Advanced Analytics with SQL Server 2016 R

Hands-on Tutorial Guide

Machine Learning & Data Science Summit Singapore, Dec 9th 2016

Microsoft Data Scientist: Fang Zhou

In today's fast-paced world, mobile phone customers have many choices and can easily switch between service providers. Improving customer attrition rates and enhancing a customer's experience are valuable ways to reduce customer acquisition costs and maintain a high-quality service.

Many industries, including mobile providers, use Churn Models to predict which customers are most likely to leave, and to understand which factors cause customers to stop using their service.

In this session, we will show a solution to help customers, especially telecommunication companies, predict customer churns. This solution provides a guidance on how to harness advanced analytics in SQL Server, by leveraging its new feature R Services and familiar SQL tools and technologies. To be specific, it covers the following phases:

- How a data scientist can make use of SQL Server R Services to do data science;
- How to operationalize the R models via T-SQL stored procedures and build an intelligent application

The high-level solution delivered via this telco churn example could be easily applied to other business problems in diverse industries.

Pre-Requisites

1. **Azure Subscription** – You will need an Azure subscription to login to Azure.

2. **Remote Desktop Connection** – You will be using Remote Desktop to connect to the Data Science VM that we have provisioned in Azure to enable you to experience SQL Server 2016 R Services

SQL Server R Services provides a platform for developing intelligent applications that uncover new insights. You can use the rich and powerful R language and the many open source packages to create models and generate predictions using your SQL Server data. Because SQL Server R Services integrates the R language with SQL Server, you can keep analytics close to the data and eliminate the costs and security risks associated with data movement.

SQL Server R Services combines R with a comprehensive set of SQL Server tools and technologies that offer superior performance, security, reliability and manageability. You can deploy R solutions using convenient, familiar tools, and your production applications can call the R runtime and retrieve predictions and visuals using Transact-SQL.

Let’s get started!

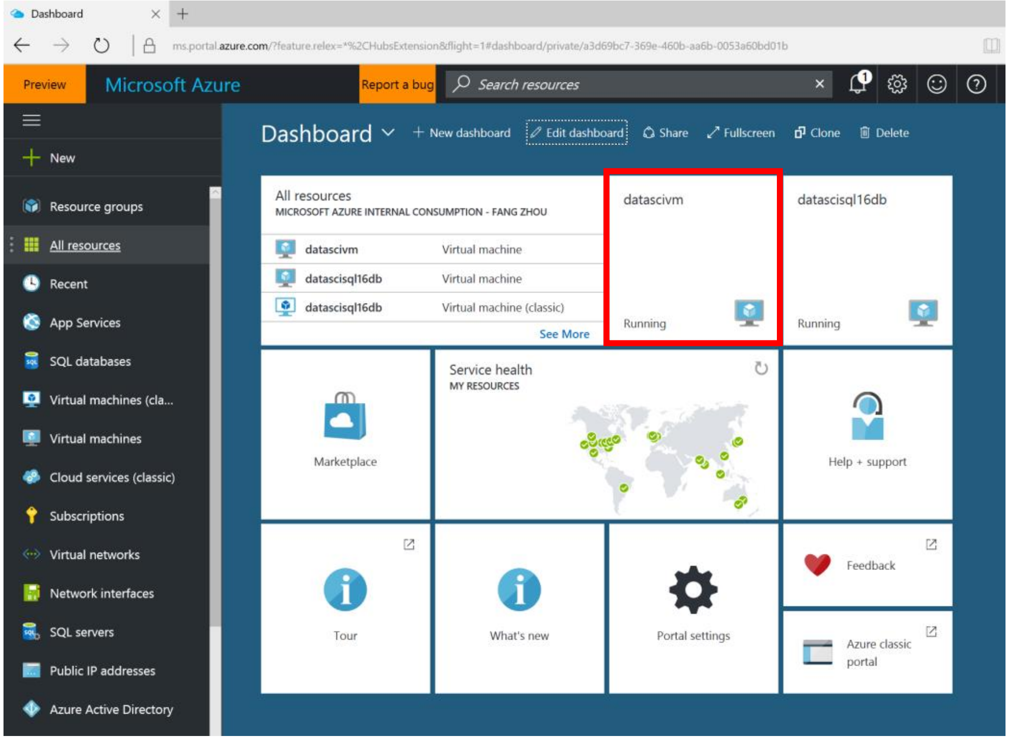
Data Scientist – Exploration and Predictive Modeling

We start the exercise by considering how a data scientist will create, and evaluate the model. Once the model has been built, this is then operationalized via T-SQL.

The Data Science VM contains a rich collection of tools which you will be using to connect to SQL Server 2016. You will use the data science virtual machine to connect to SQL Server 2016.

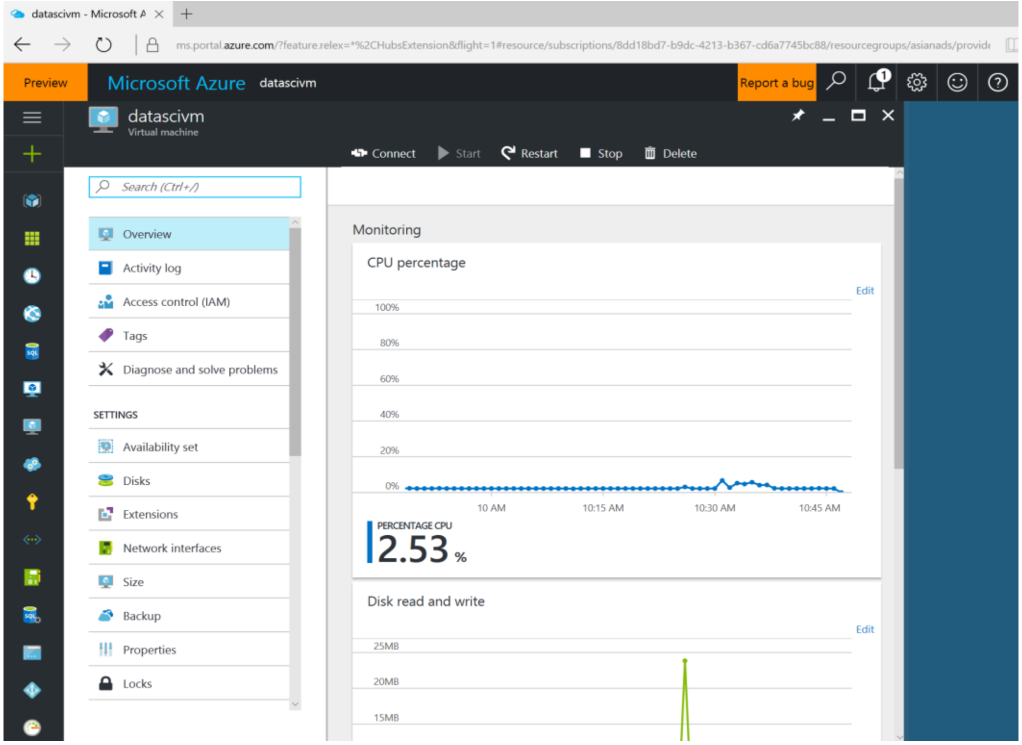
Note: If you have not created the Data Science Virtual Machine, you can use the Azure Portal to create a Data Science Virtual Machine. After the Virtual Machine has been created, you can use the steps below to login.

1. You can click on the Virtual Machine tile and Click Connect to connect to the Data Science Virtual Machine.



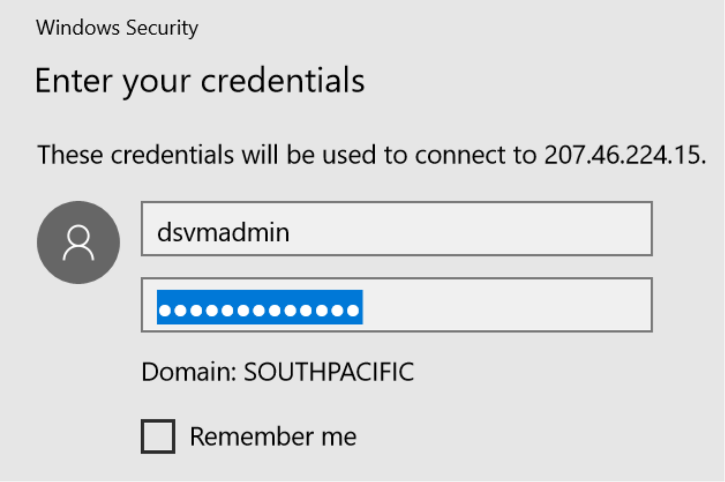
All resources
MICROSOFT AZURE INTERNAL CONSUMPTION - FANG ZHOU

datascivm	Virtual machine
datascisql16db	Virtual machine
datascisql16db	Virtual machine (classic)



2. Open the prompt after the RDP file has been downloaded, and click Connect.

3. Login to the Data Science Virtual machine
   Username: dsvmadmin
   Password: <to be provided>



Windows Security

Enter your credentials

These credentials will be used to connect to 207.46.224.15.

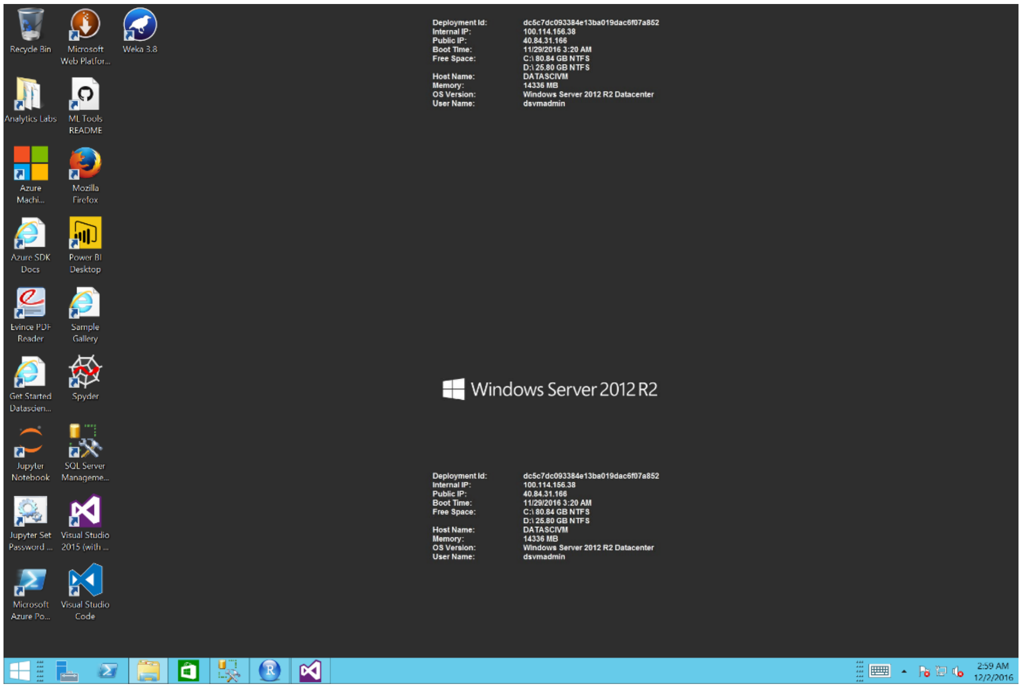
dsvmadmin

●●●●●●●●●●●●●

Domain: SOUTHPACIFIC

☐ Remember me

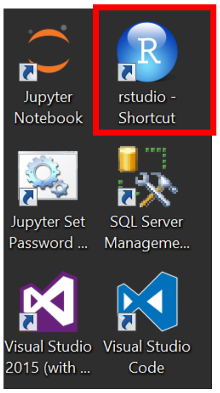
4. After you have login to the Data Science VM, you will see the following.
   You can see the various tools that can help you jumpstart into data science.



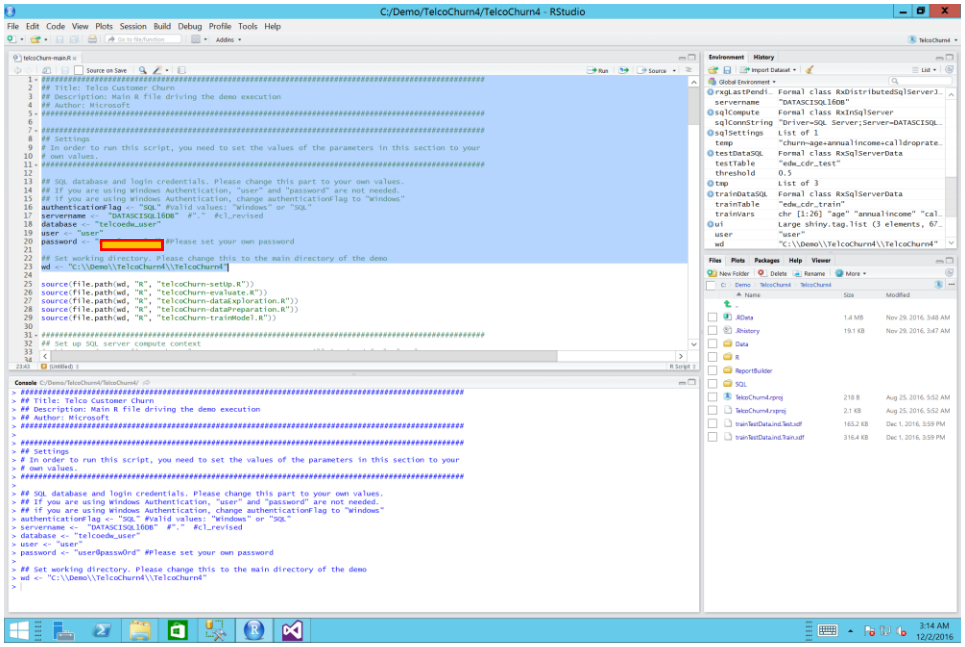
```
Deployment Id:      dc5c7dc093384e13ba019dac6f07a852
Internal IP:        100.114.156.38
Public IP:          40.84.31.166
Boot Time:          11/29/2016 3:20 AM
Free Space:         C:\ 80.84 GB NTFS
                    D:\ 25.80 GB NTFS
Host Name:          DATASCIVM
Memory:             14336 MB
OS Version:         Windows Server 2012 R2 Datacenter
User Name:          dsvmadmin
```

```
Deployment Id:      dc5c7dc093384e13ba019dac6f07a852
Internal IP:        100.114.156.38
Public IP:          40.84.31.166
Boot Time:          11/29/2016 3:20 AM
Free Space:         C:\ 80.84 GB NTFS
                    D:\ 25.80 GB NTFS
Host Name:          DATASCIVM
Memory:             14336 MB
OS Version:         Windows Server 2012 R2 Datacenter
User Name:          dsvmadmin
```

5. Click on **Rstudio**
   Note: You can use any R client (e.g. Visual Studio) to run the R code provided in this hands-on.



6. You will see the R script (telcoChurn-main.R) that have been added to the project.
   Note: If you do not see the telcoChurn-main.R file opened, you can navigate to **C:\\Demo\\TelcoChurn4\\TelcoChurn4** to open the file.



C:/Demo/TelcoChurn4/TelcoChurn4 - RStudio

```
##########################################################################################
## Title: Telco Customer Churn
## Description: Main R file driving the demo execution
## Author: Microsoft
##########################################################################################

##########################################################################################
## Settings
# In order to run this script, you need to set the values of the parameters in this section to your
# own values.
##########################################################################################

## SQL database and login credentials. Please change this part to your own values.
## If you are using Windows Authentication, "user" and "password" are not needed.
## if you are using Windows Authentication, change authenticationFlag to "Windows"
authenticationFlag <- "SQL" #Valid values: "Windows" or "SQL"
servername <-  "DATASCISQL16DB"  #"."  #cl_revised
database <- "telcoedw_user"
user <- "user"
password <- "[redacted] #Please set your own password

## Set working directory. Please change this to the main directory of the demo
wd <- "C:\\Demo\\TelcoChurn4\\TelcoChurn4"

source(file.path(wd, "R", "telcoChurn-setUp.R"))
source(file.path(wd, "R", "telcoChurn-evaluate.R"))
source(file.path(wd, "R", "telcoChurn-dataExploration.R"))
source(file.path(wd, "R", "telcoChurn-dataPreparation.R"))
source(file.path(wd, "R", "telcoChurn-trainModel.R"))

##########################################################################################
## Set up SQL server compute context
```

Console C:/Demo/TelcoChurn4/TelcoChurn4/

```
> ##########################################################################################
> ## Title: Telco Customer Churn
> ## Description: Main R file driving the demo execution
> ## Author: Microsoft
> ##########################################################################################
>
> ##########################################################################################
> ## Settings
> # In order to run this script, you need to set the values of the parameters in this section to your
> # own values.
> ##########################################################################################
>
> ## SQL database and login credentials. Please change this part to your own values.
> ## If you are using Windows Authentication, "user" and "password" are not needed.
> ## if you are using Windows Authentication, change authenticationFlag to "Windows"
> authenticationFlag <- "SQL" #Valid values: "Windows" or "SQL"
> servername <-  "DATASCISQL16DB"  #"."  #cl_revised
> database <- "telcoedw_user"
> user <- "user"
> password <- "user@passw0rd" #Please set your own password
>
> ## Set working directory. Please change this to the main directory of the demo
> wd <- "C:\\Demo\\TelcoChurn4\\TelcoChurn4"
>
```

Environment — Global Environment

```
rxgLastPendi…   Formal class RxDistributedSqlServerJ…
servername      "DATASCISQL16DB"
sqlCompute      Formal class RxInSqlServer
sqlConnString   "Driver=SQL Server;Server=DATASCISQL…
sqlSettings     List of 1
temp            "churn~age+annualincome+calldroprate…
testDataSQL     Formal class RxSqlServerData
testTable       "edw_cdr_test"
threshold       0.5
tmp             List of 3
trainDataSQL    Formal class RxSqlServerData
trainTable      "edw_cdr_train"
trainVars       chr [1:26] "age" "annualincome" "cal…
ui              Large shiny.tag.list (3 elements, 67…
user            "user"
wd              "C:\\Demo\\TelcoChurn4\\TelcoChurn4"
```

Files — C: › Demo › TelcoChurn4 › TelcoChurn4

Name	Size	Modified
..		
.RData	1.4 MB	Nov 29, 2016, 3:48 AM
.Rhistory	19.1 KB	Nov 29, 2016, 3:47 AM
Data		
R		
ReportBuilder		
SQL		
TelcoChurn4.rproj	218 B	Aug 25, 2016, 5:52 AM
TelcoChurn4.rxproj	2.1 KB	Aug 25, 2016, 5:52 AM
trainTestData.ind.Test.xdf	165.2 KB	Dec 1, 2016, 3:59 PM
trainTestData.ind.Train.xdf	316.4 KB	Dec 1, 2016, 3:59 PM

7. Run the above code to specify SQL database and login credentials as well as set the working directory and source pre-defined R scripts.

8. Next, you specify the compute context for the script. The compute context enables you to specify whether the R code runs locally on the machine where the R client is installed, or on the remote SQL Server 2016 instance.

   The <User ID> and <Password> needs to be replaced with the SQL Server Login that has been provided.

   Press Ctrl+Enter to run the selected lines.

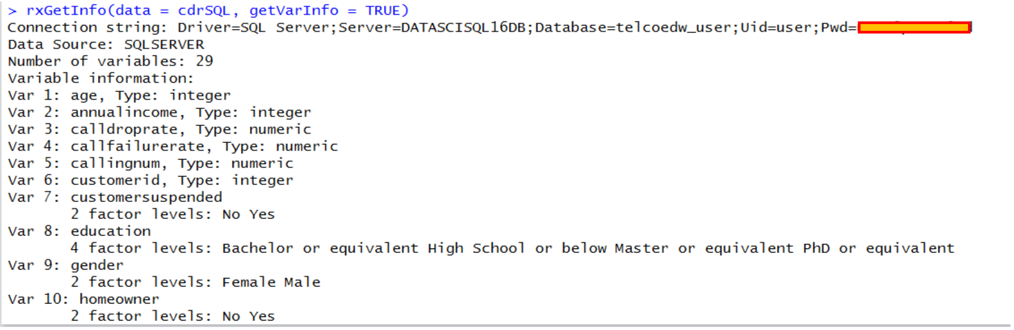
```
#######################################################################
## Set up SQL server compute context
# This part just configure the sql compute context. We are still in the
#default local compute context.
# We will swtich to SQL compute context after loading data into SQL
#tables.
#######################################################################
if (authenticationFlag == "Windows") {
    sqlConnString <- paste("Driver=SQL Server;Server=", servername,
";Database=", database, ";trusted_connection=true", sep = "")
} else if (authenticationFlag == "SQL") {
  sqlConnString <- paste("Driver=SQL Server;Server=", servername,
";Database=", database, ";Uid=", user, ";Pwd=", password, sep = "")
}

sqlCompute <- RxInSqlServer(connectionString = sqlConnString)
```

```
sqlSettings <- vector("list")
sqlSettings$connString <- sqlConnString
```

9. Load the data from local directory into SQL Server Database under local compute context and check the data source information.

```
######################################################################
## Load data into SQL tables
######################################################################
rxSetComputeContext('local')
cdrTable <- "edw_cdr"

cdrFile <- RxTextData(file.path(wd, "Data", "edw_cdr.csv"))

cdrSQL <- RxSqlServerData(table = cdrTable,
                          connectionString = sqlConnString,
                          colInfo = cdrColInfo)

rxDataStep(inData = cdrFile, outFile = cdrSQL, overwrite = TRUE)

## View raw data information
rxGetInfo(data = cdrSQL, getVarInfo = TRUE)
```

After the selected R code executes, you will see the following output in the Output window.

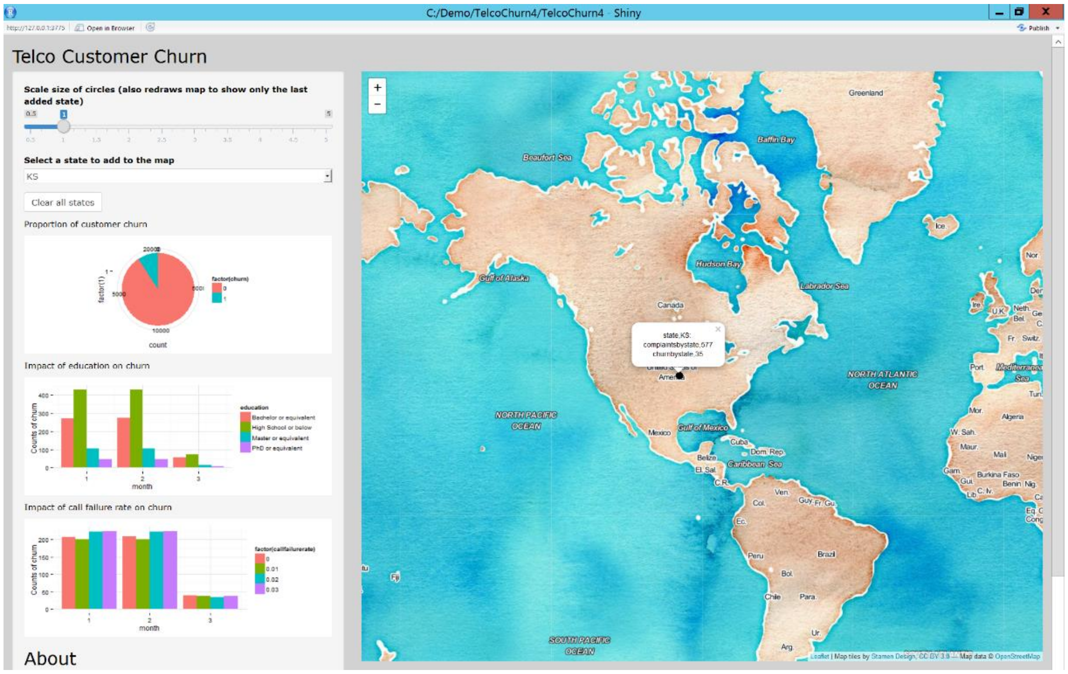


```
> rxGetInfo(data = cdrSQL, getVarInfo = TRUE)
Connection string: Driver=SQL Server;Server=DATASCISQL16DB;Database=telcoedw_user;Uid=user;Pwd=[redacted]
Data Source: SQLSERVER
Number of variables: 29
Variable information:
Var 1: age, Type: integer
Var 2: annualincome, Type: integer
Var 3: calldroprate, Type: numeric
Var 4: callfailurerate, Type: numeric
Var 5: callingnum, Type: numeric
Var 6: customerid, Type: integer
Var 7: customersuspended
       2 factor levels: No Yes
Var 8: education
       4 factor levels: Bachelor or equivalent High School or below Master or equivalent PhD or equivalent
Var 9: gender
       2 factor levels: Female Male
Var 10: homeowner
       2 factor levels: No Yes
```

10. Let's further explore and visualize the data by running the already built shiny application. **Press Ctrl+Enter to run the selected lines.**

```
######################################################################
## Data exploration and visualization
######################################################################
shinyApp(ui, server)
```

After successful execution, an interactive visualization web application will be generated.



C:/Demo/TelcoChurn4/TelcoChurn4 - Shiny

http://127.0.0.1:3775

Telco Customer Churn

Scale size of circles (also redraws map to show only the last added state): 1

Select a state to add to the map: KS

state,KS:
complaintsbystate,577
churnbystate,35

11. Next, we will do data manipulation and feature engineering.

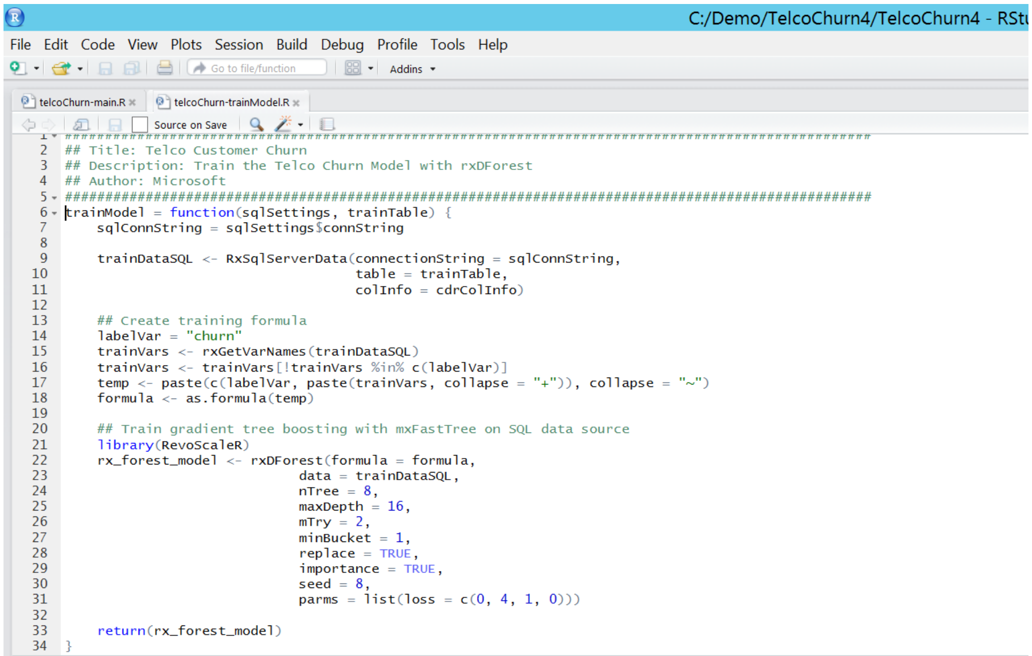
```
########################################################################
## Data preparation and feature engineering
########################################################################

## SQL table names
inputTable <- cdrTable
trainTable <- "edw_cdr_train"
testTable <- "edw_cdr_test"
predTable <- "edw_cdr_pred"

## Data preparation.
# We now delete unnecessary columns, clean missing values, remove
#duplicate rows,
# but more importantly, split the raw data into training and testing
#data sets followed by SMOTE.
system.time({
    dataPreparation(sqlSettings, trainTable, testTable)
})

## View the number of churn events in training and testing data sets.
trainDataSQL <- RxSqlServerData(connectionString = sqlConnString,
                                   table = trainTable,
                                   colInfo = cdrColInfo)
testDataSQL <- RxSqlServerData(connectionString = sqlConnString,
                                  table = testTable,
                                  colInfo = cdrColInfo)
rxGetInfo(trainDataSQL, getVarInfo = T)
rxGetInfo(testDataSQL, getVarInfo = T)
rxSummary( ~ churn, data = trainDataSQL)
rxSummary( ~ churn, data = testDataSQL)
```

12. Next, we will train the decision forest.

    You will notice that we have set the Compute Context to be SQL Server, specified by `rxSetComputeContext(sql)`

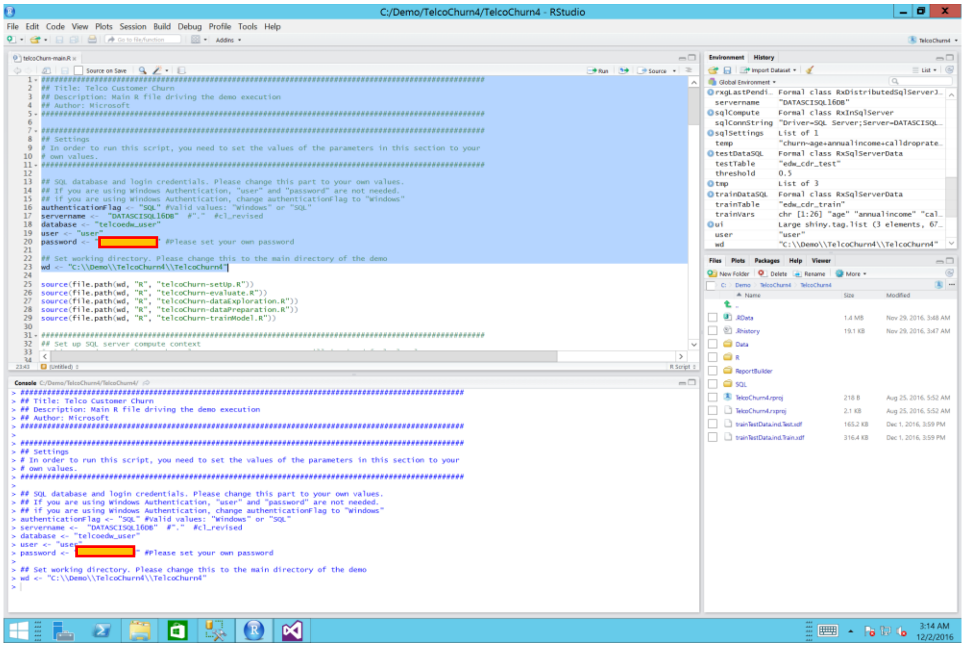
```
#######################################################################
## Train model
#######################################################################
## Switch to sql compute context.
# From now on, all the executions will be done in the SQL server
rxSetComputeContext(sqlCompute)

## Train Decision Forest model with rxDForest
system.time({
rx_forest_model <- trainModel(sqlSettings, trainTable)
})

## View model results
rx_forest_model
plot(rx_forest_model)
rxVarImpPlot(rx_forest_model)
```

You can have a close look at the function trainModel() by opening the R script ‘telcoChurn-trainModel.R’.


C:/Demo/TelcoChurn4/TelcoChurn4 - RStu
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
telcoChurn-main.R
telcoChurn-trainModel.R
Source on Save

```
####################################################################################################
## Title: Telco Customer Churn
## Description: Train the Telco Churn Model with rxDForest
## Author: Microsoft
####################################################################################################
trainModel = function(sqlSettings, trainTable) {
    sqlConnString = sqlSettings$connString

    trainDataSQL <- RxSqlServerData(connectionString = sqlConnString,
                                    table = trainTable,
                                    colInfo = cdrColInfo)

    ## Create training formula
    labelVar = "churn"
    trainVars <- rxGetVarNames(trainDataSQL)
    trainVars <- trainVars[!trainVars %in% c(labelVar)]
    temp <- paste(c(labelVar, paste(trainVars, collapse = "+")), collapse = "~")
    formula <- as.formula(temp)

    ## Train gradient tree boosting with mxFastTree on SQL data source
    library(RevoScaleR)
    rx_forest_model <- rxDForest(formula = formula,
                                 data = trainDataSQL,
                                 nTree = 8,
                                 maxDepth = 16,
                                 mTry = 2,
                                 minBucket = 1,
                                 replace = TRUE,
                                 importance = TRUE,
                                 seed = 8,
                                 parms = list(loss = c(0, 4, 1, 0)))

    return(rx_forest_model)
}
```

13. Putting it all together – Run the whole script by pressing **Ctrl+Alt+R**.



C:/Demo/TelcoChurn4/TelcoChurn4 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
TelcoChurn4
telcoChurn-main.R
Source on Save
Run
Source
Environment History
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
New Folder Delete Rename More
C: Demo TelcoChurn4 TelcoChurn4
23:43
(Untitled)
R Script
3:14 AM
12/2/2016

```
##########################################################################################
## Title: Telco Customer Churn
## Description: Main R file driving the demo execution
## Author: Microsoft
##########################################################################################

##########################################################################################
## Settings
# In order to run this script, you need to set the values of the parameters in this section to your
# own values.
##########################################################################################

## SQL database and login credentials. Please change this part to your own values.
## If you are using Windows Authentication, "user" and "password" are not needed.
## if you are using Windows Authentication, change authenticationFlag to "Windows"
authenticationFlag <- "SQL" #Valid values: "Windows" or "SQL"
servername <-  "DATASCISQL16DB"  #"."  #cl_revised
database <- "telcoedw_user"
user <- "user"
password <- "[redacted]" #Please set your own password

## Set working directory. Please change this to the main directory of the demo
wd <- "C:\\Demo\\TelcoChurn4\\TelcoChurn4"

source(file.path(wd, "R", "telcoChurn-setUp.R"))
source(file.path(wd, "R", "telcoChurn-evaluate.R"))
source(file.path(wd, "R", "telcoChurn-dataExploration.R"))
source(file.path(wd, "R", "telcoChurn-dataPreparation.R"))
source(file.path(wd, "R", "telcoChurn-trainModel.R"))

##########################################################################################
## Set up SQL server compute context
```

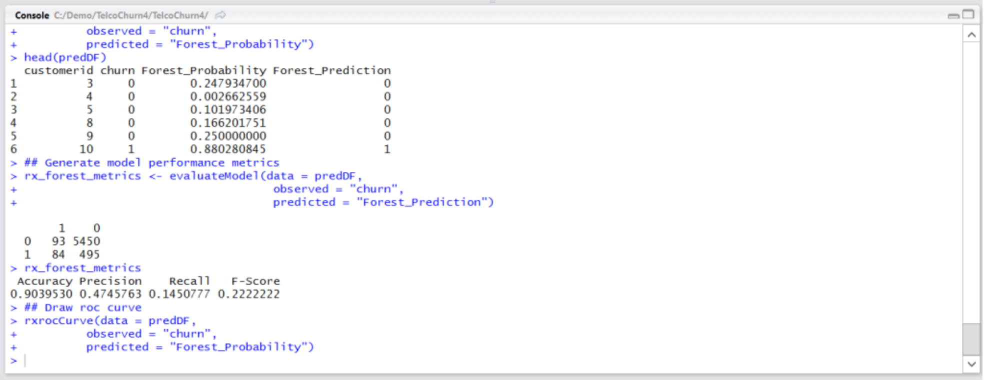
Console C:/Demo/TelcoChurn4/TelcoChurn4/

```
> ##########################################################################################
> ## Title: Telco Customer Churn
> ## Description: Main R file driving the demo execution
> ## Author: Microsoft
> ##########################################################################################
>
> ##########################################################################################
> ## Settings
> # In order to run this script, you need to set the values of the parameters in this section to your
> # own values.
> ##########################################################################################
>
> ## SQL database and login credentials. Please change this part to your own values.
> ## If you are using Windows Authentication, "user" and "password" are not needed.
> ## if you are using Windows Authentication, change authenticationFlag to "Windows"
> authenticationFlag <- "SQL" #Valid values: "Windows" or "SQL"
> servername <-  "DATASCISQL16DB"  #"."  #cl_revised
> database <- "telcoedw_user"
> user <- "user"
> password <- "[redacted]" #Please set your own password
>
> ## Set working directory. Please change this to the main directory of the demo
> wd <- "C:\\Demo\\TelcoChurn4\\TelcoChurn4"
> 
```

```
rxgLastPendi…   Formal class RxDistributedSqlServerJ…
servername      "DATASCISQL16DB"
sqlCompute      Formal class RxInSqlServer
sqlConnString   "Driver=SQL Server;Server=DATASCISQL…
sqlSettings     List of 1
temp            "churn~age+annualincome+calldroprate…
testDataSQL     Formal class RxSqlServerData
testTable       "edw_cdr_test"
threshold       0.5
tmp             List of 3
trainDataSQL    Formal class RxSqlServerData
trainTable      "edw_cdr_train"
trainVars       chr [1:26] "age" "annualincome" "cal…
ui              Large shiny.tag.list (3 elements, 67…
user            "user"
wd              "C:\\Demo\\TelcoChurn4\\TelcoChurn4"
```

Name	Size	Modified
..		
.RData	1.4 MB	Nov 29, 2016, 3:48 AM
.Rhistory	19.1 KB	Nov 29, 2016, 3:47 AM
Data		
R		
ReportBuilder		
SQL		
TelcoChurn4.rproj	218 B	Aug 25, 2016, 5:52 AM
TelcoChurn4.rxproj	2.1 KB	Aug 25, 2016, 5:52 AM
trainTestData.ind.Test.xdf	165.2 KB	Dec 1, 2016, 3:59 PM
trainTestData.ind.Train.xdf	316.4 KB	Dec 1, 2016, 3:59 PM

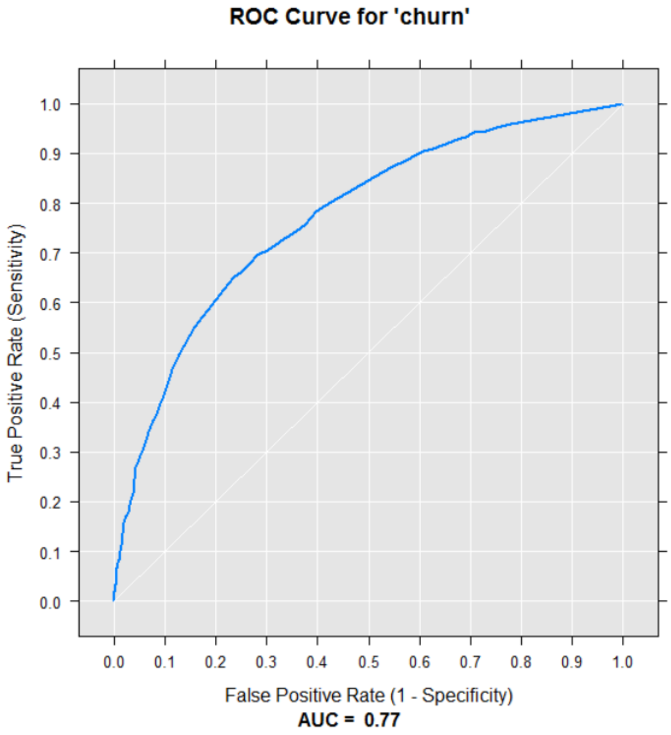
14. After you successfully run the R script, you will see the output in the Console Window.



Console C:/Demo/TelcoChurn4/TelcoChurn4/

```
+           observed = "churn",
+           predicted = "Forest_Probability")
> head(predDF)
  customerid churn Forest_Probability Forest_Prediction
1          3     0        0.247934700                 0
2          4     0        0.002662559                 0
3          5     0        0.101973406                 0
4          8     0        0.166201751                 0
5          9     0        0.250000000                 0
6         10     1        0.880280845                 1
> ## Generate model performance metrics
> rx_forest_metrics <- evaluateModel(data = predDF,
+                                    observed = "churn",
+                                    predicted = "Forest_Prediction")

       1    0
  0   93 5450
  1   84  495
> rx_forest_metrics
 Accuracy Precision    Recall   F-Score
0.9039530 0.4745763 0.1450777 0.2222222
> ## Draw roc curve
> rxrocCurve(data = predDF,
+           observed = "churn",
+           predicted = "Forest_Probability")
> 
```

The ROC Curve for the model is also shown in a separate window.


ROC Curve for 'churn'
True Positive Rate (Sensitivity)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
False Positive Rate (1 - Specificity)
AUC = 0.77

Operationalizing the Model using T-SQL

After the R script or model is ready for production use, the data scientist or a database developer can embed the code or model in system stored procedures and invoke the saved code from an application.

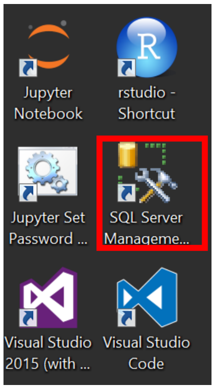
When using R with SQL Server, you can leverage the familiar Transact-SQL interface, and make sure that all computations take place in the database, avoiding unnecessary data movement. When the R code is deployed to production, SQL Server R Services provides the best of the R and SQL worlds.

You can use R for statistical computations that are difficult to implement using SQL, and also leverage the power of SQL Server to achieve maximum performance. For example, you can now leverage the power of the in-memory database engine and columnstore indexes.

In this section, you will use Transact-SQL to generate scores from a predictive model in production, or return plots generated by R and present them in an application such as Reporting Services.

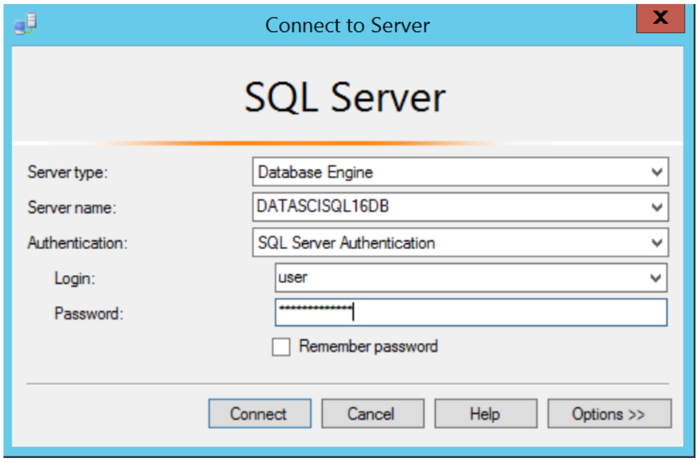
This section assumes that you have already login to the virtual machine.

1. Click on **SQL Server Management Studio**.



2. Connect to SQL Server 2016 with the username and password that you have been provided.
   Username: user<Id>
   Example: user01

   Password: <to be provided>



Server type: Database Engine

Server name: DATASCISQL16DB

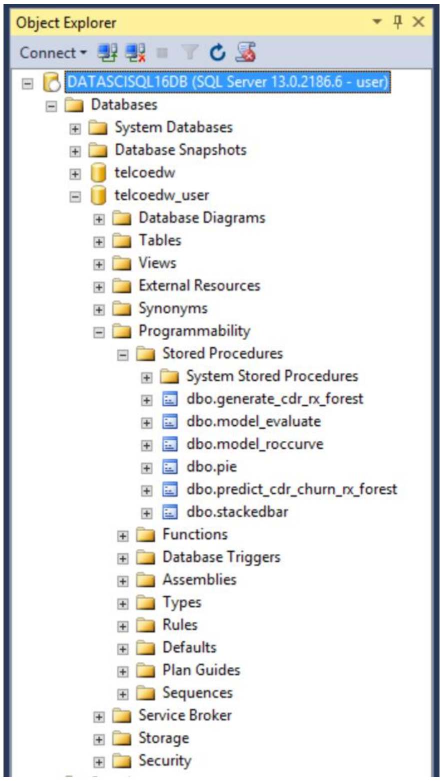
Authentication: SQL Server Authentication

Login: user

Password: *************

□ Remember password

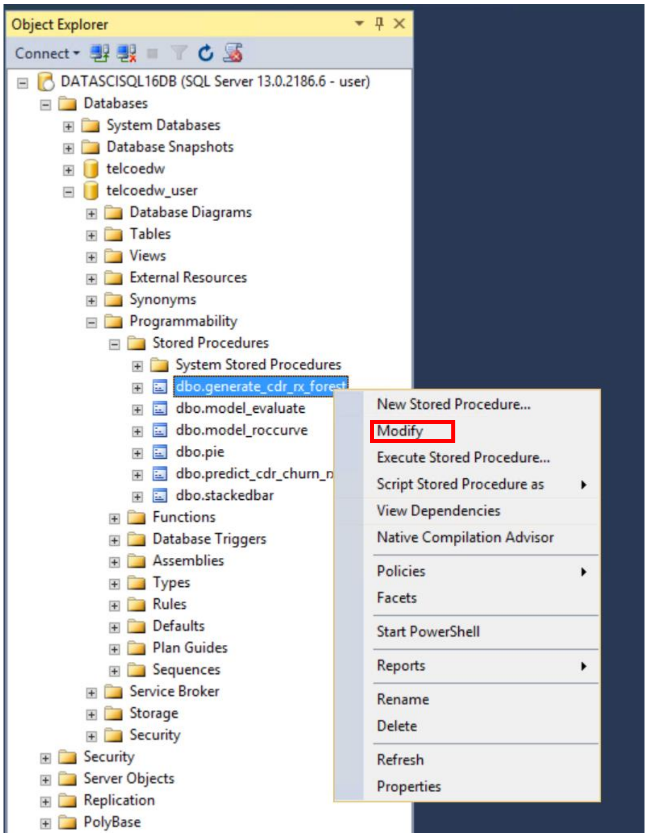
3. Navigate to the Database **telcoedw_user**.
   Expand the node for **Programmability | Stored Procedures**



- DATASCISQL16DB (SQL Server 13.0.2186.6 - user)
  - Databases
    - System Databases
    - Database Snapshots
    - telcoedw
    - telcoedw_user
      - Database Diagrams
      - Tables
      - Views
      - External Resources
      - Synonyms
      - Programmability
        - Stored Procedures
          - System Stored Procedures
          - dbo.generate_cdr_rx_forest
          - dbo.model_evaluate
          - dbo.model_roccurve
          - dbo.pie
          - dbo.predict_cdr_churn_rx_forest
          - dbo.stackedbar
        - Functions
        - Database Triggers
        - Assemblies
        - Types
        - Rules
        - Defaults
        - Plan Guides
        - Sequences
      - Service Broker
      - Storage
      - Security

4. You will see several stored procedures that have been created for training, evaluating and prediction. Each of these intelligent stored procedures harness the power of R to deliver advanced analytics to the application developers.

5. Let’s explore one of the stored procedure (**dbo.generate_cdr_rx_forest**). To do this, right click on the stored procedure, and choose Modify.



6. **Training** - You will see the T-SQL and R code that uses the ScaleR libraries for training a decision forest. We use the r***xDForest()*** function to build a classification model for determining whether the customer will likely to churn.

   We used the ***rxGetVarNames()*** function to get the features that will be used for training. In the R code, the following 2 lines defines the label and features that will be used in training, as follows:

   churn ~ <feature 1> + <feature 2> …

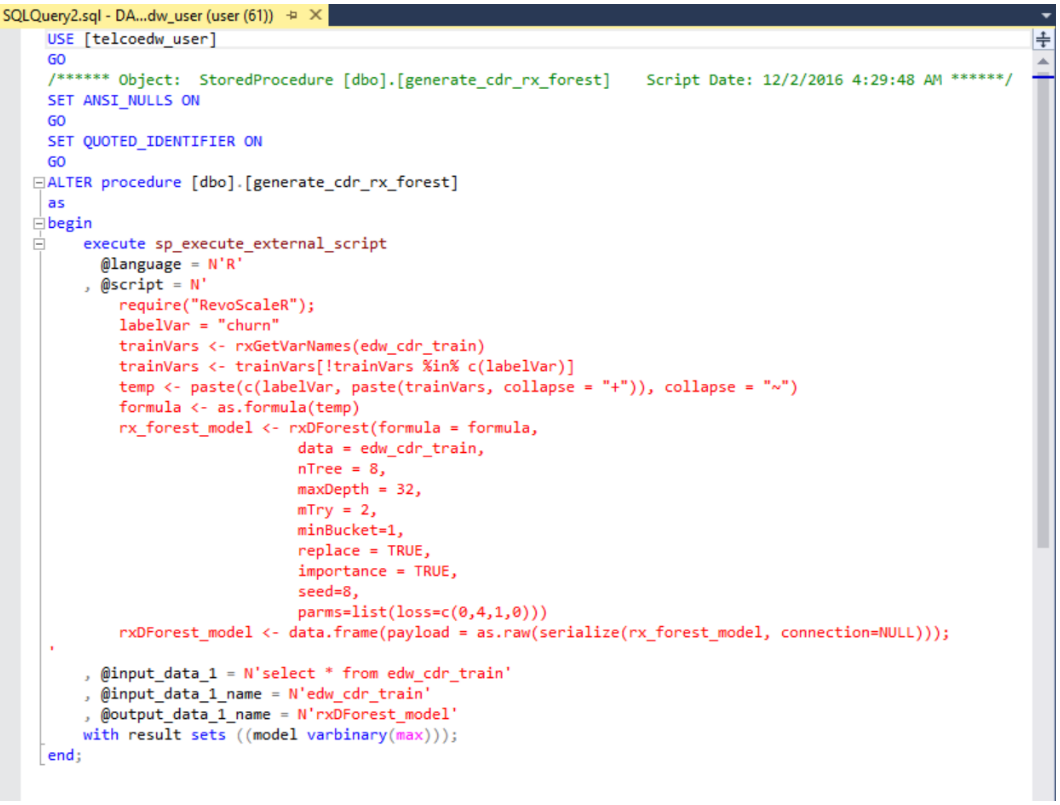
```
temp<-paste(c("churn",paste(train_vars, collapse="+") ),collapse="~")
formula<-as.formula(temp)
```

   The data used for training is stored in the **table edw_cdr_train**.

Note:

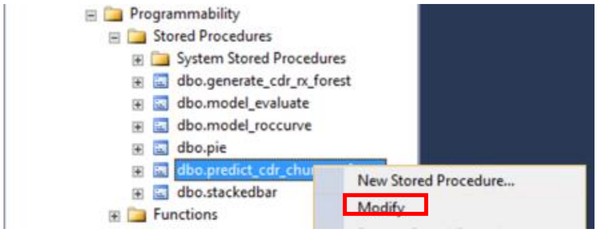
rxDForest() is a parallel external memory decision forest algorithm, that is designed to work with very large data sets, based on the random forest work done by Leo Breiman and Adele Cutler, and the randomForest package of Andy Liaw and Matthew Weiner.

- http://www.rdocumentation.org/packages/RevoScaleR/functions/rxDForest
- http://blog.revolutionanalytics.com/2014/01/a-first-look-at-rxdforest.html

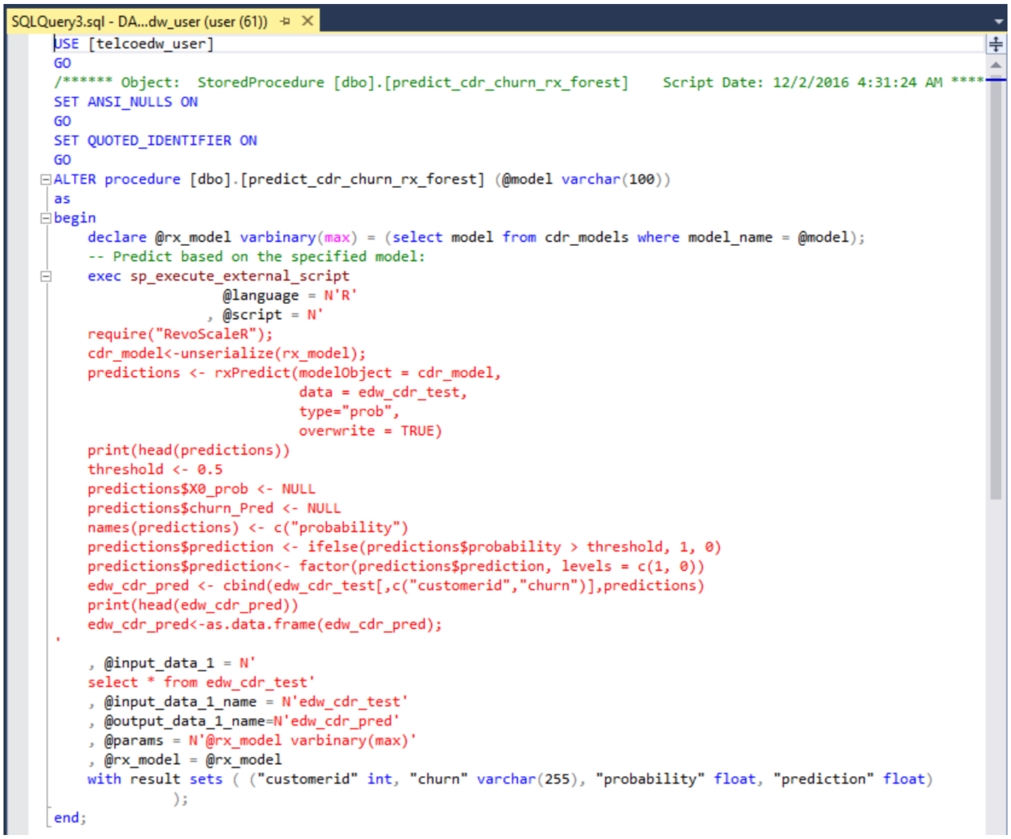


```
USE [telcoedw_user]
GO
/****** Object:  StoredProcedure [dbo].[generate_cdr_rx_forest]    Script Date: 12/2/2016 4:29:48 AM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[generate_cdr_rx_forest]
as
begin
    execute sp_execute_external_script
      @language = N'R'
    , @script = N'
        require("RevoScaleR");
        labelVar = "churn"
        trainVars <- rxGetVarNames(edw_cdr_train)
        trainVars <- trainVars[!trainVars %in% c(labelVar)]
        temp <- paste(c(labelVar, paste(trainVars, collapse = "+")), collapse = "~")
        formula <- as.formula(temp)
        rx_forest_model <- rxDForest(formula = formula,
                                    data = edw_cdr_train,
                                    nTree = 8,
                                    maxDepth = 32,
                                    mTry = 2,
                                    minBucket=1,
                                    replace = TRUE,
                                    importance = TRUE,
                                    seed=8,
                                    parms=list(loss=c(0,4,1,0)))
        rxDForest_model <- data.frame(payload = as.raw(serialize(rx_forest_model, connection=NULL)));
'
    , @input_data_1 = N'select * from edw_cdr_train'
    , @input_data_1_name = N'edw_cdr_train'
    , @output_data_1_name = N'rxDForest_model'
    with result sets ((model varbinary(max)));
end;
```

7. **Using the model for prediction** – Next, let’s take a look at how we can create a stored procedure that is used for performing scoring of the data. To do this, right click on the stored procedure ***predict_cdr_churn_forest***, and choose **Modify**



You will see how the **rxPredict() function** is used to predict the customers that are likely to churn.



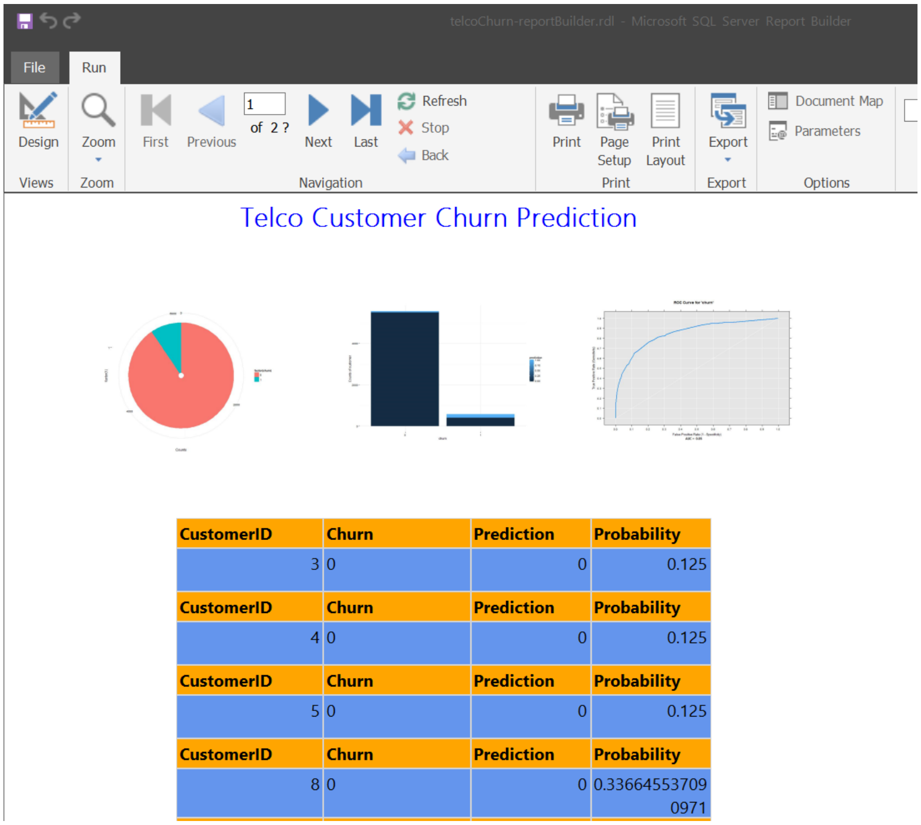
```
USE [telcoedw_user]
GO
/****** Object:  StoredProcedure [dbo].[predict_cdr_churn_rx_forest]    Script Date: 12/2/2016 4:31:24 AM ****
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[predict_cdr_churn_rx_forest] (@model varchar(100))
as
begin
    declare @rx_model varbinary(max) = (select model from cdr_models where model_name = @model);
    -- Predict based on the specified model:
    exec sp_execute_external_script
                    @language = N'R'
                  , @script = N'
    require("RevoScaleR");
    cdr_model<-unserialize(rx_model);
    predictions <- rxPredict(modelObject = cdr_model,
                             data = edw_cdr_test,
                             type="prob",
                             overwrite = TRUE)
    print(head(predictions))
    threshold <- 0.5
    predictions$X0_prob <- NULL
    predictions$churn_Pred <- NULL
    names(predictions) <- c("probability")
    predictions$prediction <- ifelse(predictions$probability > threshold, 1, 0)
    predictions$prediction<- factor(predictions$prediction, levels = c(1, 0))
    edw_cdr_pred <- cbind(edw_cdr_test[,c("customerid","churn")],predictions)
    print(head(edw_cdr_pred))
    edw_cdr_pred<-as.data.frame(edw_cdr_pred);
'

    , @input_data_1 = N'
    select * from edw_cdr_test'
    , @input_data_1_name = N'edw_cdr_test'
    , @output_data_1_name=N'edw_cdr_pred'
    , @params = N'@rx_model varbinary(max)'
    , @rx_model = @rx_model
    with result sets ( ("customerid" int, "churn" varchar(255), "probability" float, "prediction" float)
              );
end;
```

8. Putting it together – Let us put all these together, and see how we can leverage the stored procedures in an intelligent application.

```
--Set DB
use telcoedw_user
go

-- Show the serialized model
select * from cdr_models

----------------------------------------------------------------------------------
-- rxDForest
----------------------------------------------------------------------------------
-- Step 1 - Train the customer churn model
-- After successful execution, this will create a binary representation of the
model
exec generate_cdr_rx_forest;

-- Step 2 - Evaluate the model
-- This uses test data to evaluate the performance of the model.
```

```
exec model_evaluate

-- Step 3 - Score the model- In this step, you will invoke the stored procedure
predict_cdr_churn_forest
-- The stored procedure uses the rxPredict function to predict the customers that
are likely to churn
-- Results are returned as an output dataset
-- Execute scoring procedure
exec predict_cdr_churn_rx_forest 'rxDForest';
go
```

9. Open and execute the telcoChurn-reportBuilder.rdl from SQL Server Report Builder, you will see the below intelligent report for telco customer churn.



Telco Customer Churn Prediction

CustomerID	Churn	Prediction	Probability
3	0	0	0.125
CustomerID	Churn	Prediction	Probability
4	0	0	0.125
CustomerID	Churn	Prediction	Probability
5	0	0	0.125
CustomerID	Churn	Prediction	Probability
8	0	0	0.336645537090971