Machine Learning using Spark:

Module 5, Lesson 4  
Build a Pipeline Machine Learning application Hands-On Lab

## Overview

In this lab, students will learn how to build a machine learning algorithm on a Spark cluster in HDInsight. Students will use the Jupyter notebook to build and test the application. The application uses the same *HVAC.csv* data that we looked at in the previous lab.

## Objectives

In this hands-on lab you will learn how to:

* Setup data to input into a pipeline of stages that will result in a new model
* Setup the pipeline and execute it
* Setup test data and run the new model against the test data

## Prerequisites

The following are required to complete this hands-on lab:

* A web browser
* A Spark cluster in Azure HDInsight
* WinsCP is helpful to download the CSV file but not necessary.

Note: The Azure portal is continually improved and changed. 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 portal.

## Exercises

This hands-on lab includes the following exercises:

* Exercise 1: Review the data
* Exercise 2: Write Machine learning application

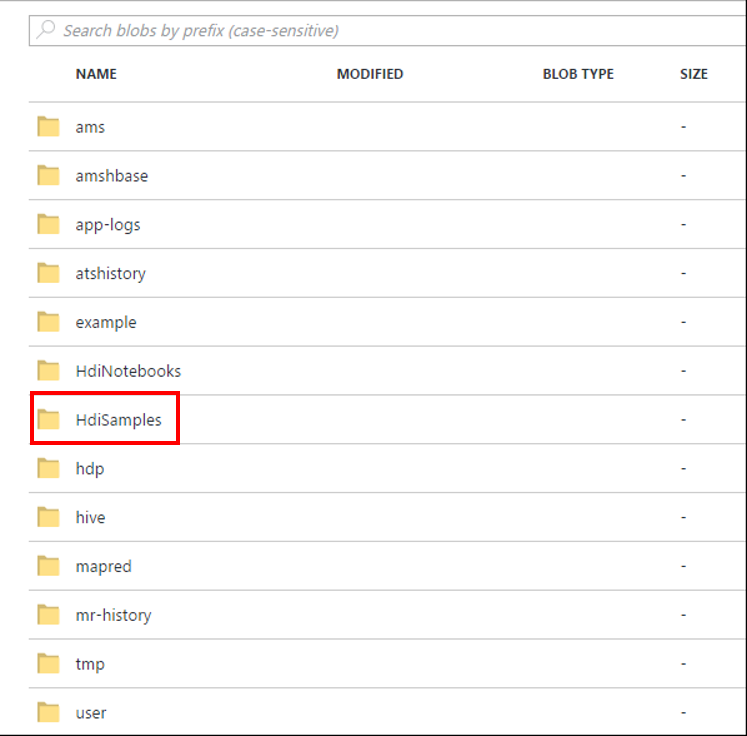
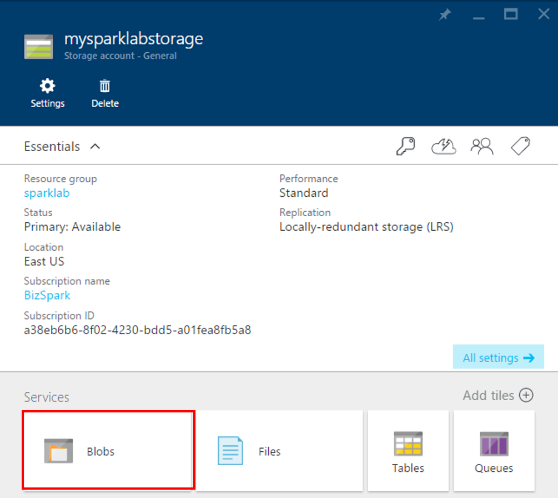
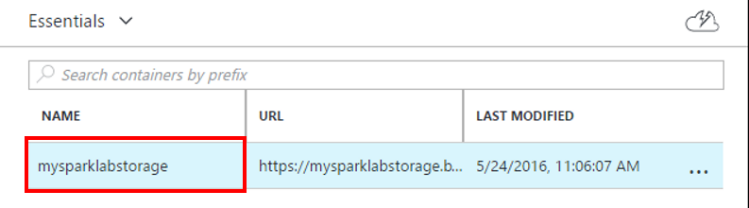
## Exercise 1: Review the Data

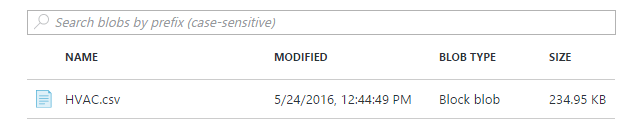
This exercise will cover how to download the data file from /HdiSamples/HdiSamples/SensorSampleData/hvac and how to open the CSV file to review the data.

1. Download the datafile

From the Azure Portal, open the Spark storage resource that was created in the previous lab – Creating a Spark cluster. In that example, we used *mysparklabstorage* as the name for the storage. Students can also see all resources associated with the Spark cluster created in the previous lab by selecting the name of the resource group (in our example, *sparklab* was used). This resource group will have the HDInsight cluster resource as well as the storage resource created.

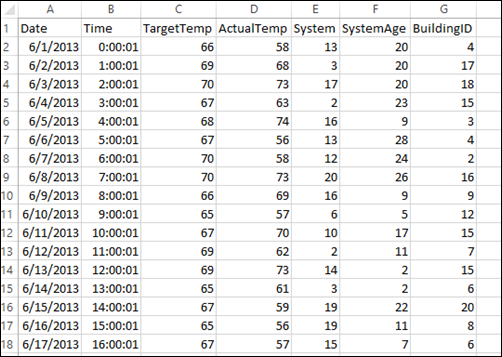
Select the storage resource created last time, and then click on **Blobs** under Services to access the container. In our example, we used *mysparkstorage* as the name of the container. Click on the container to see a list of files and folders in the HDInsight cluster. From there, find and select the HdiSamples folder and drill down (HdiSamples > HdiSamples > SensorSampleData > hvac > HVAC.csv) to HVAC.csv. Select the file and click **Download** to copy to the students Windows computer.





1. Examine the CSV data file

Open the CSV file to review the data file.

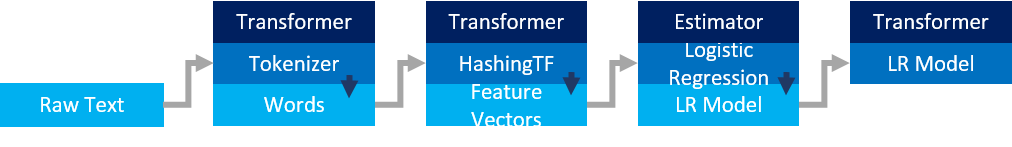


The data shows the target temperature and actual temperature of buildings (differentiated with BuildingID) with HVAC systems installed. Assume that the System column provides a SystemID for each of the HVAC systems and the SystemAge is how old the HVAC system is.

Students will use this data to predict whether a building will be hotter or colder based on the target temperature, given a SystemID and system age.

## Exercise 2: Write Machine learning application

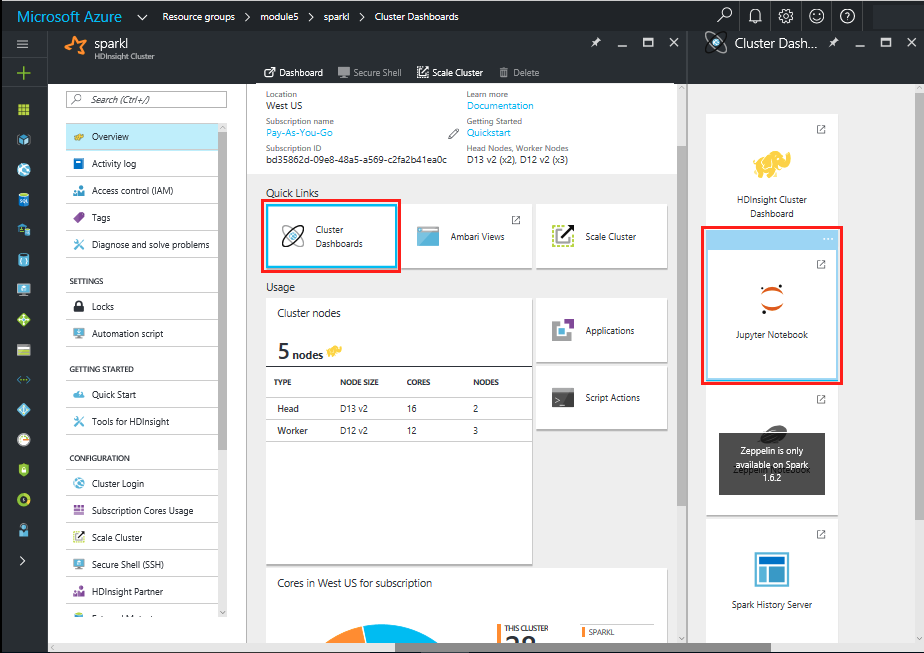
In this Spark machine learning application, we will use the Spark ML pipeline to perform document classification. In each pipeline stage, the data document is split into words, words are converted to a numerical feature vector, and finally a prediction model is built using the feature vectors and labels.



1. After provisioning an Spark Cluster ([Module 5 Lesson 2 Lab](https://github.com/MSFTImagine/computerscience/tree/master/Complimentary%20Course%20Content/Module5/Labs)) Create a Jupyter notebook with PySpark kernel

From the Azure Portal, navigate to the Spark cluster that the student just created. It can be found under **Browse All>HDInsight Clusters**.

From QuickLinks, select **Cluster Dashboards** and then select **Jupyter Notebook**. If prompted enter the admin credentials for the cluster. This will open up a browser window loaded with Jupyter notebook



Create **New** and select a PySpark kernel.

You may also reach the Jupyter Notebook for your cluster by opening the following URL in your browser. https://CLUSTERNAME.azurehdinsight.net/jupyter (Replace CLUSTERNAME with the name of student cluster)

1. Name the PySpark kernel with a user friendly name



1. Import required datatypes

The PySpark kernel includes the context for Spark and Hive. The student does not have to create these contexts explicitly.

To begin the exercise, import the types required for this scenario. To do so, paste the following code snippet in a cell and press SHIFT + ENTER. Pressing SHIFT + ENTER executes the entry in the current cell and moves the cursor to the next cell.

from pyspark.ml import Pipeline

from pyspark.ml.classification import LogisticRegression

from pyspark.ml.feature import HashingTF, Tokenizer

from pyspark.sql import Row

import os

import sys

from pyspark.sql.types import \*

from pyspark.mllib.classification import LogisticRegressionWithSGD

from pyspark.mllib.regression import LabeledPoint

from numpy import array

1. Define a parsing function and load data

The application must load the data, parse it, and use it to train the model. To do this, a new function must be defined that checks whether the actual temperature of the building is greater or lower than the target temperature. If it is greater, the building is hot, denoted with a value of 1.0 and if less, the building is cold, denoted with 0.0.

In an empty cell, paste the following code example and press SHIFT + ENTER.

# List the structure of data for better understanding. Becuase the data will be

# loaded as an array, this structure makes it easy to understand what each element

# in the array corresponds to

# 0 Date

# 1 Time

# 2 TargetTemp

# 3 ActualTemp

# 4 System

# 5 SystemAge

# 6 BuildingID

LabeledDocument = Row("BuildingID", "SystemInfo", "label")

# Define a function that parses the raw CSV file and returns an object of type LabeledDocument

def parseDocument(line):

values = [str(x) for x in line.split(',')]

if (values[3] > values[2]):

hot = 1.0

else:

hot = 0.0

textValue = str(values[4]) + " " + str(values[5])

return LabeledDocument((values[6]), textValue, hot)

# Load the raw HVAC.csv file, parse it using the function

data = sc.textFile("wasb:///HdiSamples/HdiSamples/SensorSampleData/hvac/HVAC.csv")

documents = data.filter(lambda s: "Date" not in s).map(parseDocument)

training = documents.toDF()

1. Configure the Spark machine learning pipeline

This Spark machine learning pipeline consists of three stages. The tokenizer Transformer stage takes in raw text and converts them to words DataFrame. The hashingTF Transformer stage takes those words and creates a feature vector DataFrame. Finally, the logistic regression Estimator takes in the feature vectors and fits them to create a new model – which is a Transformer.

In an empty cell, paste the following code example and press SHIFT + ENTER.

tokenizer = Tokenizer(inputCol="SystemInfo", outputCol="words")

hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")

lr = LogisticRegression(maxIter=10, regParam=0.01)

pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])

1. Fit the pipeline to the training document

Call the pipeline fit method to begin fitting the training document over the pipeline that was set up in the previous step.

In an empty cell, paste the following code example and press SHIFT + ENTER.

model = pipeline.fit(training)

1. Verify training document

Verify the training document to checkpoint the application progress.

In an empty cell, paste the following code example and press SHIFT + ENTER.

training.show()

This should give output similar to the following:

+----------+----------+-----+

|BuildingID|SystemInfo|label|

+----------+----------+-----+

| 4| 13 20| 0.0|

| 17| 3 20| 0.0|

| 18| 17 20| 1.0|

| 15| 2 23| 0.0|

| 3| 16 9| 1.0|

| 4| 13 28| 0.0|

| 2| 12 24| 0.0|

| 16| 20 26| 1.0|

| 9| 16 9| 1.0|

| 12| 6 5| 0.0|

| 15| 10 17| 1.0|

| 7| 2 11| 0.0|

| 15| 14 2| 1.0|

| 6| 3 2| 0.0|

| 20| 19 22| 0.0|

| 8| 19 11| 0.0|

| 6| 15 7| 0.0|

| 13| 12 5| 0.0|

| 4| 8 22| 0.0|

| 7| 17 5| 0.0|

+----------+----------+-----+

Compare this output against the raw CSV file that was downloaded in an earlier step.



The first row item is BuildingID 4. The ActualTemp (58) is lower than the TargetTemp (66) suggesting that the building is cold. Therefore, in our training output, the value for the label should be 0.0.

1. Prepare data set to run against trained model

The model accepts a System ID and system age as input – denoted SystemInfo in the training output. The model than predicts whether the building with that system ID and system age would be hotter (1.0) or cooler(0.0) than the target temperature.

In an empty cell, paste the following code example and press SHIFT + ENTER.

# SystemInfo here is a combination of system ID followed by system age

Document = Row("id", "SystemInfo")

test = sc.parallelize([(1L, "20 25"),

(2L, "4 15"),

(3L, "16 9"),

(4L, "9 22"),

(5L, "17 10"),

(6L, "7 22")]) \

.map(lambda x: Document(\*x)).toDF()

1. Make a prediction on test data

*model* is the new Transformer that resulted from executing the pipeline. We will now use the transform method of the newly created predictor model against the test data.

In an empty cell, paste the following code example and press SHIFT + ENTER.

# Make predictions on test documents and print columns of interest

prediction = model.transform(test)

selected = prediction.select("SystemInfo", "prediction", "probability")

for row in selected.collect():

print row

1. You should see output similar to the following:

Row(SystemInfo=u'20 25', prediction=1.0, probability=DenseVector([0.4999, 0.5001]))

Row(SystemInfo=u'4 15', prediction=0.0, probability=DenseVector([0.5016, 0.4984]))

Row(SystemInfo=u'16 9', prediction=1.0, probability=DenseVector([0.4785, 0.5215]))

Row(SystemInfo=u'9 22', prediction=1.0, probability=DenseVector([0.4549, 0.5451]))

Row(SystemInfo=u'17 10', prediction=1.0, probability=DenseVector([0.4925, 0.5075]))

Row(SystemInfo=u'7 22', prediction=0.0, probability=DenseVector([0.5015, 0.4985]))

1. Shutdown the Jupyter notebook to release any resources.

Click the **File** menu on the notebook and **Close and Halt.**

1. Delete the Spark Cluster

HDInsight clusters are billed on a per minute basis. In order to avoid costly charges, delete your cluster after you have finished using it.

## Summary

In this hands-on lab, you learned how to:

* How to create a function to parse text input data
* How to read a data file, parse it, and save it as a DataFrames
* How to setup a pipeline so that the stages are performed in sequence
* How to execute a pipeline to output a predictor model