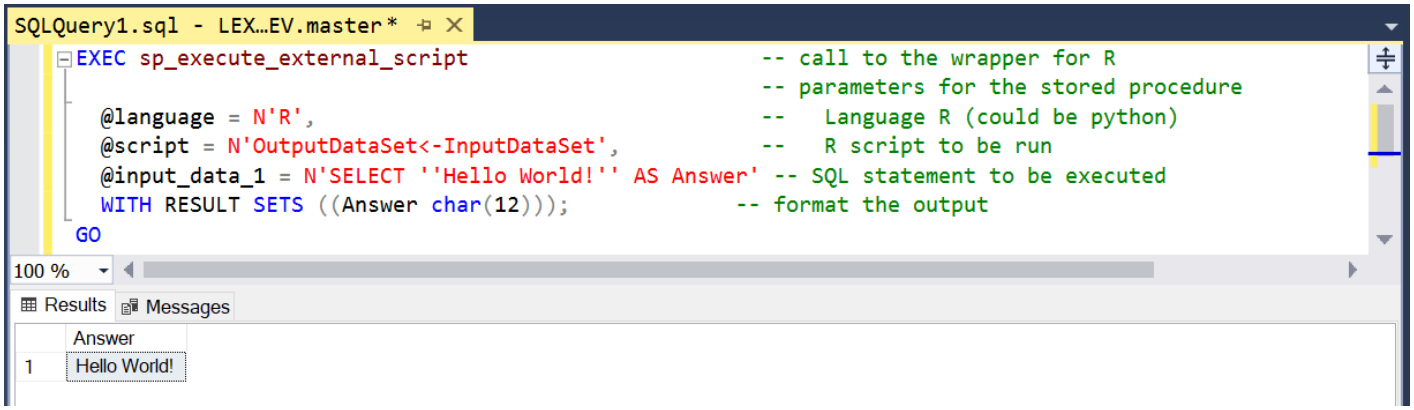
Using R inside of SQL Server (i.e., using R in SSMS)

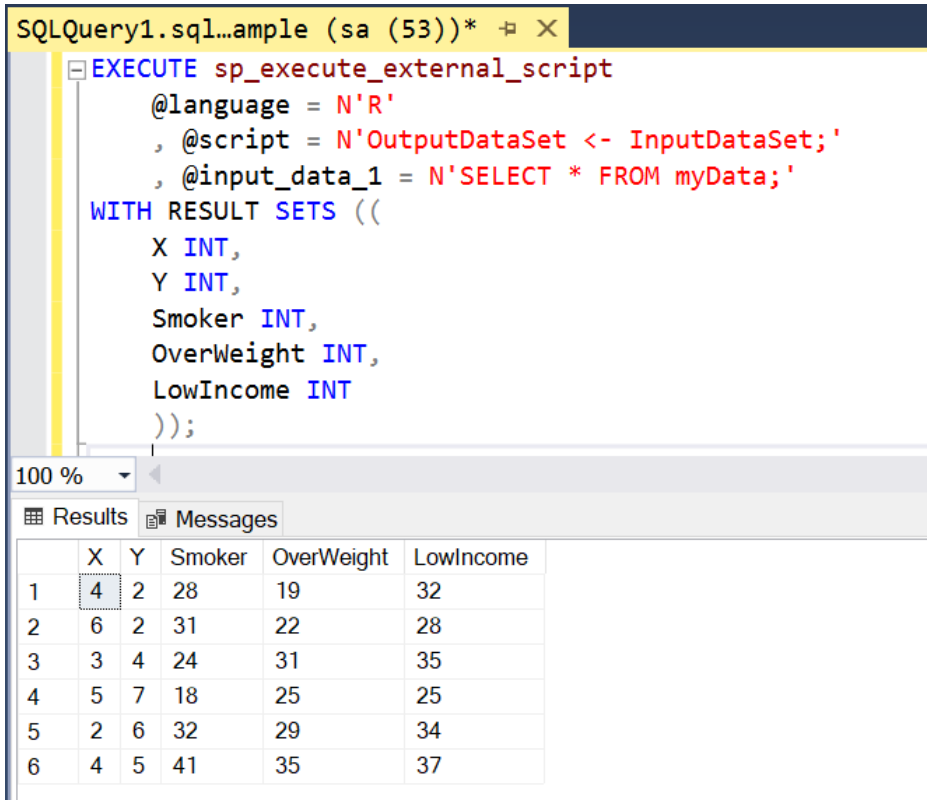
Basically what happens is that after SQL Server 2017 Machine Learning Services has been installed in an SQL Server instance, a set of stored procedures is available that can be used to wrap R code and execute it within the SQL Server engine (i.e., on the server-side).

Here is the quintessential example – Hello World!



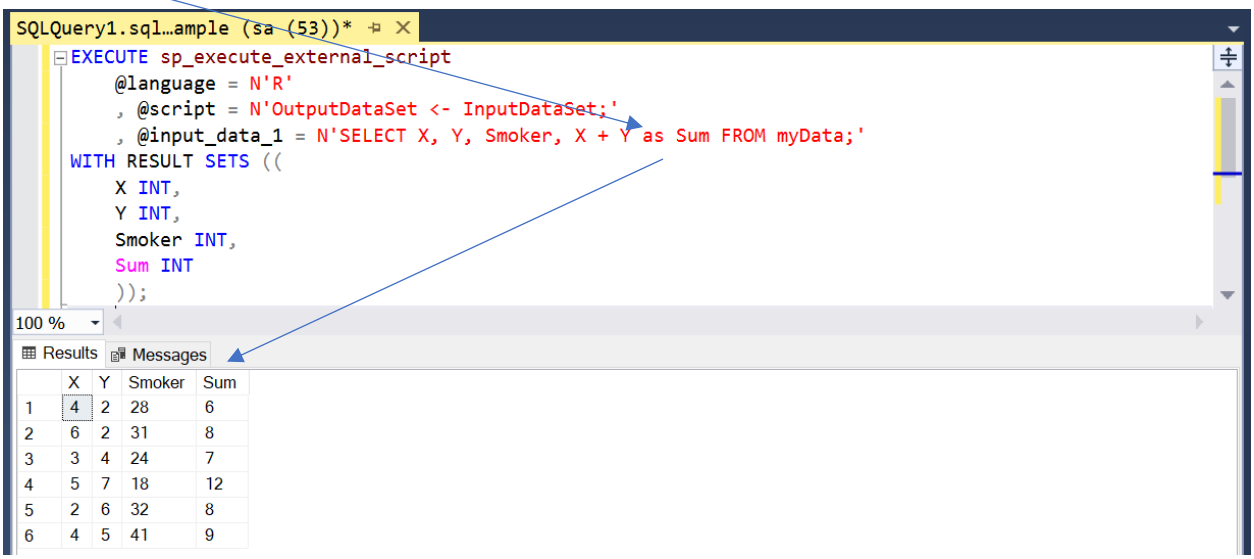
The important point is that the R code is passed as a parameter to a special stored procedure that passes the code off to Machine Learning Services where it is executed and the result set is sent back. If you don't assign a specific name to your input data, the default variable name would be *InputDataSet*.

A simple query against the test database yields



	X	Y	Smoker	OverWeight	LowIncome
1	4	2	28	19	32
2	6	2	31	22	28
3	3	4	24	31	35
4	5	7	18	25	25
5	2	6	32	29	34
6	4	5	41	35	37

Create a new column in the result set.



	X	Y	Smoker	Sum
1	4	2	28	6
2	6	2	31	8
3	3	4	24	7
4	5	7	18	12
5	2	6	32	8
6	4	5	41	9

Creating a Predictive Model

This is an example from https://docs.microsoft.com/en-us/sql/advanced-analytics/tutorials/quickstart-r-create-predictive-model?view=sql-server-2017

When R is installed there are a number of example data sets installed too. This example uses one of these datasets.

The task is to build a model that is a simple generalized linear model (GLM) that predicts probability that a vehicle has been fitted with a manual transmission. The process uses the **mtcars** dataset included with R.

The first step is to create a database, a table, and load the data.

```
SQLQuery2.sql…Model (sa (52))*
-- Predictive Model code
-- database
use master;
create database PredictModel;
go
use PredictModel;
go
-- table
CREATE TABLE dbo.MTCars(
    mpg decimal(10, 1) NOT NULL,
    cyl int NOT NULL,
    disp decimal(10, 1) NOT NULL,
    hp int NOT NULL,
    drat decimal(10, 2) NOT NULL,
    wt decimal(10, 3) NOT NULL,
    qsec decimal(10, 2) NOT NULL,
    vs int NOT NULL,
    am int NOT NULL,
    gear int NOT NULL,
    carb int NOT NULL
);
INSERT INTO dbo.MTCars
EXEC sp_execute_external_script    -- this is R code
        @language = N'R'
        , @script = N'MTCars <- mtcars;'
        , @input_data_1 = N''
        , @output_data_1_name = N'MTCars';
select * from MTCars;
```

Results Messages

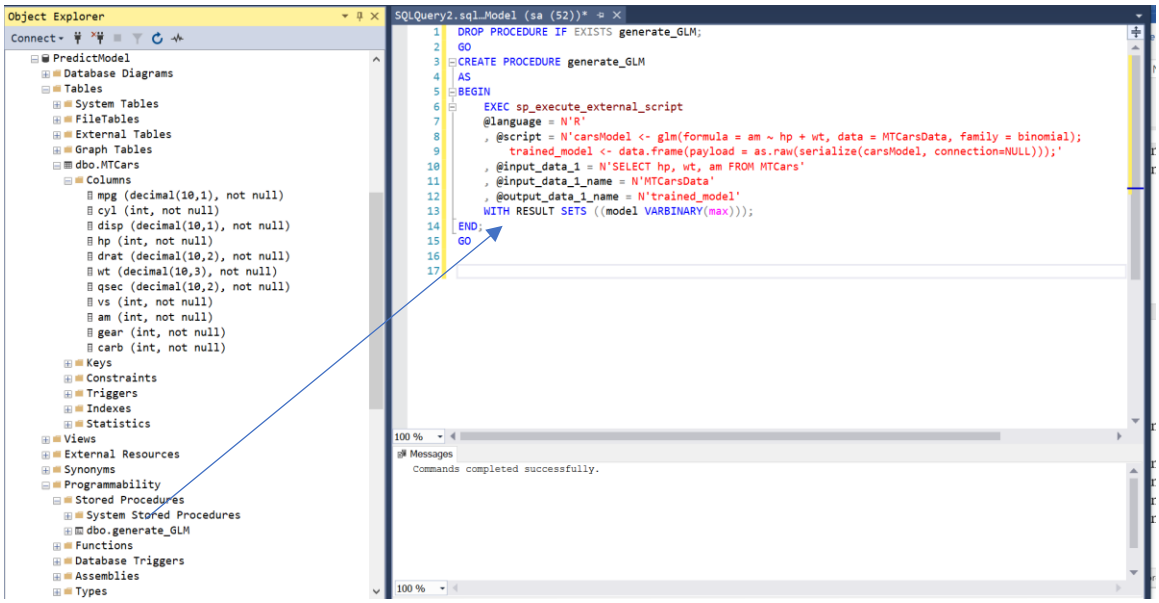
	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
2	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
3	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
4	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
5	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
6	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
7	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
8	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
9	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
10	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
11	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
12	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
13	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
14	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
15	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
16	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
17	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
18	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
19	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
20	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
21	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
22	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
23	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
24	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
25	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
26	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
27	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
28	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
29	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
30	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
31	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
32	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

There are 32 rows of data in the dataset.

Next create the model. The code is

```
SQLQuery2.sql…Model (sa (52))*
 1  DROP PROCEDURE IF EXISTS generate_GLM;
 2  GO
 3  CREATE PROCEDURE generate_GLM
 4  AS
 5  BEGIN
 6      EXEC sp_execute_external_script
 7      @language = N'R'
 8      , @script = N'carsModel <- glm(formula = am ~ hp + wt, data = MTCarsData, family = binomial);
 9          trained_model <- data.frame(payload = as.raw(serialize(carsModel, connection=NULL)));'
10      , @input_data_1 = N'SELECT hp, wt, am FROM MTCars'
11      , @input_data_1_name = N'MTCarsData'
12      , @output_data_1_name = N'trained_model'
13      WITH RESULT SETS ((model VARBINARY(max)));
14  END;
15  GO
```

Line 1: usual best practice, drop the stored procedure before creating a new version
Line 3: stored procedure to generate the generalized linear model
Line 6: beginning for the wrapper for the R code
Line 7: language to be used is R
Line 8: build the new model (https://www.rdocumentation.org/packages/stats/versions/3.6.0/topics/glm)
glm is an R function used to create the model – it has several parameters
formula to be use in the model am ~ hp + wt defines **am** as dependent (variable)
hp (horse power) and **wt** (weight) are the independent variables
data is a data frame containing the data
family is the error distribution (in this case the binomial distribution)
Line 9: creates the trained_model (an object name) as a data.frame (a class) from a payload as raw data that is serialized (flattened structure) from carsModel which is local (no connection)
Line 10: the SQL statement to retrieve (into input_data_1) the data from the database table (MTCars)
Line 11: assign the name MTCarsData to the object input_data_1
Line 12: assign the name trained_model to the object output_data_1
Line 13: the model is binary, i.e., VARBINARY(max)



The model may be saved in the database – create a table to hold the model, recall the model is binary.

SQLQuery2.sql…Model (sa (52))*

```
1  CREATE TABLE GLM_models (
2      model_name varchar(30) not null default('default model') primary key,
3      model varbinary(max) not null
4  );
5
6  INSERT INTO GLM_models(model)
7  EXEC generate_GLM;
8
9
```

100 %

Messages

```
STDERR message(s) from external script:
During startup - Warning message:
In setJsonDatabasePath(system.file("extdata/capabilities.json",  :
  bytecode version mismatch; using eval


(1 row affected)
```

Lines 1-4: Build the table
Lines 6-7: Save the model

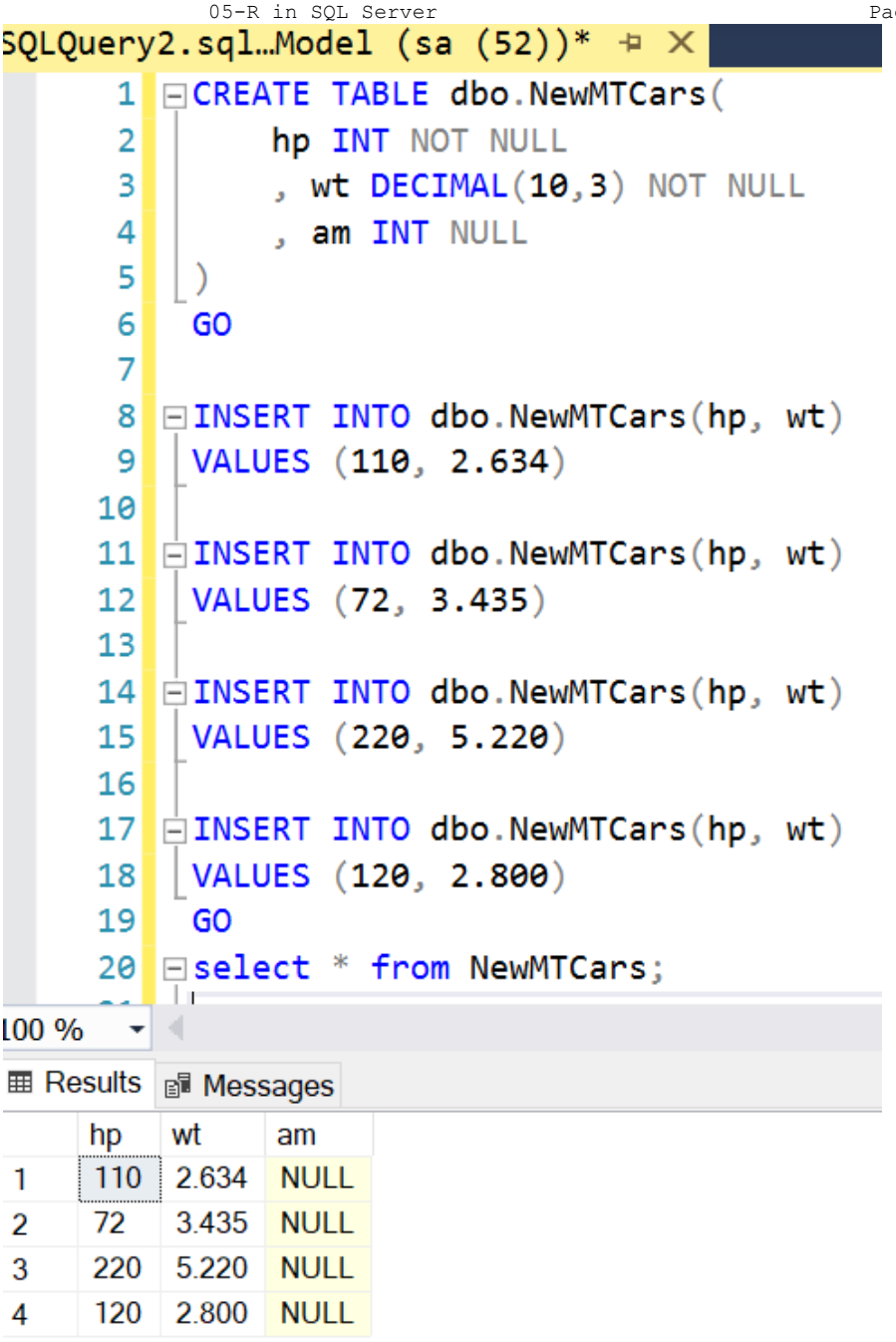
The Message presented is not relevant to the task at hand.

The content of the GLM_Models table is

LEXINGTON\SQLS…dbo.GLM_models

	model_name	model
	default model	<Binary data>
▶*	NULL	NULL

That completes the model (basically the training process). Next, data to test against the model is needed. So a NewMTCars table is created and loaded with hp and wt data.



	hp	wt	am
1	110	2.634	NULL
2	72	3.435	NULL
3	220	5.220	NULL
4	120	2.800	NULL

All the pieces are in place to run the new data against the model to determine (a probability based on the training data) as to whether the new cars have a manual transmission. Consider the following code and results:

SQLQuery2.sql…Model (sa (52))*

```
1  DECLARE @glmmodel varbinary(max) =
2      (SELECT model FROM dbo.GLM_models WHERE model_name = 'default model');
3
4  EXEC sp_execute_external_script
5      @language = N'R'
6      , @script = N'
7              current_model <- unserialize(as.raw(glmmodel));
8              new <- data.frame(NewMTCars);
9              predicted.am <- predict(current_model, new, type = "response");
10             str(predicted.am);
11             OutputDataSet <- cbind(new, predicted.am);
12             '
13     , @input_data_1 = N'SELECT hp, wt FROM dbo.NewMTCars'
14     , @input_data_1_name = N'NewMTCars'
15     , @params = N'@glmmodel varbinary(max)'
16     , @glmmodel = @glmmodel
17 WITH RESULT SETS ((new_hp INT, new_wt DECIMAL(10,3), predicted_am DECIMAL(10,3)));
```

100 %

Results Messages

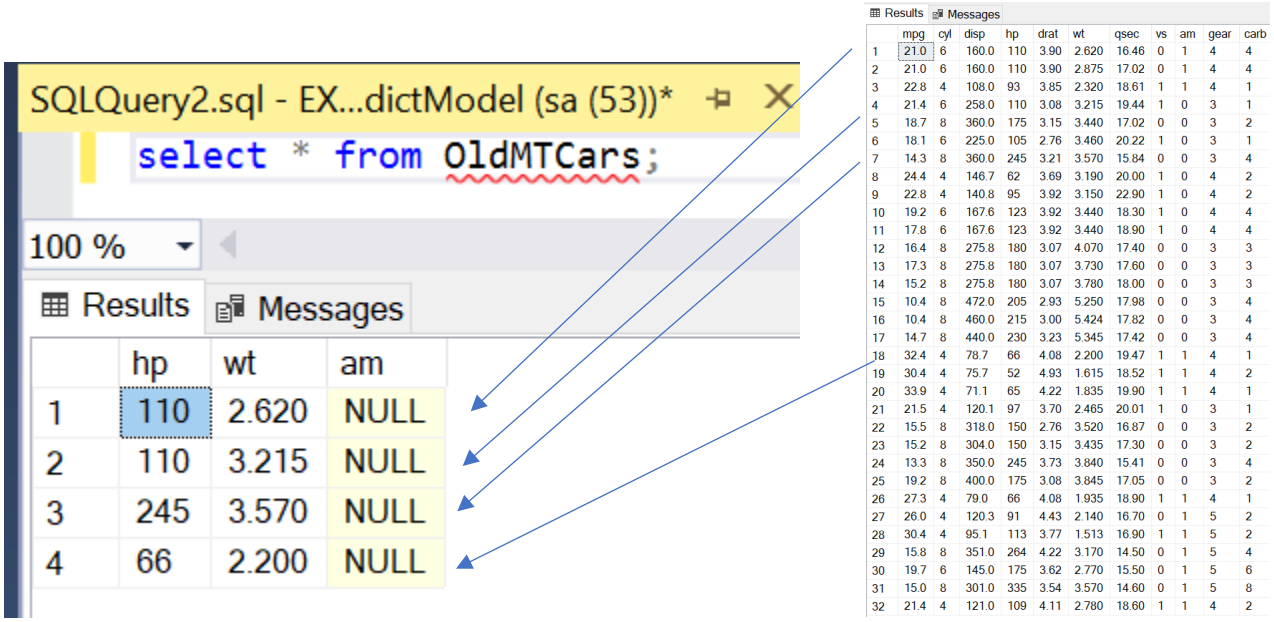
	new_hp	new_wt	predicted_am
1	110	2.634	0.827
2	72	3.435	0.002
3	220	5.220	0.000
4	120	2.800	0.642

Lines 1-2: retrieve the model from the GLM_Models table
Line 4: the wrapper procedure for R code
Line 5: the code is R code
Lines 6-12: the R code to be executed
Line 7: get the current model
Line 8: get the new data
Line 9: run the prediction (https://www.rdocumentation.org/packages/stats/versions/3.6.0/topics/predict.lm) passing in the current_model and the new data and specifying a response type, results go to am.
Line 10: shows the structure of the results (not actually needed)
Line 11: define the output – aligns the result with new data (https://www.rdocumentation.org/packages/base/versions/3.6.0/topics/cbind)
Line 13: SQL statement to gather the new data
Line 14: name for the input
Line 15-16: parameter definitions
Line 17: format for the output

The results indicate that new car 1 has a probability of 0.827 of having a manual transmission based on the data in the training set. Car 2 is not likely to have a manual transmission.

An interesting question is, how well the prediction be if a test is done using a subset of the training data.

A new table OldMTCars contains a subset of rows (1, 4,7, 18) from the training data.



The predicted am values for 1, 4, and 7 are as expected, for 18 0.642 would indicate a manual transmission which agrees with the am value in the training data.

Results Messages

	new_hp	new_wt	predicted_am
1	110	2.634	0.827
2	72	3.435	0.002
3	220	5.220	0.000
4	120	2.800	0.642

(blank)