Implementing Predictive Analytics with Hadoop in Azure HDInsight

Lab 4 – Using R Server with Spark

Overview

In this lab, you will use R Server with Spark

What You'll Need

To complete the labs, you need the following:

- A web browser
- A Microsoft account
- A Microsoft Azure subscription
- A Windows, Linux, or Mac OS X computer
- An SSH client. For Linux or Max OS X use the built in SSH client. For windows download PuTTY.

Note: To set up the required environment for the lab, follow the instructions in the Setup document for this course. Specifically, you must have signed up for an Azure subscription.

Provisioning R Server on HDInsight

Note: The Microsoft Azure portal is continually improved in response to customer feedback. The steps in this exercise reflect the user interface of the Microsoft Azure portal at the time of writing, but may not match the latest design of the portal exactly.

Provision an R Server HDInsight Cluster

1. In a web browser, navigate to http://portal.azure.com, and if prompted, sign in using the Microsoft account that is associated with your Azure subscription.
2. In the Microsoft Azure portal, in the Hub Menu, click **New**. Then in the **Intelligence + Analytics** menu, click **HDInsight**.
3. In the **New HDInsight Cluster** blade, enter the following settings, and then click **Create**:
    - **Cluster Name**: *Enter a unique name (and make a note of it!)*
    - **Subscription**: *Select your Azure subscription*
    - **Select Cluster Type:**
        - **Cluster Type:** R Server on Spark
        - **Operating System:** Linux
        - **Version:** *Choose the latest available version of Spark*
        - **Cluster Tier:** Premium
    - **Applications**: R Server for HDInsight
    - **Credentials:**

        - **Cluster Login Username**: *Enter a user name of your choice (and make a note of it!)*
        - **Cluster Login Password**: *Enter and confirm a strong password (and make a note of it!)*
        - **SSH Username:** *Enter another user name of your choice (and make a note of it!)*
        - **SSH Authentication Type**: Password
        - **SSH Password:** *Enter and confirm a strong password (and make a note of it!)*
    - **Data Source:**
        - **Create a new storage account**: *Enter a unique name consisting of lower-case letters and numbers only (and make a note of it!)*
        - **Choose Default Container**: *Enter the cluster name you specified previously*
        - **Location**: *Select any available region*

          Note: Selecting the location for the data source will also set the location of the HDInsight Cluster. The storage account must be in the same region as the HDInsight Cluster
    - **Pricing:**
        - **Number of Worker nodes**: 1
        - **Worker Node Size**: *Use the default selection*
        - **Head Node Size**: *Use the default selection*
    - **Optional Configuration:** *None*
    - **Resource Group**: *Create a new resource group with a unique name*
    - **Pin to dashboard**: *Not selected*
4. In the Azure portal, view **Notifications** to verify that deployment has started. Then wait for the cluster to be deployed (this can take a long time – often 30 minutes or more. Put your feet up and do a crossword puzzle or something!)

Note: As soon as an HDInsight cluster is running, the credit in your Azure subscription will start to be charged. The free-trial subscription includes a credit limit of approximately $200 (or local equivalent) that you can spend over a period of 30 days, which is enough to complete the labs in this course as long as clusters are deleted when not in use. If you decide not to complete this lab, follow the instructions in the *Clean Up* procedure at the end of the lab to delete your cluster in order to avoid using your Azure credit unnecessarily.

Connect to Your Cluster

After your cluster has been provisioned, you can connect to it using a secure shell session.

1. Use an SSH client (Putty on Windows or the native client on a Mac or Linux) to connect to the cluster. The endpoint to SSH into your cluster will be:

   ***cluster_name*-ssh.azurehdinsight.net**.

2. Log in using the SSH User credentials you specified when creating the cluster (***not*** the HTTP user credentials)
3. At the bash prompt you can enter **R** to start the R Console. This provides a command-line interface for writing R code.
4. To quit the R Console type **q()**, and enter **n** when prompted to save the workspace image.
5. Keep the SSH session open.

Install RStudio

While the R Console can be useful for running short R commands, RStudio provides a more fully-featured development environment for R code.

1. In the SSH session command line, enter the following command to set the user context to a more privileged user:

   ```
   sudo su -
   ```

2. Enter the following command to download a script to install RStudio:

   ```
   wget http://mrsactionscripts.blob.core.windows.net/rstudio-server-community-v01/InstallRStudio.sh
   ```

3. Enter the following command to modify the permissions on the script:

   ```
   chmod 755 InstallRStudio.sh
   ```

4. Enter the following command to run the script.

   ```
   ./InstallRStudio.sh
   ```

5. When the script has finished, close the SSH session window.

Configure an SSH Tunnel

In order to connect to RStudio on your HDInsight Cluster you must create an SSH Tunnel.

If you are using Windows

1. Open PuTTY
2. Enter the cluster connection information as before, and then under the **Connection** node on the left hand side, expand **SSH** and select **Tunnels**. Enter the following information:
   - **Source Port:** 8787
   - **Destination:** localhost:8787
   - Click **Add**
3. Click **Open** to open a new SSH session, and log in using your SSH credentials

If you are using Linux or Mac OS X

1. Open a terminal session.
2. At the prompt enter the following command, replacing *usr* with your SSH user name and replacing *clust* with your cluster name.

   ```
   ssh -L localhost:8787:localhost:8787 usr@r-server.clust-ssh.azurehdinsight.net
   ```

Open RStudio

Now that you have configured a tunnel, you can open RStudio on your cluster head node.

1. In a web browser, navigate to **http://localhost:8787**.
2. Enter your SSH username and password in the login box to login to RStudio.
3. Keep the RStudio browser session open, you will return to it later.

Using R to Perform Linear Regression

Now that you have a working R Server environment, you can use it to perform linear regression on some data.

Upload the Source Data

You will create a liner regression model for Iris identification, based on a commonly used multiclass dataset. Before you can create the model, you must upload the data file to the shared storage for

your Spark cluster. You can use any Azure blob storage tool to do this – these instructions assume you are using the Azure command line interface.

1. Open a new command line window.
2. Enter the following command to switch the Azure CLI to resource manager mode.

   ```
   azure config mode arm
   ```

   Note: If a command not found error is displayed, ensure that you have followed the instructions in the setup guide to install the Azure CLI.

3. Enter the following command to log into your Azure subscription:

   ```
   azure login
   ```

4. Follow the instructions that are displayed to browse to the Azure device login site and enter the authentication code provided. Then sign into your Azure subscription using your Microsoft account.
5. Enter the following command to view your Azure resources:

   ```
   azure resource list
   ```

6. Verify that your HDInsight cluster and the related storage account are both listed. Note that the information provided includes the resource group name as well as the individual resource names.
7. Note the resource group and storage account name.
8. Enter the following command on a single line to determine the connection string for your Azure storage account, specifying the storage account and resource group names you noted earlier:

   ```
   azure storage account connectionstring show account -g resource_group
   ```

9. Note the connection string, copying it to the clipboard if your command line tool supports it.
10. If you are working on a Windows client computer, enter the following command to set a system variable for the connection string:

    ```
    SET AZURE_STORAGE_CONNECTION_STRING=your_connection_string
    ```

    If you are using a Linux or Mac OS X client computer, enter the following command to set a system variable for the connection string (enter the connection string in quotation marks):

    ```
    export AZURE_STORAGE_CONNECTION_STRING="your_connection_string"
    ```

11. Enter a `dir` (Windows) or `ls` (Mac OS X or Linux) command to view the contents of the **Lab04** folder where you extracted the lab files for this course (for example, `dir c:\DAT202.3x\Lab04` or `ls DAT202.3x/Lab04`), and verify that this folder contains a file named **iris-multiclass.csv**. This file contains observations of different flowers, their attributes and types.
12. Enter the following command on a single line to upload the iris-multiclass.csv file to a blob in the container used by your HDInsight cluster. Replace ***file*** with the local path to iris-multiclass.csv (for example *c:\DAT202.3x\Lab04\iris-multiclass.csv* or *DAT202.3x/Lab04/iris-multiclass.csv*) and replace ***container*** with the name of the storage container used by your cluster (which should be the same as the cluster name):

    ```
    azure storage blob upload file container iris/iris-multiclass.csv
    ```

13. Wait for the file to be uploaded.

Load the Data into an R Data Frame

1. In the RStudio browser window, on the **File** menu, point to **New File**, and click **R Script** to create a new R Script file.
2. In the new R Script file, enter the following code (which you can copy and paste from **iris-regression.txt** in the lab files folder for this lab).

```
# Set context to Spark
mySparkCluster <- RxSpark(consoleOutput=TRUE)
rxSetComputeContext(mySparkCluster)

# Get the data
hdfsFS <- RxHdfsFileSystem()
bigDataDirRoot <- "/iris"
inputDir <- file.path(bigDataDirRoot,"iris-multiclass.csv")
irisDataColInfo <- list(
  list(index = 1, type = "numeric", newName = "sepallength"),
  list(index = 2, type = "numeric", newName = "sepalwidth"),
  list(index = 3, type = "numeric", newName = "petallength"),
  list(index = 4, type = "numeric", newName = "petalwidth"),
  list(index = 5, type = "factor", newName = "class"))

irisDS <- RxTextData(file = inputDir, missingValueString = "M",
colInfo = irisDataColInfo,
                     fileSystem = hdfsFS)
irisData <- rxImport(inData = irisDS)
rxGetInfo(irisDS, getVarInfo=TRUE)
```

3. In the script pane, select the code you pasted and then on the **Code** menu, click **Run Selected Line(s)** to run it.
4. Review the output in the **Console** pane, noting that the data has been loaded, and the variables it contains identified.

Perform a Simple Linear Regression

1. In the RStudio browser window, in the script pane, add the following code (which you can copy and paste from **iris-regression.txt**) to your script:

```
# Perform simple linear regression
linRegSepalPetallength <- rxLinMod(formula = sepallength ~
petallength, data = irisDS)
linRegSepalPetallength
```

2. In the script pane, select the code you added and then on the **Code** menu, click **Run Selected Line(s)** to run it.
3. Review the output in the **Console** pane, noting the intercept and coefficient for the regression that has been calculated. In this case, the regression equation is:

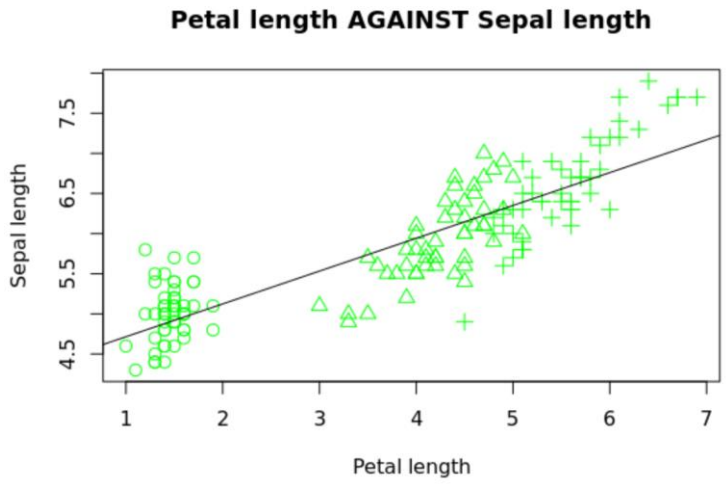
   sepallength = 4.3055655 + (0.4091259 x *petallength*)

4. In the RStudio browser window, in the script pane, add the following code (which you can copy and paste from **iris-regression.txt**) to your script:

```
# Plot regression
plot( irisData$petallength, irisData$sepallength,
      pch = as.integer(irisData$class), cex = 1.3,
```

```
      col = "green",
      main = "Petal length AGAINST Sepal length",
      xlab = "Petal length", ylab = "Sepal length")
abline(linRegSepalPetallength)
```

5. In the script pane, select the code you added and then on the **Code** menu, click **Run Selected Line(s)** to run it.
6. Review the output in the **Plots** pane, which shows the plotted *sepallength* and *petallength* values and the regression line that has been calculated:



Perform Multiple Variable Regression

1. In the RStudio browser window, in the script pane, add the following code (which you can copy and paste from **iris-regression.txt**) to your script:

```
# Perform multiple variable regression
f <- sepallength ~ sepalwidth + petallength + petalwidth + class
mrAllVar <- rxLinMod(f , data = irisDS)
summary(mrAllVar)
```

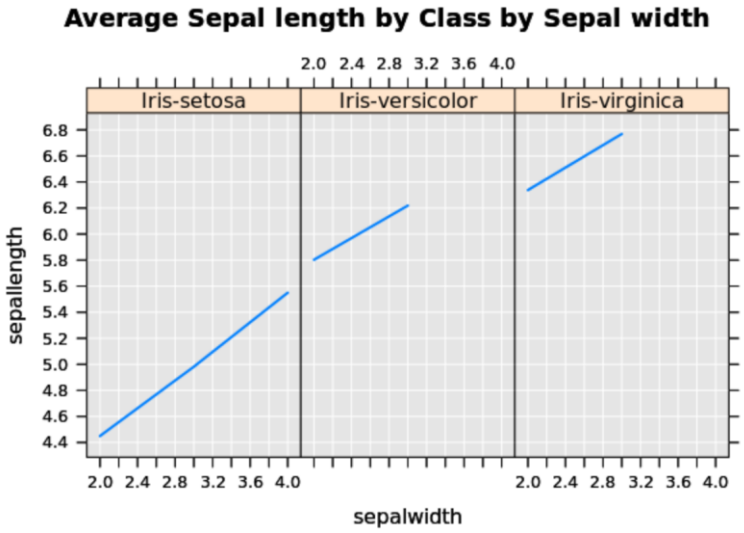
2. In the script pane, select the code you added and then on the **Code** menu, click **Run Selected Line(s)** to run it.
3. Review the output in the **Console** pane, noting the statistics that have been calculated.
4. In the RStudio browser window, in the script pane, add the following code (which you can copy and paste from **iris-regression.txt**) to your script:

```
# Test the dependency of sepal length on class and sepal width
mrSepallenghtSepalwidthClass <- rxLinMod(sepallength ~
class:F(sepalwidth), data = irisDS, cube = TRUE)
mrCount <- rxResultsDF(mrSepallenghtSepalwidthClass, type = "counts")
head(mrCount,30)
mrCount <- na.omit(mrCount)
```

5. Review the output in the **Console** pane.
6. In the RStudio browser window, in the script pane, add the following code (which you can copy and paste from **iris-regression.txt**) to your script:

```
# Plot the relationship of sepal length, class, and sepal width
rxLinePlot( sepallength~sepalwidth|class, data = mrCount,
            title = "Average Sepal length by Class by Sepal width")
```

7. Review the output in the **Plots** pane, which shows the plotted *sepallength* by *sepalwidth* and *class*:



Using R to Perform Clustering

Clustering is a common unsupervised machine learning technique that is used to determine groups, or *clusters*, of similar data points within a dataset. In this exercise, you'll use R to perform K-Means clustering.

Create a New Script

1. In the RStudio browser window, on the **File** menu, point to **New File**, and click **R Script** to create a new R Script file.
2. In the new R Script file, enter the following code (which you can copy and paste from **iris-clustering.txt** in the lab files folder for this lab).

```
# Set context to Spark
mySparkCluster <- RxSpark(consoleOutput=TRUE)
rxSetComputeContext(mySparkCluster)

# Get the data
hdfsFS <- RxHdfsFileSystem()
bigDataDirRoot <- "/iris"
inputDir <- file.path(bigDataDirRoot,"iris-multiclass.csv")
irisDataColInfo <- list(
  list(index = 1, type = "numeric", newName = "sepallength"),
  list(index = 2, type = "numeric", newName = "sepalwidth"),
  list(index = 3, type = "numeric", newName = "petallength"),
  list(index = 4, type = "numeric", newName = "petalwidth"),
  list(index = 5, type = "factor", newName = "class"))

irisDS <- RxTextData(file = inputDir, missingValueString = "M",
colInfo = irisDataColInfo,
                     fileSystem = hdfsFS)
irisData <- rxImport(inData = irisDS)
rxGetInfo(irisDS, getVarInfo=TRUE)
```

3. In the script pane, select the code you pasted and then on the **Code** menu, click **Run Selected Line(s)** to run it.

4. Review the output in the **Console** pane, noting that the data has been loaded, and the variables it contains identified (this is the same data you loaded for regression).

Run K-Means Clustering

1. In the RStudio browser window, in the script pane, add the following code (which you can copy and paste from **iris-clustering.txt**) to your script:

```
#Run K-means
irisCluster <- rxKmeans(~ sepallength + sepalwidth + petallength,
                        data = irisData, numClusters = 3, seed = 30)
irisCluster
```

2. In the script pane, select the code you added and then on the **Code** menu, click **Run Selected Line(s)** to run it.
3. Review the output in the **Console** pane, noting the cluster details.

Visualize the Clusters

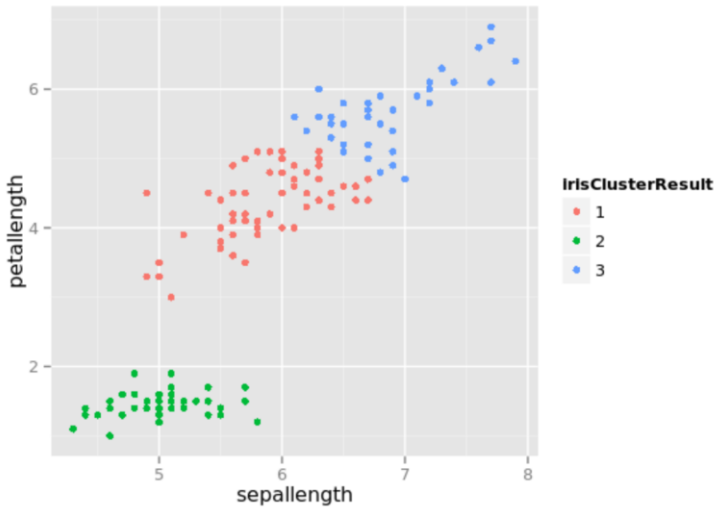
1. In the Console pane, enter the following command to install the **ggplot2** library (this can take a few minutes):

```
install.packages("ggplot2")
```

2. In the RStudio browser window, in the script pane, add the following code (which you can copy and paste from **iris-clustering.txt**) to your script:

```
# Visualize the clusters
library(ggplot2)
irisClusterResult<- as.factor(irisCluster$cluster)
ggplot(irisData, aes(sepallength, petallength,
                     color = irisClusterResult)) + geom_point()
```

3. In the script pane, select the code you added and then on the **Code** menu, click **Run Selected Line(s)** to run it.
4. Review the output in the **Plots** pane, which should show the clusters identified using K-Means:



5. Close the RStudio pane in your browser.

Clean Up

Now that you have finished the lab, you should delete your cluster and the associated storage account. This ensures that you avoid being charged for cluster resources when you are not using them. If you are using a trial Azure subscription that includes a limited free credit value, deleting the cluster maximizes your credit and helps to prevent using it all before the free trial period has ended.

Delete the Resource Group for your HDInsight Cluster

1. If it is not already open in a tab in your web browser, browse to the new Azure portal at https://portal.azure.com.
2. In the Azure portal, view your **Resource groups** and select the resource group you created for your cluster. This resource group contains your cluster and the associated storage account.
3. In the blade for your resource group, click **Delete**. When prompted to confirm the deletion, enter the resource group name and click **Delete**.
4. Wait for a notification that your resource group has been deleted.