

DTREG COM Library

Copyright © 2004-2008, Phillip H. Sherrod
All Rights Reserved

Phillip H. Sherrod
6430 Annandale Cove
Brentwood, TN 37027-6313 USA
phil.sherrod@sandh.com

www.dtreg.com

Contents

Contents	1
Introduction.....	2
Registering the DTREG COM Library.....	2
Data Types Used With the DTREG COM Methods.....	2
Calling DTREG from Visual Basic	3
Referencing DTREG from Visual Basic.....	3
Declaring the DTREG COM object in a Visual Basic program.....	4
Example Visual Basic program	5
Calling DTREG from ASP Scripts.....	6
Methods and Properties in the DTREG COM Library.....	8
LastStatus -- Status code for the last operation.....	8
OpenProjectFile() -- Open a DTREG project file.....	8
ModelType -- Get type of model.....	9
NumberOfVariables -- Get variable count from a project.....	9
VariableIndex -- Get index number for a variable.....	10
VariableName -- Get the name of a variable from its index number.....	10
VariableClass -- Get the class of a variable.....	10
VariableType -- Get the type of a variable.....	11
VariableImportance -- Get the importance of a variable.....	11
NumberOfCategories -- Get number of categories of a variable.....	11
CategoryIndex -- Get index number for a category.....	12
CategoryLabel -- Get the label of a category from its index number.....	12
SetVariableValue -- Set the value for a variable.....	13
MissingValue -- Numeric value to use for missing values.....	13
SetVariableCategory -- Set the category for a variable.....	13
PredictedTargetValue -- Compute predicted target value.....	14
PredictedTargetCategory -- Compute predicted target category.....	15
TargetCategoryProbability -- Compute probability of target category.....	15
Index	17

Introduction

The DTREG COM (Component Object Model) library makes it easy for production applications to call DTREG as an “engine” to compute the predicted value for data records using a predictive model. You must use the GUI version of DTREG to construct a model before you can use it with the DTREG COM library to predict values.

Any type of model (Single Tree, TreeBoost, Decision Tree Forest, SVM, LDA or Logistic Regression) can be used with the DTREG COM library to generate predicted values. All of the advanced scoring features such as the use of surrogate splitters to handle missing predictor values are used in the DTREG COM library.

Because of the standardization of the COM interface, it is easy to call the DTREG COM library from programs written in Visual Basic, Visual C++, VBA, Excel, Access, ASP and other languages. The DTREG COM library is designed to run as an in-process DLL for speed of execution.

Registering the DTREG COM Library

Before you can use the DTREG COM library, you must register it so that Windows is aware that it exists and knows where it is located. To do this, get to the command prompt in Windows and then change directory to the directory where DTREGcom.dll resides. Then issue the following command:

```
REGSVR32 DTREGcom.dll
```

A window will be displayed confirming that DTREGcom.dll has been registered.

Data Types Used With the DTREG COM Methods

The descriptions of the DTREG methods show the recommended data types for the input and output arguments. For example, the `SetVariableValue` method has two arguments:

```
SetVariableValue(inVariableIndex As Long, inValue As Double) As Long
```

The recommended type for the *inVariableIndex* argument is Long and the recommended type for the *inValue* argument is Double. The value returned by `SetVariableValue` is of type Long.

However, DTREG accepts all of the incoming arguments as VARIANT types and converts them to the appropriate types before using them. So all of the following calls to `SetVariableValue` will work:

```
SetVariableValue(1, 4.0)
SetVariableValue(1.0, 4)
SetVariableValue("1", "4")
```

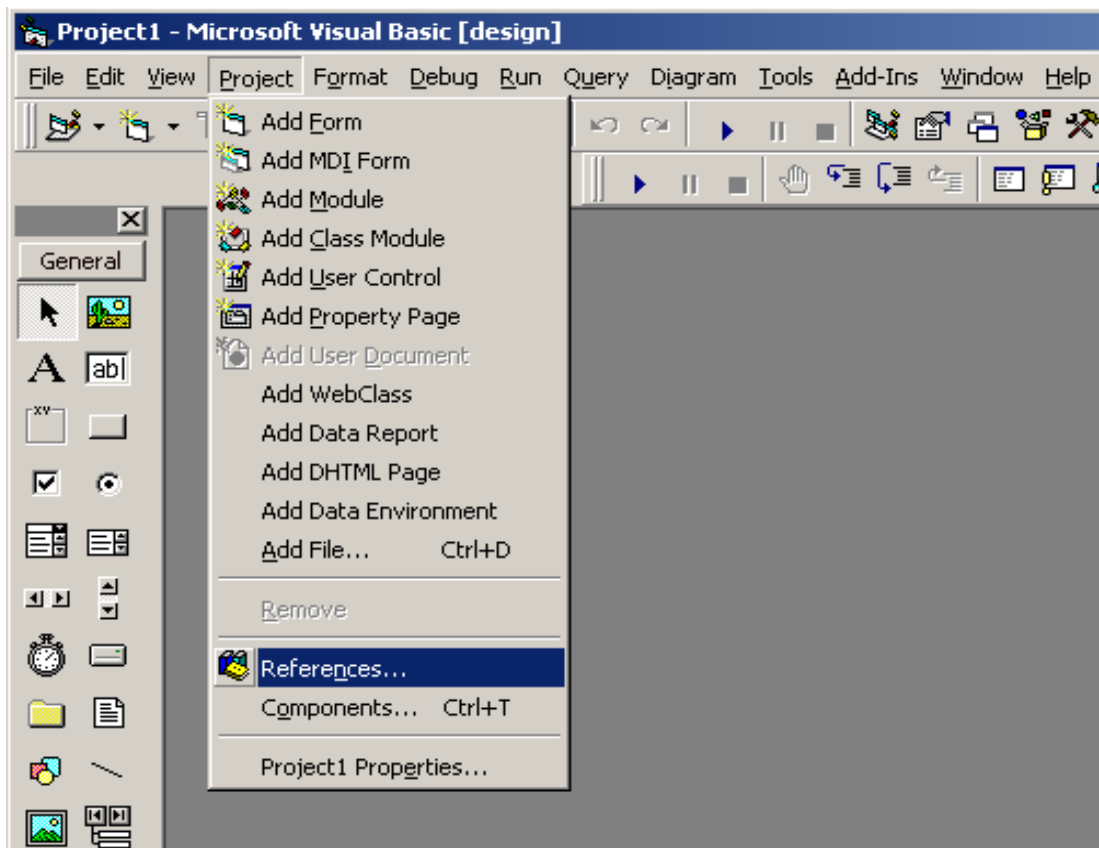
This dynamic data type conversion makes it easy to call DTREG methods from languages such as ASP which do not have data type declarations.

Calling DTREG from Visual Basic

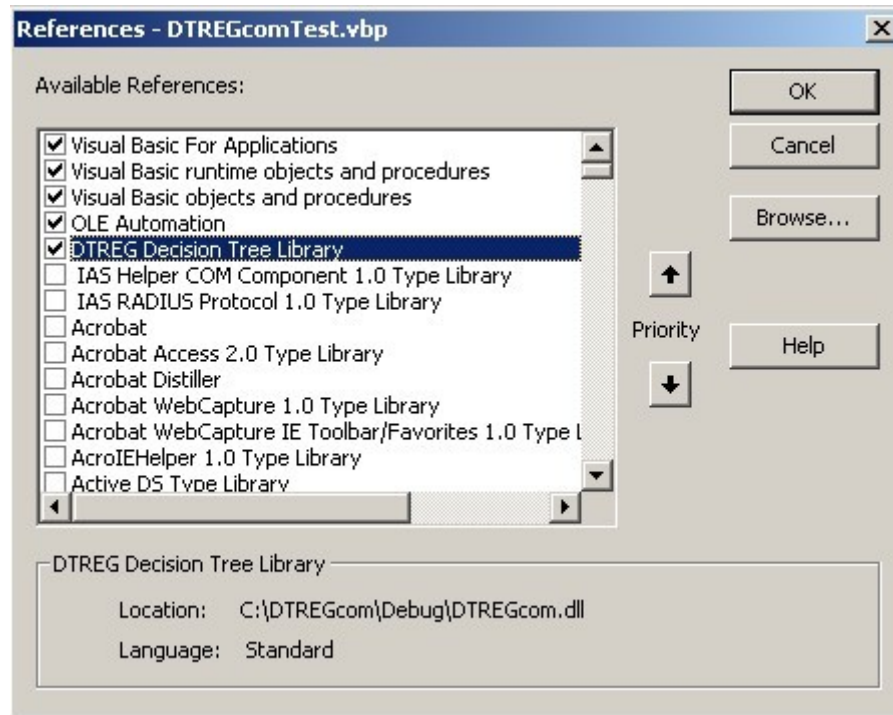
It is easy to call DTREG methods from Visual Basic because Visual Basic has excellent support for COM objects.

Referencing DTREG from Visual Basic

The first step in using DTREG with a Visual Basic program is to tell Visual Basic to reference DTREG. To do this, start Visual Basic, then click "Project" on its main menu, then click "References..." on the drop-down menu.



A popup screen will appear with a list of the registered COM objects. Search through the list and check the box next to “DTREG Decision Tree Library”. Then click “OK” to exit from the references screen.



If the “DTREG Decision Tree Library” entry doesn’t appear in the references list, then the DTREG COM library has not been properly registered, and you should perform the registration procedure described above.

Once you have established a reference to DTREG within Visual Basic, you can add declarations and method calls to your Basic programs.

Declaring the DTREG COM object in a Visual Basic program

Each Visual Basic procedure that calls a DTREG method must contain the following declarations:

```
Dim dtreg As DTREGCOMLib.dtreg  
Set dtreg = New DTREGCOMLib.dtreg
```

This declares a variable named “dtreg” to reference the DTREG COM library. Once you do that, you can then reference DTREG methods by typing “dtreg.methodname()”. (You do not have to name the variable “dtreg”, but it is recommended that you do so to make it clear that the methods are part of the DTREG COM library.)

Example Visual Basic program

Here is a complete example Visual Basic program that calls the DTREG library:

```
Private Sub RunTest_Click()
'
' Reference the DTREG COM library.
'
Dim dtreg As DTREGCOMLib.dtreg
Set dtreg = New DTREGCOMLib.dtreg
'
' Miscellaneous variable declarations.
'
Dim ProjectFile As String
Dim ModelType, status, index As Long
Dim NumVar, NumCat As Long
Dim VarClass, VarType As Long
Dim VarName, CatLabel As String
Dim ixSepalLength, ixSepalWidth As Long
Dim ixPetalLength, ixPetalWidth As Long
Dim ixSpecies As Long
Dim PredictedClass As String
Dim CatProb As Double
'
' Open the DTREG project file (TreeBoostIris.dtr).
'
ProjectFile = "c:\DTREG\Test\TreeBoostIris.dtr"
status = dtreg.OpenProjectFile(ProjectFile)
If (status <> 0) Then
    boxStatus = "Error opening project file: " + Format(status, "###")
    Stop
End If
'
' Find out what type of model this is
'
ModelType = dtreg.ModelType
'
' Find out how many variables are in the model.
'
NumVar = dtreg.NumberOfVariables
'
' Check the name and properties of each variable.
'
For index = 0 To NumVar - 1
    VarName = dtreg.VariableName(index)
    VarClass = dtreg.VariableClass(index)
    VarType = dtreg.VariableType(index)
Next
```

```

' Get the index numbers of the variables variables.
'
ixSpecies = dtreg.VariableIndex("Species")
ixSepalLength = dtreg.VariableIndex("Sepal length")
ixSepalWidth = dtreg.VariableIndex("Sepal width")
ixPetalLength = dtreg.VariableIndex("Petal length")
ixPetalWidth = dtreg.VariableIndex("Petal width")
'
' Set the values of the predictors we want to score.
'
status = dtreg.SetVariableValue(ixSepalLength, 5.1)
status = dtreg.SetVariableValue(ixSepalWidth, 3.5)
status = dtreg.SetVariableValue(ixPetalLength, 1.4)
status = dtreg.SetVariableValue(ixPetalWidth, 0.2)
'
' Compute the predicted target category.
'
PredictedClass = dtreg.PredictedTargetCategory
'
' See if any error occurred during the computation.
'
status = dtreg.LastStatus
If status <> 0 Then
    boxStatus = "Error computing target: " + Format(status, "##")
    Stop
End If
'
' If we are using a TreeBoost model, check the probabilities
' for each of the target variable categories.
'
If ModelType = 2 Then
    NumCat = dtreg.NumberOfCategories(ixSpecies)
    For index = 0 To NumCat - 1
        CatLabel = dtreg.CategoryLabel(ixSpecies, index)
        CatProb = dtreg.TargetCategoryProbability(index)
    Next
End If

End Sub

```

Calling DTREG from ASP Scripts

It is easy to call DTREG methods from ASP (Active Server Page) scripts. You must declare an object to reference the DTREG COM library. This is done by using the following statement:

```
set dtreg = server.createobject("dtregcom.dtreg")
```

After doing that, you can invoke DTREG methods by specifying `dtreg.method()` in the script. Note, unlike Visual Basic, there is no way to set up a “Reference” to the DTREG COM library in ASP scripts. Because of this, if you mistype the name of a method or specify an incorrect number of arguments, the error will not be detected until the ASP page attempts to execute the statement with the error.

Here is the header for an ASP script that declares a reference to the DTREG COM library:

```
<HTML>
<%
    set dtreg = server.createobject("dtregcom.dtreg")
    ival = dtreg.OpenProjectFile("C:\DTREG\Iris.dtr")
    response.write "OpenProject result = " & ival & "<p>"
%>
</HTML>
```

Methods and Properties in the DTREG COM Library

The DTREG library uses both **methods** and **properties**. The difference between a method and a property is that the name of a method is followed by parentheses enclosing one or more argument values. A method is sometimes called a **function**. A property is referenced using its name with no parentheses or arguments.

Here is an example of a DTREG method:

```
Index = dtreg.VariableIndex("age")
```

Here is an example of a DTREG property:

```
Numvar = dtreg.NumberOfVariables
```

LastStatus -- Status code for the last operation

Property: LastStatus As Long

The LastStatus property returns a status code indicating whether the last operation performed by the DTREG library was successful or not. If the last operation was completed successfully, a value of 0 (zero) is returned. If some error occurred on the last operation, the value indicates what type of error occurred.

List of status codes:

0	No error
1	Unable to open project file
2	The project file is corrupt
3	No DTREG project file is currently open
4	The variable name or index is not valid
5	Invalid category value
6	Unable to compute a score for the predictor values
7	No model has been computed for the project

OpenProjectFile() -- Open a DTREG project file

Method: OpenProjectFile (*inFileName* As String) As Long

The OpenProjectFile() method opens a DTREG project file (usually with a .dtr file type), loads the information from it into memory and returns a status code indicating if the operation was successful or not (the status code also can be retrieved using the LastStatus property). The GUI version of the DTREG program is used to create a project file.

You must call `OpenProjectFile` to open a project file before you can use any other method or property. If a project file is open when you call `OpenProjectFile`, the currently-open project is closed and then the new project is opened.

Arguments:

inFileName = A string containing the name of the DTREG project file. It is best to provide a full file specification including the device name and directory.

Returned value:

The value 0 is returned by `OpenProjectFile()` if the project file was successfully opened. A non-zero status code is returned if there were compile errors (see the `LastStatus` property for a list of the error code values).

ModelType -- Get type of model

Property: `ModelType` As Long

The `ModelType` property returns a value that indicates what type of model is described by the current project file. These values can be returned:

- 1** = Single tree model
- 2** = TreeBoost model
- 3** = Decision tree forest model
- 4** = Logistic regression model
- 5** = SVM model
- 7** = Linear discriminant analysis model
- 9** = Multi-layer feed-forward neural network
- 10** = PNN or GRNN neural network
- 11** = RBF neural network
- 12** = Cascade correlation network
- 13** = Gene Expression Programming (GEP)

NumberOfVariables -- Get variable count from a project

Property: `NumberOfVariables` As Long

The `NumberOfVariables` property returns a count of the total number of variables defined in the project. This count includes the target variable, predictor variables, weight variable and any unused variables.

VariableIndex -- Get index number for a variable

Method: VariableIndex (*inVariableName* As String) As Long

Many of the methods in DTREG use index numbers to identify variables. The VariableIndex method converts the name of a variable to its corresponding index number. The index number of the first variable is 0 (zero).

Arguments:

InVariableName is a string containing the name of the variable whose index number is to be found.

Returned value:

The returned value of the method is the index number of the variable. A value of -1 is returned if no variable can be found with the specified name. Use the LastStatus property to check for errors after calling VariableIndex.

VariableName -- Get the name of a variable from its index number

Method: VariableName (*inVariableIndex* As Long) As String

The VariableName method is the complement of the VariableIndex method. The VariableName method takes the index number of a variable as the argument and returns the name of the variable as its result.

Arguments:

InVariableIndex is a numeric (long) value with the index number of the variable. You can use the VariableIndex method to convert the name of a variable to its index number.

VariableClass -- Get the class of a variable

Method: VariableClass (*inVariableIndex* As Long) As Long

The VariableClass method returns a number indicating the class of a variable. There are four possible class values:

- 0** = Unused variable
- 1** = Predictor variable
- 2** = Target variable
- 3** = Weight variable

Arguments:

InVariableIndex is a numeric (long) value with the index number of the variable. You can use the *VariableIndex* method to convert the name of a variable to its index number.

VariableType -- Get the type of a variable

Method: *VariableType* (*InVariableIndex* As Long) As Long

The *VariableType* method returns a number indicating the type of a variable. There are two type values:

- 0 = Continuous variable (e.g., age, income, height, weight).
- 1 = Categorical variable (e.g., sex, race, religion)

Arguments:

InVariableIndex is a numeric (long) value with the index number of the variable. You can use the *VariableIndex* method to convert the name of a variable to its index number.

VariableImportance -- Get the importance of a variable

Method: *VariableImportance* (*InVariableIndex* As Long) As Double

The *VariableImportance* method returns a double precision real number in the range 0.0 to 1.0 indicating the relative importance of a predictor variable. The most important variable has an importance score of 1.0. Other variables have relatively smaller importance scores. Note: in the report generated by DTREG, the importance scores are multiplied by 100.

Arguments:

InVariableIndex is a numeric (long) value with the index number of the variable. You can use the *VariableIndex* method to convert the name of a variable to its index number. The *VariableImportance* function should only be called with index numbers for predictor variables.

NumberOfCategories -- Get number of categories of a variable

Method: *NumberOfCategories* (*InVariableIndex* As Long) As Long

The *NumberOfCategories* method returns the number of categories for a categorical variable. For example, the number of categories for a “sex” variable will be 2.

Arguments:

InVariableIndex is a numeric (long) value with the index number of the variable. You can use the *VariableIndex* method to convert the name of a variable to its index number.

CategoryIndex -- Get index number for a category

Method: *CategoryIndex* (*InVariableIndex* As Long, *InCategoryLabel* As String) As Long

The *CategoryIndex* method returns an index number that identifies a category of a categorical variable. For example, a categorical “sex” variable might have two category labels, “male” and “female”. The *CategoryIndex* method takes the index number of a variable and a category label as its arguments and returns the corresponding category index number. The index number of the first category is 0 (zero).

Arguments:

InVariableIndex is a numeric (long) value with the index number of the variable. You can use the *VariableIndex* method to convert the name of a variable to its index number.

InCategoryLabel is a string containing the label of the category whose index number is to be found.

The returned value of the method is the index number of the category. A value of -1 is returned if no category with the label is associated with the specified variable.

CategoryLabel -- Get the label of a category from its index number

Method: *CategoryLabel* (*InVariableIndex* As Long, *InCategoryIndex* As Long) As String

The *CategoryLabel* method is the complement of the *CategoryIndex* method. The *CategoryLabel* method takes the index number of a category label as the argument and returns the corresponding label string as its result.

Arguments:

InVariableIndex is a numeric (long) value with the index number of the variable. You can use the *VariableIndex* method to convert the name of a variable to its index number.

InCategoryIndex is the index number of the category whose label is desired.

SetVariableValue -- Set the value for a variable

Method: SetVariableValue (*inVariableIndex* As Long, *inValue* As Double) As Long

The SetVariableValue method is used to set the value of a continuous (numeric) predictor variable prior to computing the predicted target value. Use the SetVariableCategory method (described below) to set the value for a categorical variable.

You must use SetVariableValue and/or SetVariableCategory to specify the values of all predictor variables before you reference the PredictedTargetValue or PredictedTargetCategory properties to predict the target value based on the current set of predictor values.

Arguments:

InVariableIndex is a numeric (long) value with the index number of the variable. You can use the VariableIndex method to convert the name of a variable to its index number.

inValue is a double-precision value that is to be set for the specified variable. Use the MissingValue property (see below) to get the value that you should specify to indicate a missing value for a variable.

The returned value of the method is 0 for success or a non-zero status code if an error occurs. See the description of the LastStatus method for information about status codes.

MissingValue -- Numeric value to use for missing values

Property: MissingValue As Double

The MissingValue property returns a numeric value that can be used with the SetVariableValue method to indicate a missing value for a variable.

SetVariableCategory -- Set the category for a variable

Method: SetVariableCategory (*inVariableIndex* As Long, *inCategoryLabel* As String) As Long

The SetVariableCategory method is used to set the value of a categorical predictor variable prior to computing the predicted target value. Use the SetVariableValue method (described above) to set the value for a continuous (numeric) variable.

You must use SetVariableValue and/or SetVariableCategory to specify the values of all predictor variables before you reference the PredictedTargetValue or

PredictedTargetCategory properties to predict the target value based on the current set of predictor values.

Arguments:

InVariableIndex is a numeric (long) value with the index number of the variable. You can use the *VariableIndex* method to convert the name of a variable to its index number.

inCategoryLabel is a string value with the category label that is to be set for the specified variable. If the variable has numeric category values, you may specify a numeric value for the category label instead of a string.

For missing values, specify either a null (empty) string or a string consisting of a single question-mark (“?”) character.

The returned value of the method is 0 for success or a non-zero status code if an error occurs. See the description of the *LastStatus* method for information about status codes.

PredictedTargetValue -- Compute predicted target value

Property: PredictedTargetValue As Double

When you reference the *PredictedTargetValue* property, DTREG computes the predicted value of the target variable using the current model and the predictor variable values most recently set using the *SetVariableValue* and/or *SetVariableCategory* methods. This property returns a double-precision numeric value that is appropriate when the target variable is continuous. Use the *PredictedTargetCategory* property to get the predicted value for categorical target variables.

If you need to use the predicted target value more than once, you should assign the value of *PredictedTargetValue* to a work variable, and then use the work variable rather than causing DTREG to recompute the target value by referencing *PredictedTargetValue* multiple times for the same set of predictor values.

Here is an outline of how to use *PredictedTargetValue* to score multiple records:

Call *OpenProjectFile* to open the DTREG project file.

Start of loop

 Call *SetVariableValue* and/or *SetVariableCategory* to set predictor values

 Reference *PredictedTargetValue* to compute the predicted target value

 Reference *LastStatus* to check whether any errors occurred.

End of loop

If an error occurs when computing the predicted target value, `PredictedTargetValue` returns -1.0 as the predicted value. You should use the `LastStatus` property to check the status of the computation after referencing `PredictedTargetValue`.

PredictedTargetCategory -- Compute predicted target category

Property: `PredictedTargetCategory` As String

When you reference the `PredictedTargetCategory` property, DTREG computes the predicted category of the target variable using the current model and the predictor variable values most recently set using the `SetVariableValue` and/or `SetVariableCategory` methods. This property returns a string value which is appropriate when the target variable is categorical. For example, the predicted category might be “male”, “female”, “purchase”, “no-purchase”, etc. Use the `PredictedTargetValue` property to get the predicted value for continuous (numeric) target variables.

If you need to use the predicted target value more than once, you should assign the value of `PredictedTargetCategory` to a work variable, and then use the work variable rather than causing DTREG to recompute the target value by referencing `PredictedTargetCategory` multiple times for the same set of predictor values.

Here is an outline of how to use `PredictedTargetCategory` to score multiple records:

Call `OpenProjectFile` to open the DTREG project file.

Start of loop

- Call `SetVariableValue` and/or `SetVariableCategory` to set predictor values.

- Reference `PredictedTargetCategory` to compute the predicted target category.

- Reference `LastStatus` to check whether any errors occurred.

End of loop

If an error occurs when computing the predicted target category, `PredictedTargetCategory` returns a null (empty) string as the predicted value. You should use the `LastStatus` property to check the status of the computation after referencing `PredictedTargetCategory`.

TargetCategoryProbability -- Compute probability of target category

Method: `TargetCategoryProbability(inCategoryIndex` As Long) As Double

If you are predicting target categories for a classification tree that uses a `TreeBoost`, `Decision Tree Forest`, `Logistic Regression`, `SVM` or `LDA` model, you can use the `TargetCategoryProbability` method to obtain the computed probability score for each possible category of the target variable. This method can be used only if (1) the target

variable is categorical, and (2) a TreeBoost, Decision Tree Forest, Logistic Regression SVM or LDA model is being used.

You must reference the `PredictedTargetCategory` property (see above) to evaluate the model for the current predictor values before you use the `TargetCategoryProbability` method to get the probability values. Probability values are in the range 0.00 to 1.00.

Arguments:

inCategoryIndex is the index of the target category whose probability is being requested. You can use the `CategoryIndex` method to convert a category label to a category index number.

Index

A

ASP scripts, 6

C

CategoryIndex method, 12

CategoryLabel method, 12

D

Data types, 2

DTREG project file, 8

L

LastStatus property, 8

M

MissingValue property, 13

ModelType property, 9

N

Number of variables, 9

NumberOfCategories method, 11

NumberOfVariables property, 9

O

OpenProjectFile method, 8

P

PredictedTargetCategory property, 15

PredictedTargetValue property, 14

Project file, 8

R

Registering DTREGcom.dll, 2

REGSVR32, 2

S

SetVariableCategory method, 13

SetVariableValue method, 13

Sherrod, Phillip H., 1

T

TargetCategoryProbability property,
15

V

VariableClass method, 10

VariableImportance method, 11

VariableIndex method, 10

VariableName method, 10

VariableType method, 11

VARIANT data types, 2

Visual Basic, 3