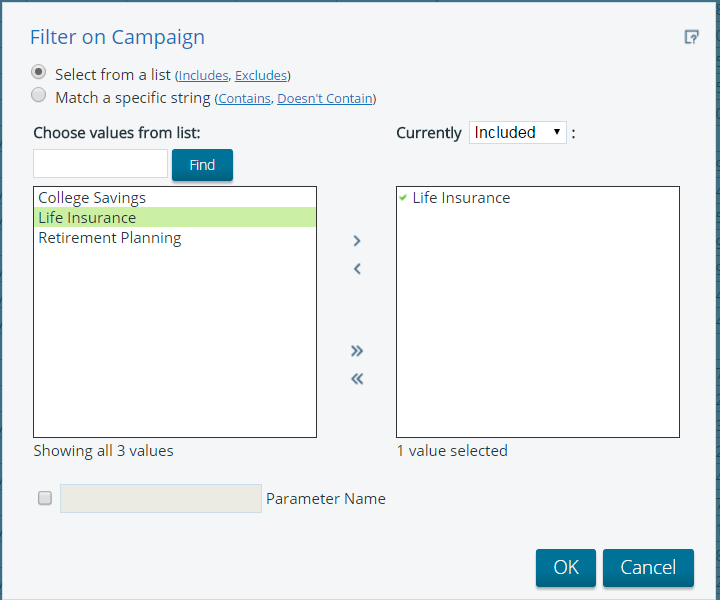
So, to start, Ben drags out the following fields:

* Transaction Code - This uniquely identifies each customer response and outcome
* Purchased Product - Whether the customer purchased or not. Yes/No.
* Contact
* Education
* Housing
* Job
* Loan
* Marital
* # of Calls
* Account Balance
* Age
* Call Duration
* Time Between Campaigns

Ben can see his fields and their value in Analyzer's pivot table view:



Yep... the data looks right. However, Ben remembers he needs to filter on just his campaigns so he creates a new filter on Campaign = Life Insurance which is easy to do in Analyzer:

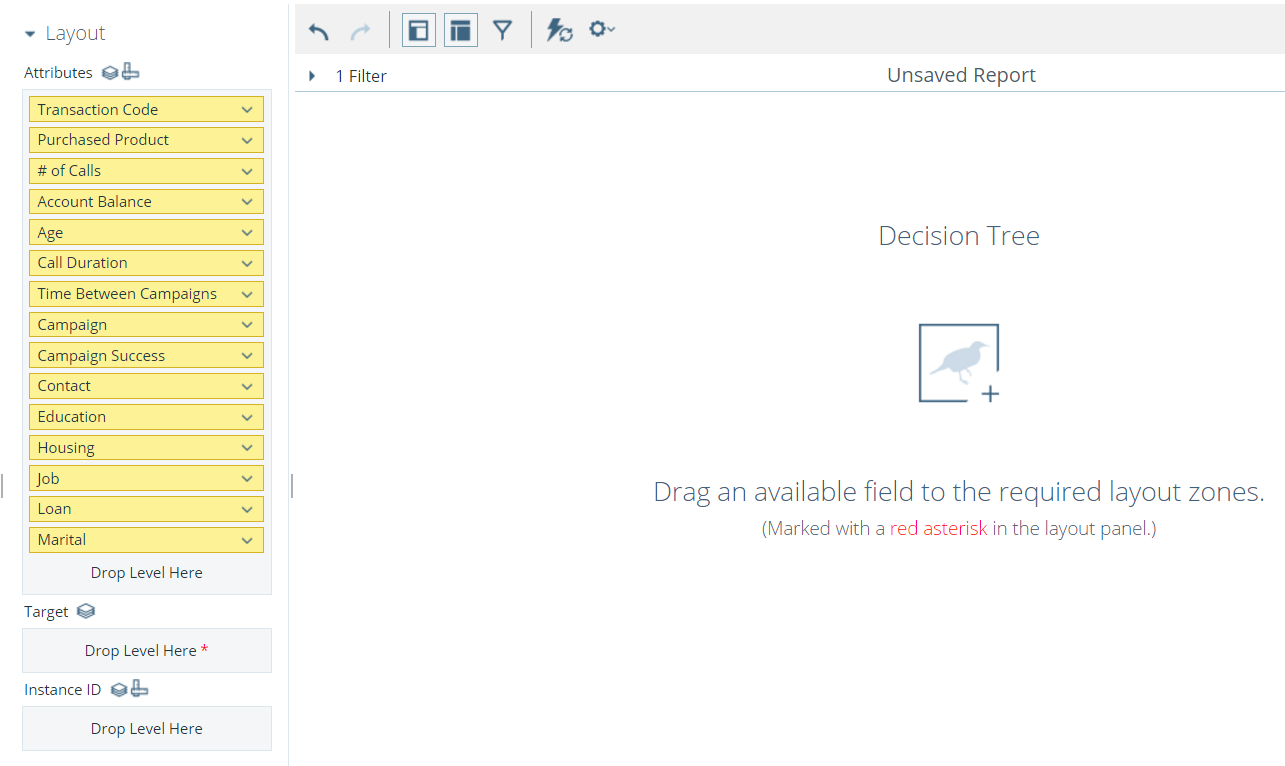


So, next, let's try out the first algorithm: Decision Trees.

# Classification: Decision Tree

Decision Trees are a type of classification model that can predict categorical class labels (ex. "Purchased Product" class with labels "Yes" and "No") based on numeric or categorical inputs. They result in a white box model which is easy to understand and interpret because the branch rules make business sense.

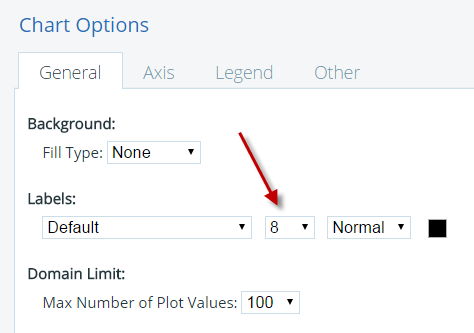
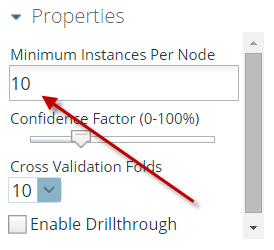
Now Ben opens the View As chart dropdown menu and selects Decision Tree as shown below:



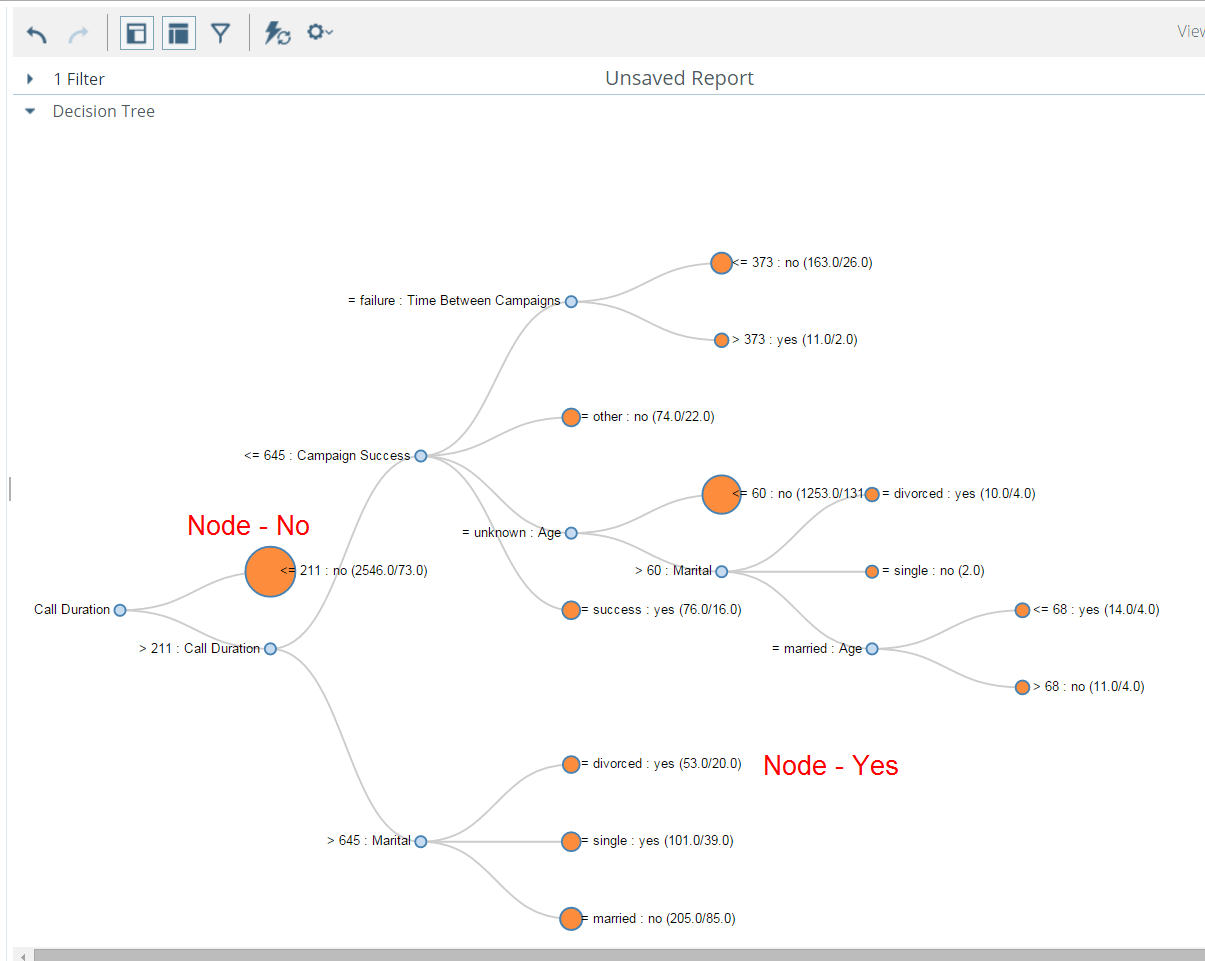
The layout panel for all of the Weka visualizations is the same and contains the following layout zones:

* **Attributes** - These are the independent variables that will go into the model and are used to predict the target.
* **Target** - This is the dependent variable which the model is predicting or testing against.
* **Instance ID** - This attribute is used to uniquely identify instances but will not be passed into the model. For example, Transaction Code is needed to uniquely identify each customer outcome but we don't need this code in the model. If we didn't include Transaction Code in Instance ID, then the instances would be aggregated up to the Attributes and Target fields.

Because Ben is trying to predict which customers purchased a product, he drags Purchased Product to Target and drags Transaction Code to Instance ID. In order to make the results easier to understand, Ben sets the Minimum Instances per Node to 10 and Chart Options Label size to 8.

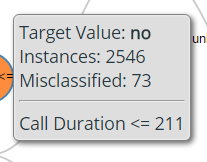


This results in the below decision tree:



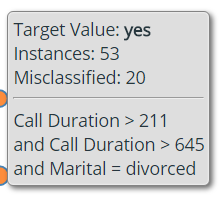
Let's take a look at what's happening here...

The "Node - No" is predicting that the customers in that node will NOT purchase a product:



On mouse over, Ben can see that if the telemarketing call duration was less than 211 seconds, then the predicted outcome was no product purchased. In his dataset, 2546 of the 4519 records had call duration <= 211 seconds. Of those 2546, only 73 were misclassified as not purchasing a product. Those 73 actually did end up purchasing and thus are false negative. From a marketing point of view, it is obvious that if you can't keep the customer on the phone, then you won't be able to generate interest in the product.

Now, Ben takes a look at the "Node - Yes".



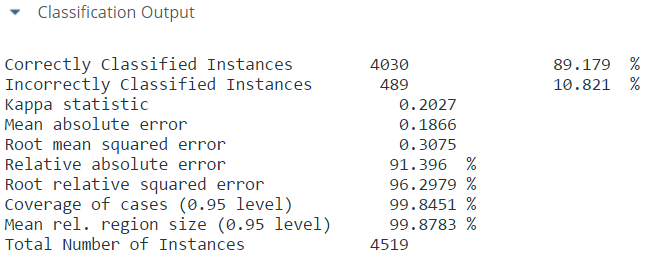
This node is predicting that these customers will purchase a product. It says that if Call Duration > 211 AND Call Duration > 645 seconds and Marital Status = Divorced than that customer will likely purchase. While 20 false postives out of 53 total may seem inaccurate, if a typical campaign response rate is less than 1%, this node's rules coupled with other nodes that predict "Yes" may still improve the campaign response rate dramatically targeting the best customers.

At this point, Ben can utilize his domain knowledge and try out different model inputs. For example, he may think that "Call Duration" is too correlated with purchasing a product because a customer willing to talk to you inherently means that he is interested. In that case, Ben can drop Call Duration from the model and predict on other model inputs.

## Other Model Outputs

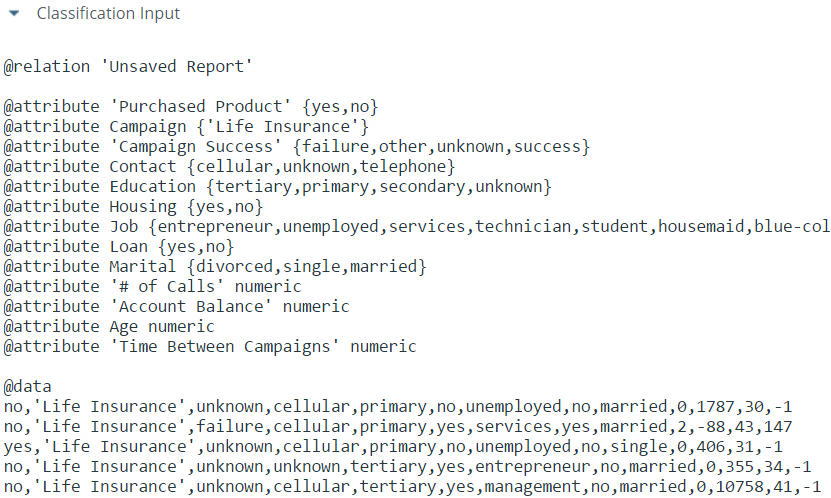
The decision tree visualization includes dropdown panels for various model outputs described below:

### Classification Output



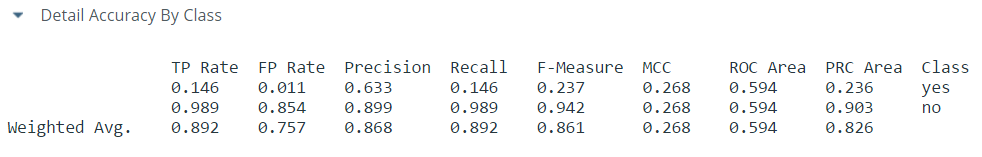
This measures the predictive performance of the classification model by running the model N fold times and averaging the results. Each fold consists of a 90-10 ratio of training data to test data. In Ben's model, 89.179% of the test data was correctly classified averaged across 10 different runs.

### Classification Input

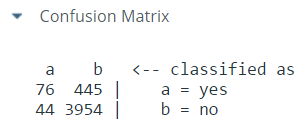


This is the raw instance data that is fed into the Weka data mining algorithm.

### Detailed Accuracy by Class

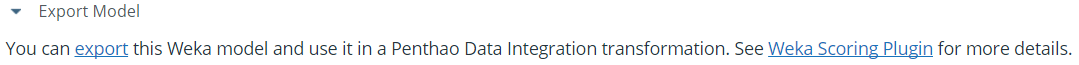


### Confusion Matrix



This matrix tells us that there were a total of 76+445=521 real Yes outcomes of which 76 were correctly predicted/classified as Yes (true positive) and 445 incorrectly classified as No (false negative).

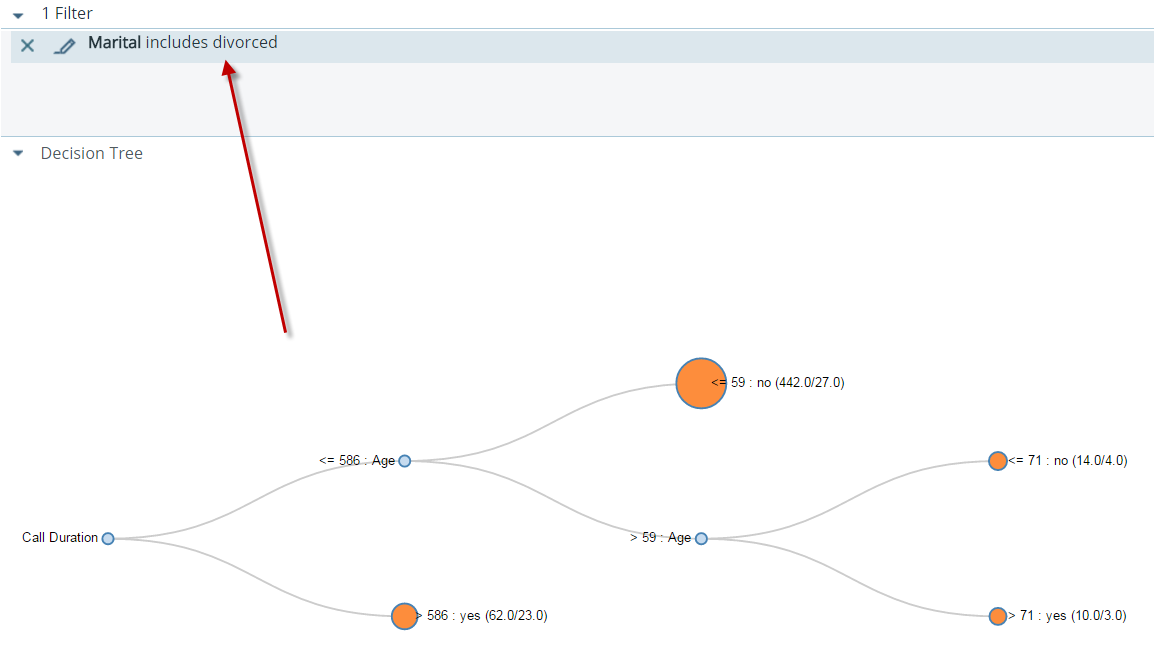
### Export Model



This link allows Ben to export the model and use it to score new customers in a PDI transformation. More on this later...

## Filter on Node's Branch Value

Visualizations can implemented custom user interaction. In this case, the Decision Tree visualization supports double clicking on a node to filter on that node's branch value. For example, Ben can double click on the "Node - Yes" and filter where Marital Status = Divorced:



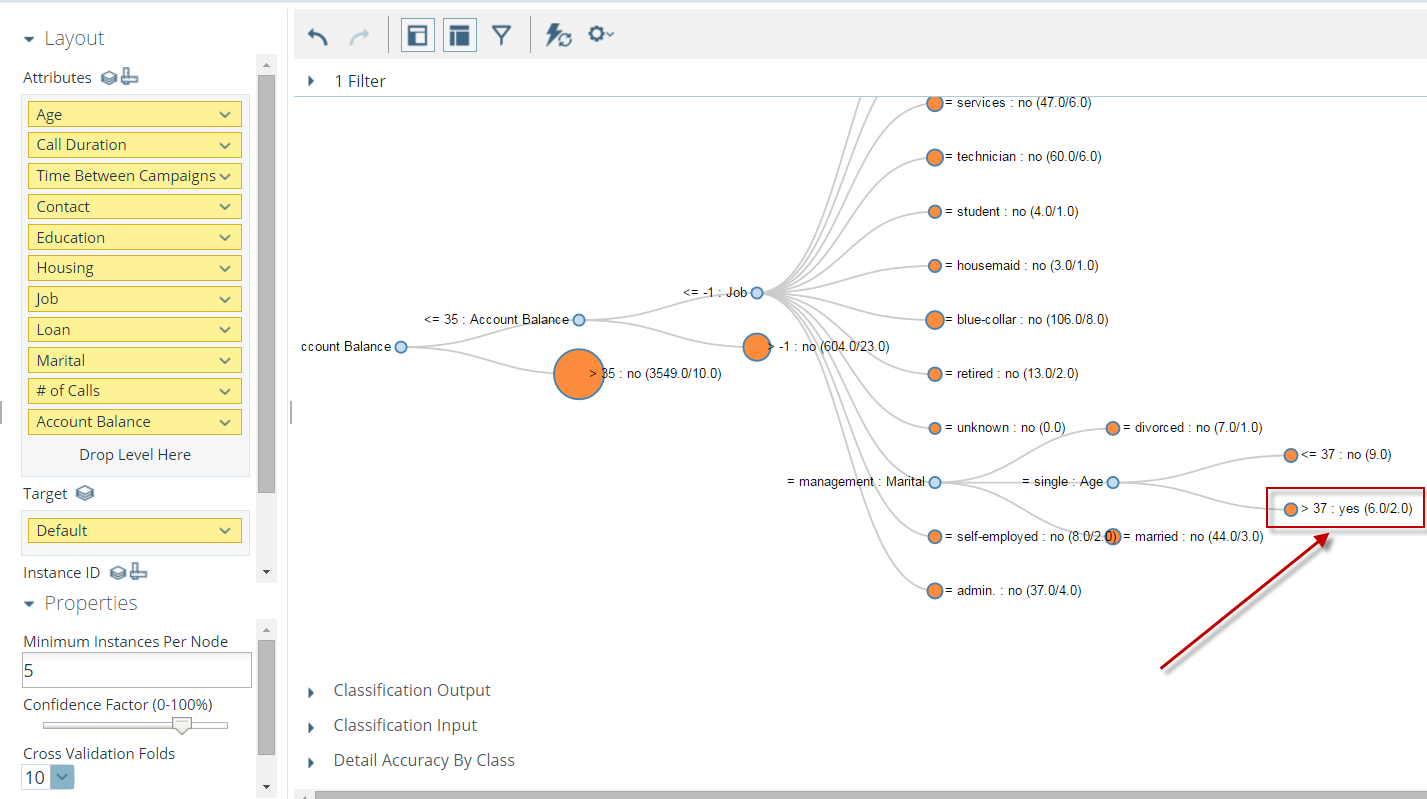
## Enabling Drill Through

One cool thing that Ben can also do is that he can see all the instance data that went into a node. For example, with the node marked as "Node - Yes", he can see all the other fields such as "Age", "Income", "Education", etc even though the decision tree did not use those fields to arrive at that node. To do this, check the "Enable Drillthrough" box and then double click on the node:



## Predicting on Customer Default

With such an easy to use interface, Ben can easily build other decision tree models to predict on other outcomes. Here is another decision tree which predicts which types of customers are more likely to default on a payment:



For the node boxed in red, we can see that:

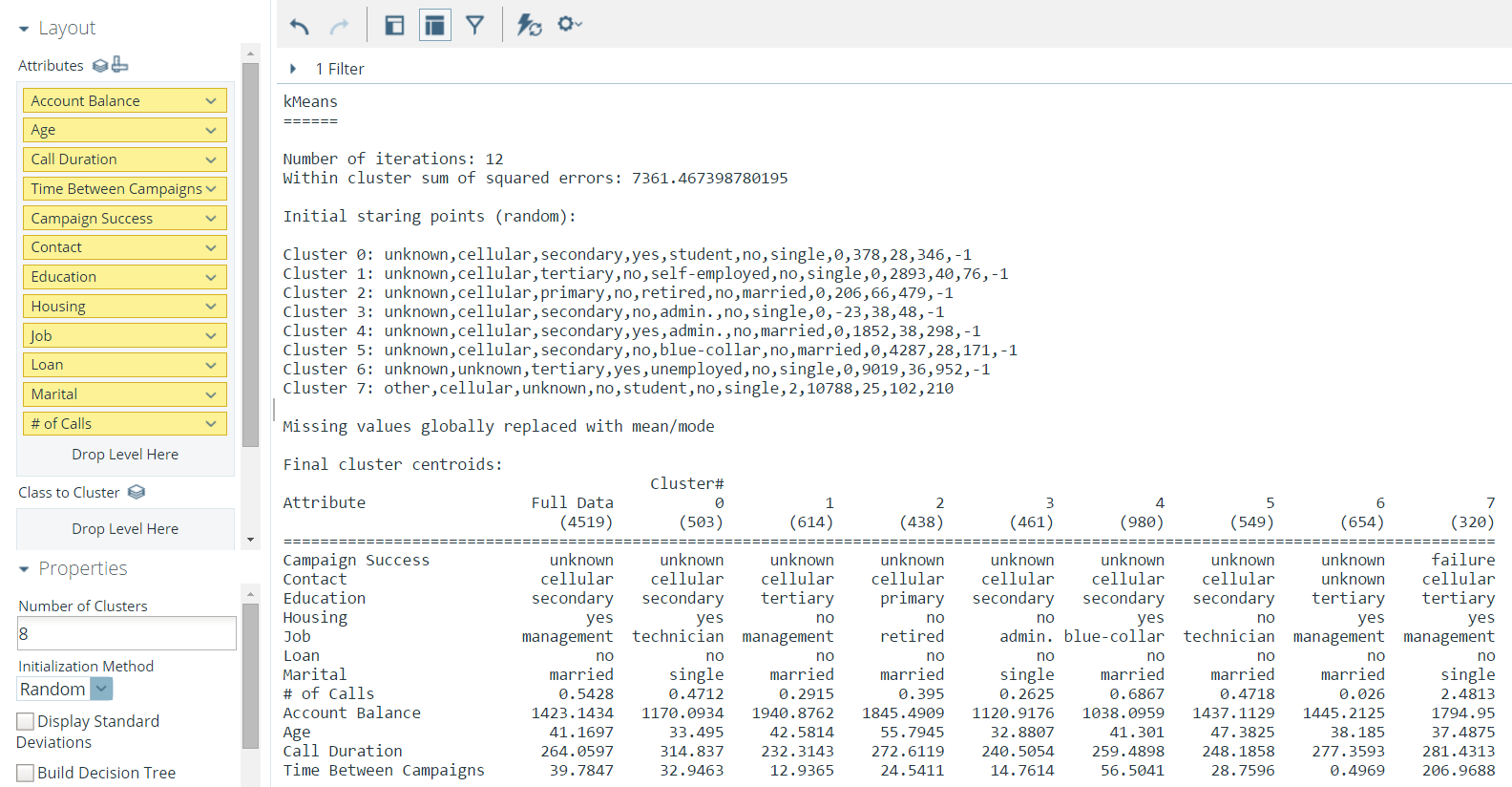
Customers with the following attributes are more likely to default on their payments:

* Account Balance <= 35
* Account Balance <= -1
* Job = Management
* Marital = Single
* Age > 37

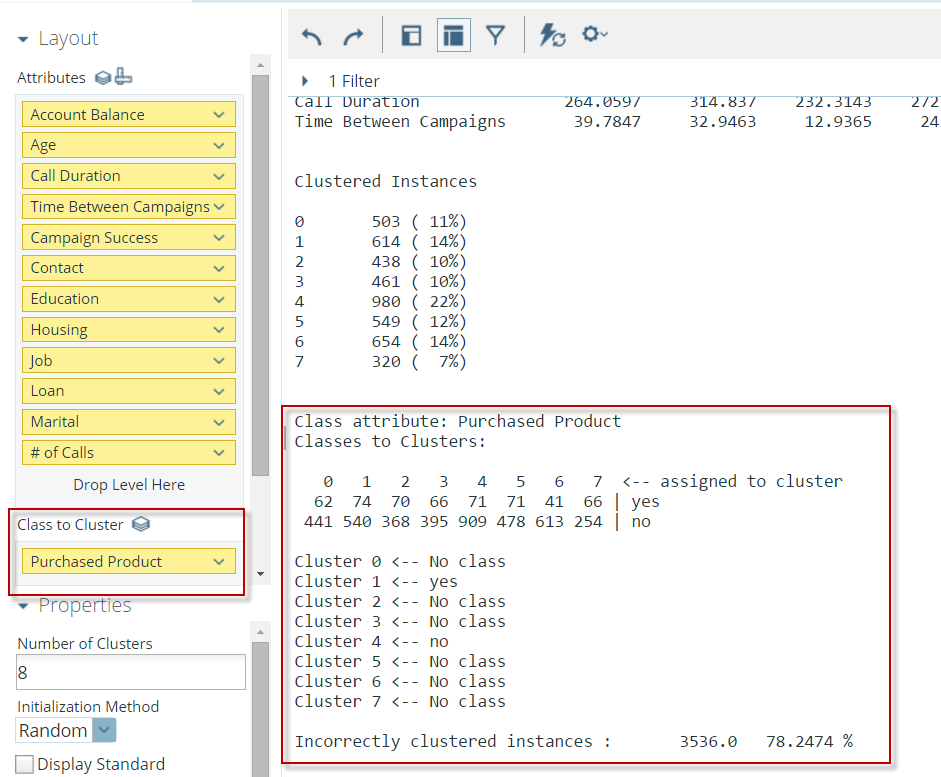
Obviously, you need a negative balance to default on a payment, the other attributes, well, are up to Ben's interpretation...

# Clustering

The cluster visualization will find groups of instances/records that are more similar to each other than to those in other groups. Groups can optionally be tested to see how well they predict a specific outcome. Let's take an example. Suppose Ben builds 8 clusters as shown below:



Now he wants to see if any cluster contains a higher percentage of product purchasers. By dragging the Product Purchase field to Class to Cluster, he is now measuring each cluster's ability to predict Product Purchase:



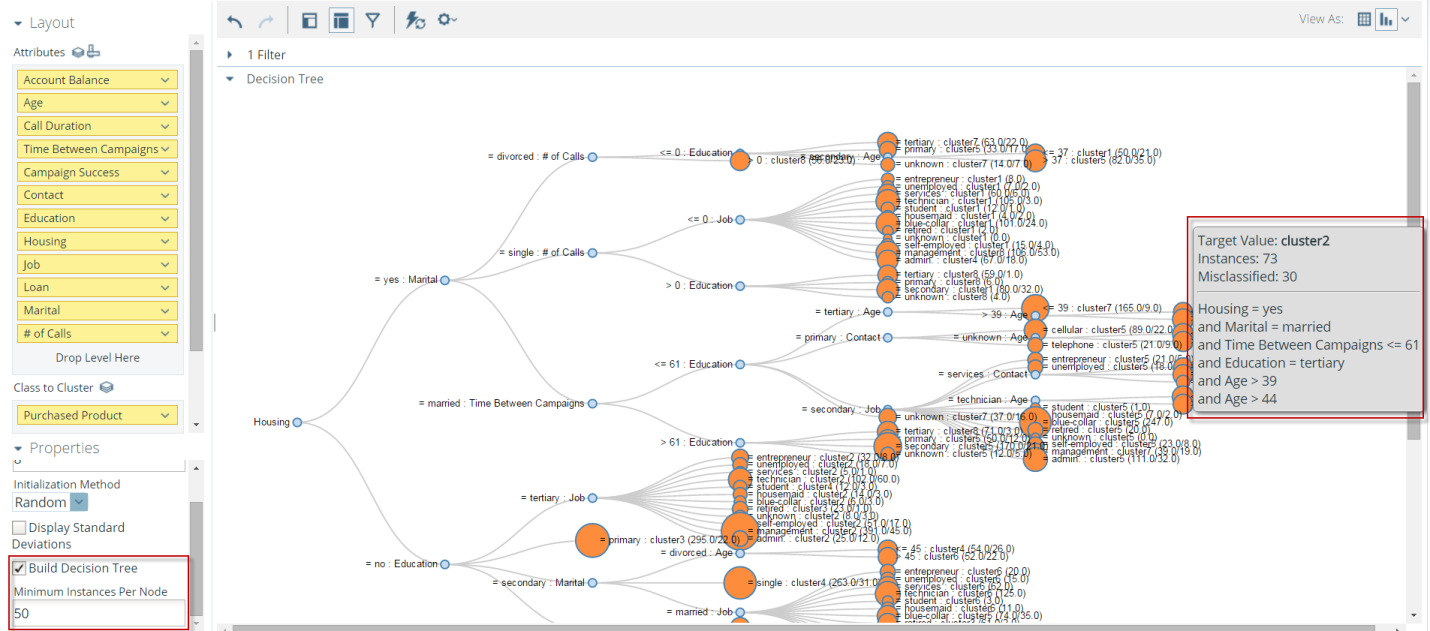
Ben can see that Cluster 2 contains the most product purchasers. Then he can read the description of Cluster 2 members from above:

* Campaign Success = Unknown
* Contact = Cellular
* Education = Primary
* Housing = No
* Job = Retired
* Loan = No
* Marital Status = Married
* Average # of Calls = 0.395
* Average Account Balance = 1845.49
* Average Age = 55
* Average Call Duration = 272
* Average Time Between Campaigns = 24.54

The above description of cluster 2 is based on the cluster 2's centroid or the middle of the cluster. A centroid is a vector containing one number for each attribute, where each number is the mean of a attribute for the instances in that cluster.

Using the cluster centroid is one way to describe the members of a cluster however this drawback is that it uses averages and considers all input attributes. It is also possible to use a decision tree to predict the cluster and thus allow the decision tree to pick the best attributes to split on.

In the properties panel, Ben checks "Build Decision Tree" and sets "Minimum Instance per Node" to 50 and builds the below decision tree on the cluster outputs. This effectively runs two models on the input dataset which assigns cluster numbers to each instance and then builds a decision tree on top of the data that explains the cluster number.



In the example node shown above, this node contains 73 instances where 43 have been correctly classified as cluster2 and 30 incorrectly classified. If Ben believes that cluster2 very likely predicts that a customer will purchase he can also describe product purchasers using the decision tree branches that lead to this node:

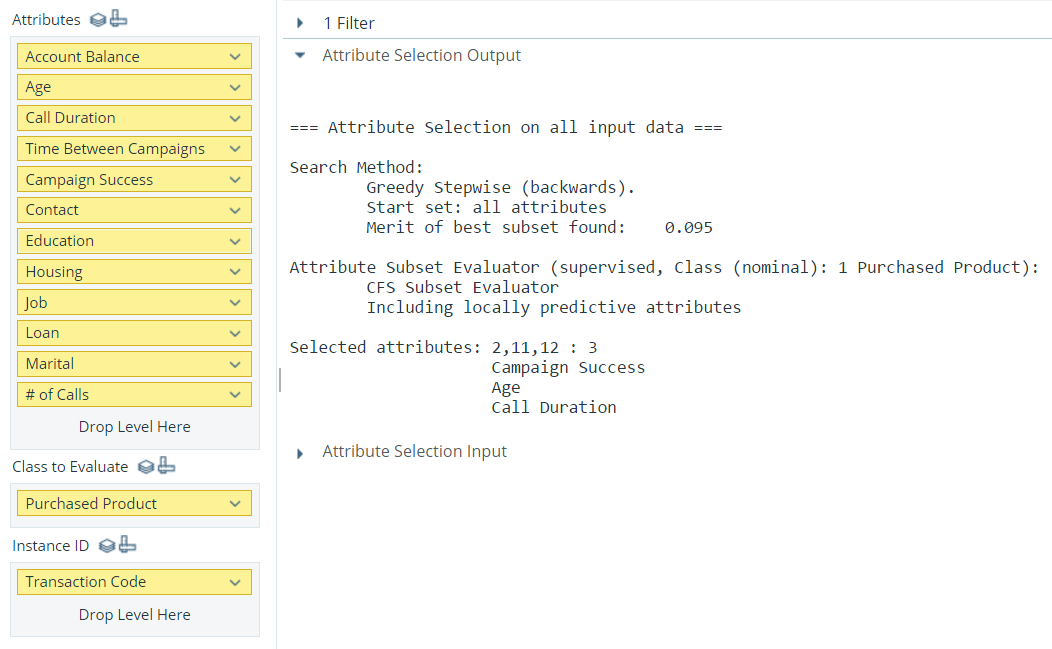
* Housing = Yes
* Maritial Status = Yes
* Time Between Campaign <= 61
* Education = Tertiary
* Age > 39 and Age > 44

He can also study all the other leaf nodes whose target value is cluster2 to see if those decision tree rules make sense for selecting product purchasers.

# Attribute Selection

With decision trees, the algorithm is supposed to select the best attributes to split on and continues this selection until it meets some criteria to generate a leaf node. However, at a high level, it may be useful to know what attributes or attribute combinations are most correlated with the predicted outcome. Using Attribute Selection, Ben can find out which input attributes are most predictive for Product Purchasers.

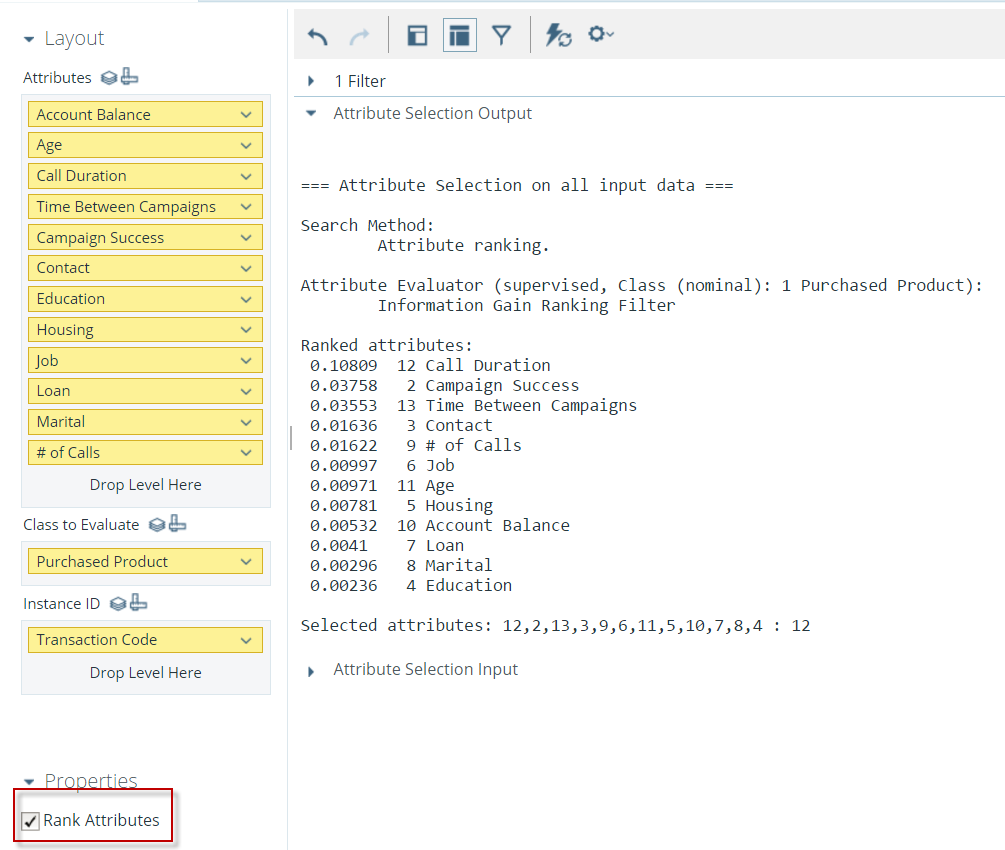
Ben selects the "Attribute Selection" visualization as shown below:



Here we see that the combination of these three attributes is most correlated with predicting product purchasers:

* Campaign Success
* Age
* Call Duration

This attribute selection method will consider all possible combinations of the input attributes. However, if Ben wanted a ranking of the best to worst attributes for predicting, he can check the "Rank Attributes" checkbox:



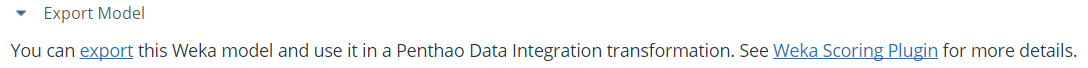
This attribute selection method considers each input attribute independently of the others.

# Scoring in PDI

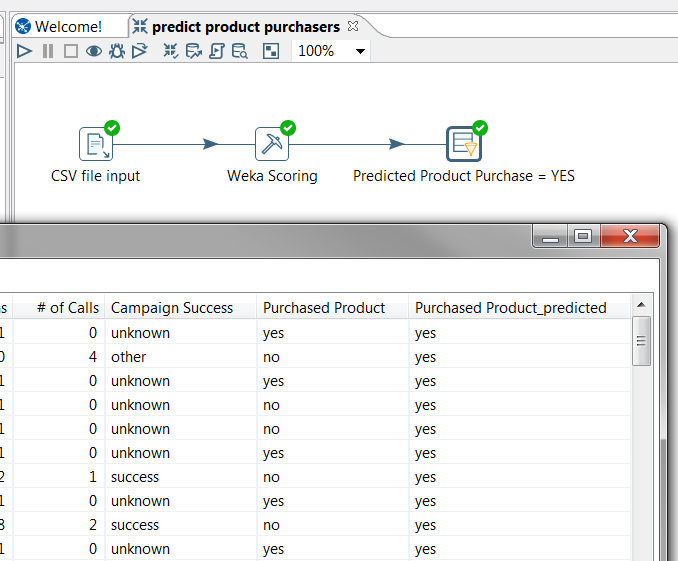
Ben is happy with his decision tree classification model and wants to use that model to score new customers to determine those most likely to purchase and target them into a follow up campaign. By exporting the model from the visualization and using them in the "Weka Scoring" step, he can build an ETL transformation that reads in new customers containing all the fields that match the model input attributes and then generate a new field that contains the "Purchase Product" predicted outcome with "yes"/"no" values. Ben can then filter on the "yes" values and target them in a new marketing campaign.

This Weka Scoring plugin is described here: [Weka Scoring Plugin](http://wiki.pentaho.com/display/DATAMINING/Using+the+Weka+Scoring+Plugin).

Using the decision tree visualization as an example, Ben can export the model as shown below:



He can then build the following transformation and use the above model:



Notice that "Purchased Product\_predicted" was added by the Weka Scoring step and then we use a Filter Rows steps to select only the "Yes" rows. You can download the [sample transformation here](https://raw.githubusercontent.com/bennychow/weka-analyzer-viz/master/tutorial/predict%20product%20purchasers.ktr) or build it yourself. If you use the pre-built sample, make sure to fix up the CSV file path. The model has already been saved into the Weka Scoring step but you can just as easily re-load it from [here](https://github.com/bennychow/weka-analyzer-viz/raw/master/tutorial/decision-tree-model).

# Building a Custom Visualization

If you are a developer and interested in learning how this plug-in was built you can watch the 2015 Pentaho World session: <https://www.pentahoworld.com/ondemand/>

You can login with: access@pentaho.com

And look for the presentation:

