Open Source Analytics

NBA Real Time Prediction using DSX

Dustin VanStee
@dustinvanstee

This document is written so that the user can reproduce the NBA real time machine learning demo. The demo consists of 2 parts:

1. Data Analysis & Modelling: We provide two approaches here:
    - IBM Data Science Experience(DSX) with RStudio
    - IBM Data Scientist Workbench(DSWB) using Zeppelin Notebooks with Scala
2. Web application desired for user interaction on Bluemix

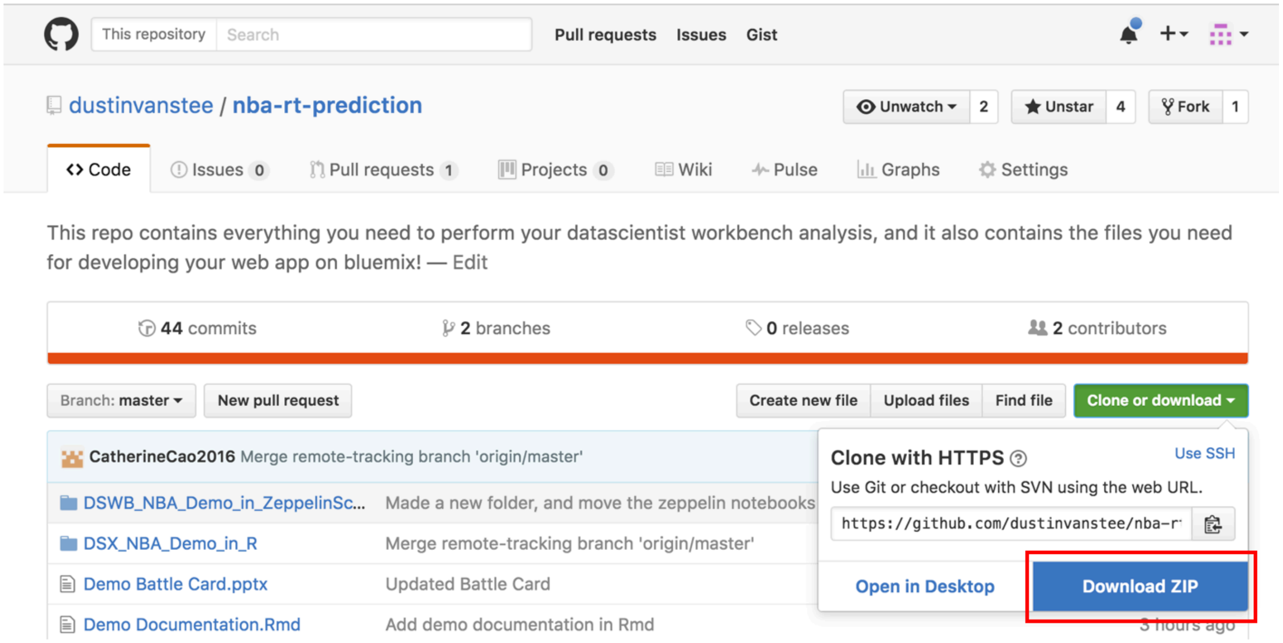
After the environment is setup, there will be a brief description of the Logistic Regression model that was built, and how it was ‘wired’ into the Bluemix environment. Upon completion, you will have a full demo environment setup, with a live website.

1 Setup the DSX Environment

The following steps will guide you through the process of setting up an environment to replicate the NBA RStudio project within Data Science Experience. At the end of this section, you will be able to conduct data analysis and data modelling in RStudio integrated within DSX.

The DSX/Rstudio environment will make use of 2 R markdown files. The next steps document how you can upload these files into DSX/RStudio and create your own HTML data science webpages.

1. Download the project from GitHub.



This repo contains everything you need to perform your datascientist workbench analysis, and it also contains the files you need for developing your web app on bluemix! — Edit

CatherineCao2016 Merge remote-tracking branch 'origin/master'

DSWB_NBA_Demo_in_ZeppelinSc...	Made a new folder, and move the zeppelin notebooks
DSX_NBA_Demo_in_R	Merge remote-tracking branch 'origin/master'
Demo Battle Card.pptx	Updated Battle Card
Demo Documentation.Rmd	Add demo documentation in Rmd

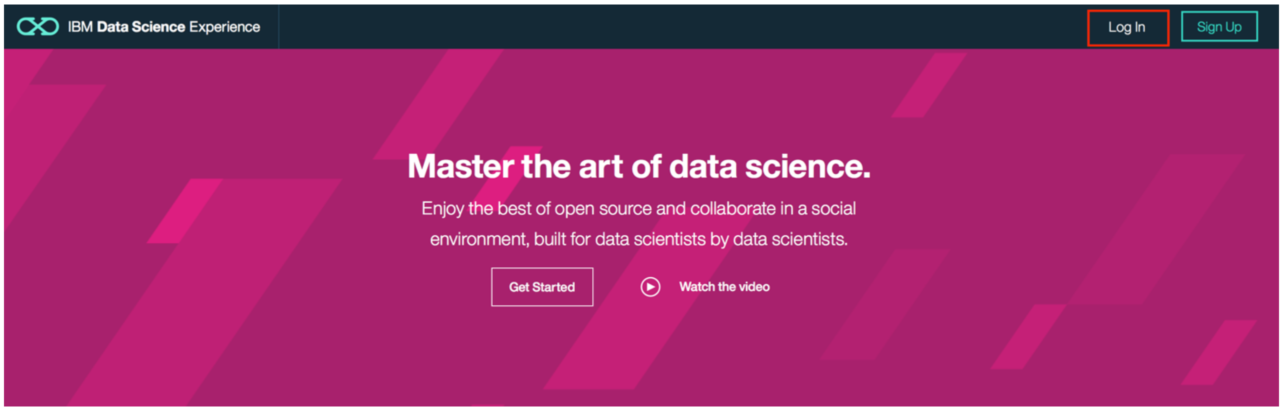
Clone with HTTPS

Use Git or checkout with SVN using the web URL.

```
https://github.com/dustinvanstee/nba-r
```

3 hours ago

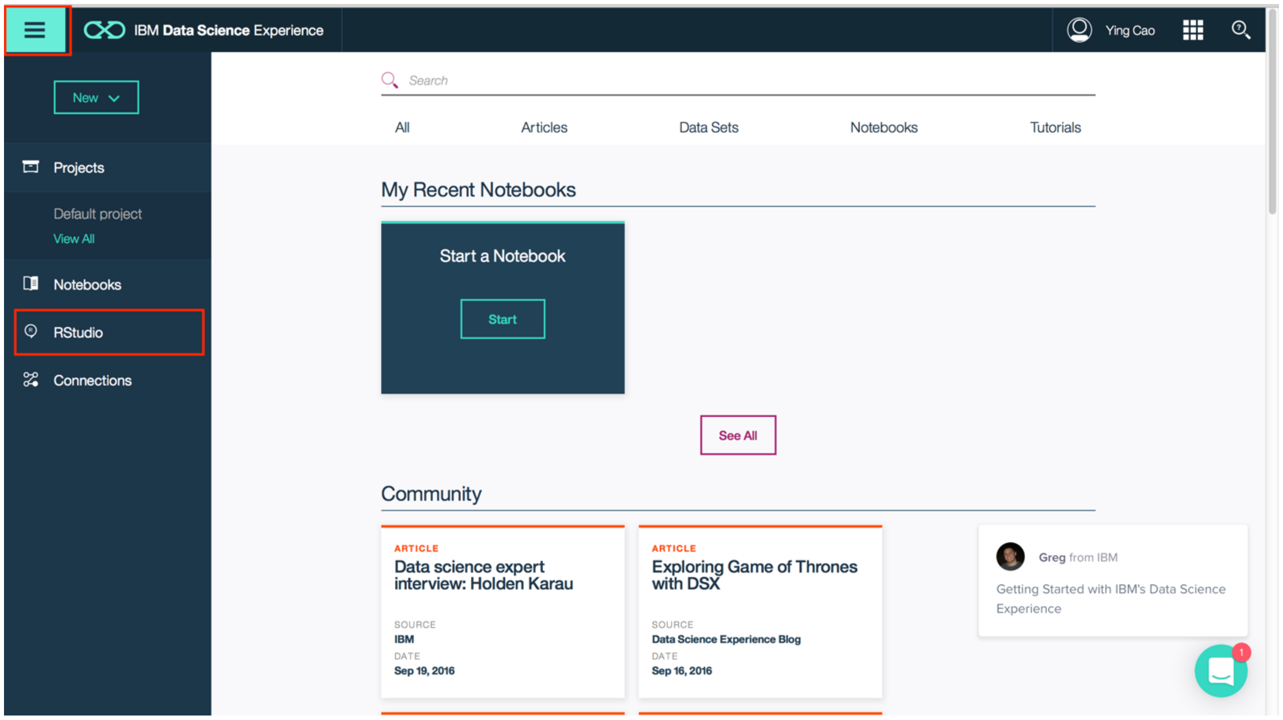
2. Log in to DSX. (Sign up if you have not done so!)



Master the art of data science.

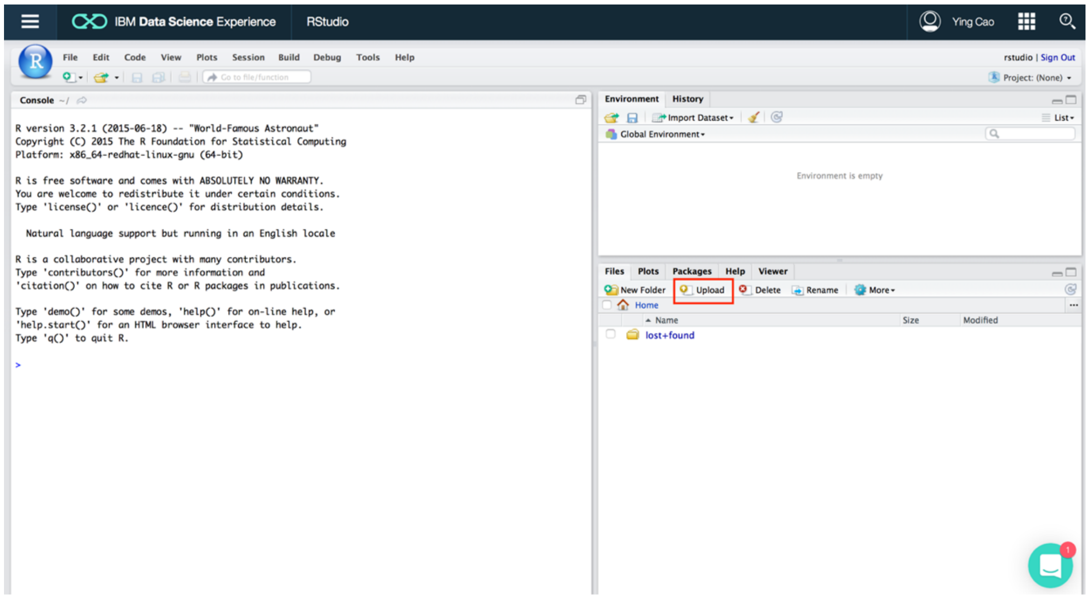
Enjoy the best of open source and collaborate in a social environment, built for data scientists by data scientists.

3. Click ≡ to show the hidden side panel on the left and open RStudio.



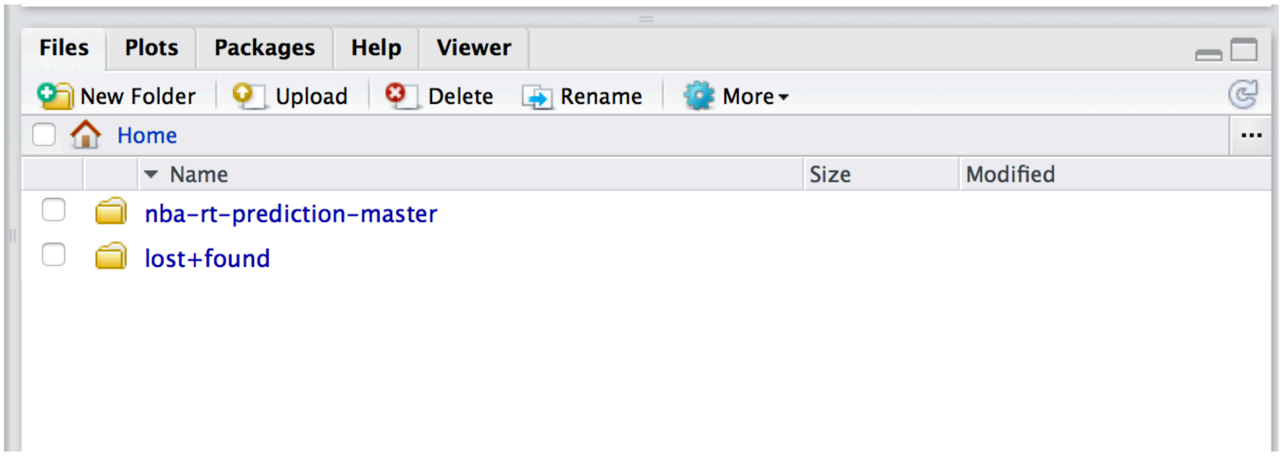
4. Once RStudio is running, you need to upload the Rstudio files into the cloud environment. Do this by clicking upload in the Rstudio file browser and use browse to find the .zip file you have just downloaded to your laptop. Next click OK.

```
Note : Rstudio will automatically unzip the file for you.
```



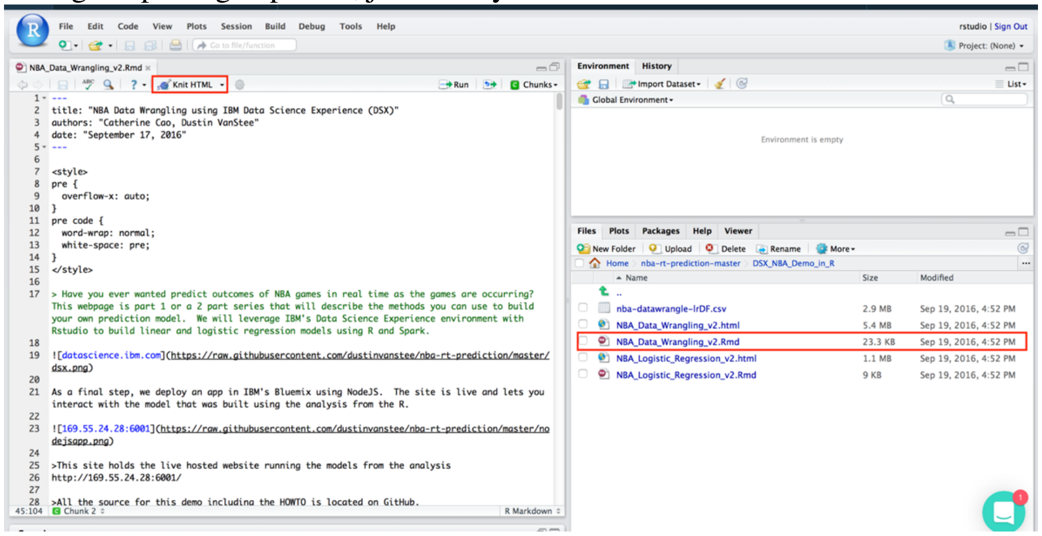
```
R version 3.2.1 (2015-06-18) -- "World-Famous Astronaut"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

5. Once uploaded into Rstudio, use the file browser and navigate nba-rt-prediction-master directory



Name	Size	Modified
nba-rt-prediction-master		
lost+found		

6. Open **NBA_Data_Wrangling.Rmd** in the folder *nba-rt-prediction/DSX_NBA_Demo_in_R*. This will open the R markdown file in the Rstudio text editor. This file embeds both R code, and code for rendering HTML documents.
7. To render the HTML document with embedded R code and graphs click **Knit HTML.** By clicking this, a webpage will get generated for presentations. If there pops up any windows asking for package updates, just click yes.



```
---
title: "NBA Data Wrangling using IBM Data Science Experience (DSX)"
authors: "Catherine Cao, Dustin VanStee"
date: "September 17, 2016"
---

<style>
pre {
  overflow-x: auto;
}
pre code {
  word-wrap: normal;
  white-space: pre;
}
</style>

> Have you ever wanted predict outcomes of NBA games in real time as the games are occurring?
This webpage is part 1 or a 2 part series that will describe the methods you can use to build
your own prediction model.  We will leverage IBM's Data Science Experience environment with
Rstudio to build linear and logistic regression models using R and Spark.

![datascience.ibm.com](https://raw.githubusercontent.com/dustinvanstee/nba-rt-prediction/master/
dsx.png)

As a final step, we deploy an app in IBM's Bluemix using NodeJS.  The site is live and lets you
interact with the model that was built using the analysis from the R.

![169.55.24.28:6001](https://raw.githubusercontent.com/dustinvanstee/nba-rt-prediction/master/no
dejsapp.png)

>This site holds the live hosted website running the models from the analysis
http://169.55.24.28:6001/

>All the source for this demo including the HOWTO is located on GitHub.
```

Name	Size	Modified
..		
nba-datawrangle-lrDF.csv	2.9 MB	Sep 19, 2016, 4:52 PM
NBA_Data_Wrangling_v2.html	5.4 MB	Sep 19, 2016, 4:52 PM
NBA_Data_Wrangling_v2.Rmd	23.3 KB	Sep 19, 2016, 4:52 PM
NBA_Logistic_Regression_v2.html	1.1 MB	Sep 19, 2016, 4:52 PM
NBA_Logistic_Regression_v2.Rmd	9 KB	Sep 19, 2016, 4:52 PM

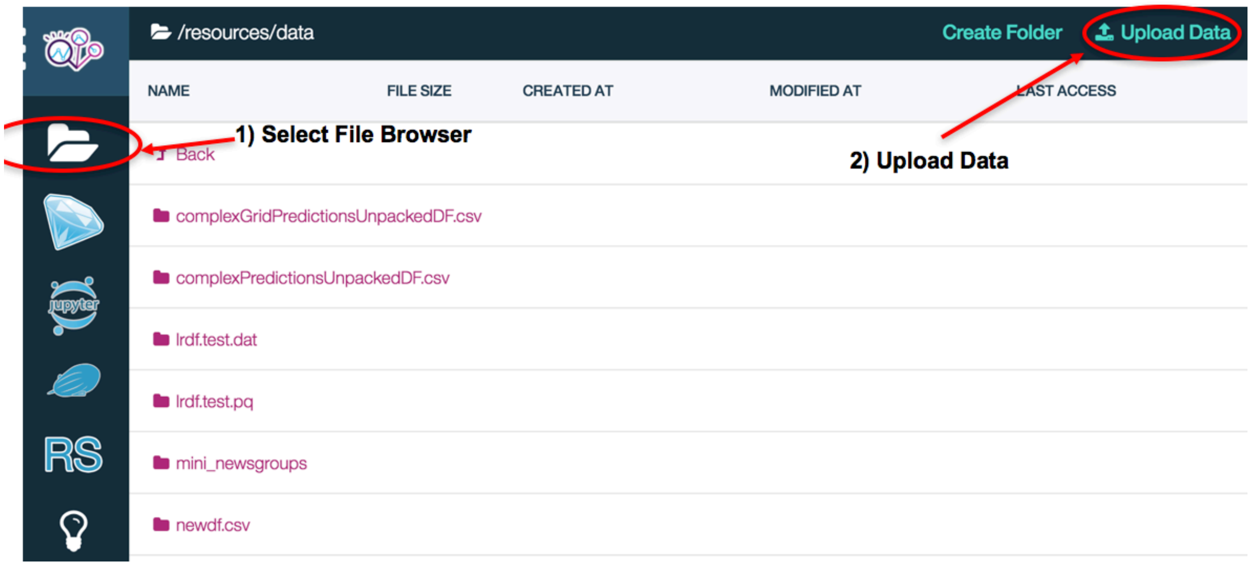
8. The HTML generation process takes some time to execute, but after a minute or two, you will have an HTML rendering of the analysis.
9. Repeat steps 6/7 for the logistic regression analysis with this file : **NBA_Logistic_Regression.Rmd** . Once complete, skip to chapter 3 to build out your Nodejs app that takes the logistic regression coefficients, and creates an online app.

```
Rmarkdown was used to make a nice looking report. You can
directly run R codes in the chunks to conduct data analysis.
There is more documentation within the Rmarkdown file(.Rmd) to
guide you through that portion of the demo.
```

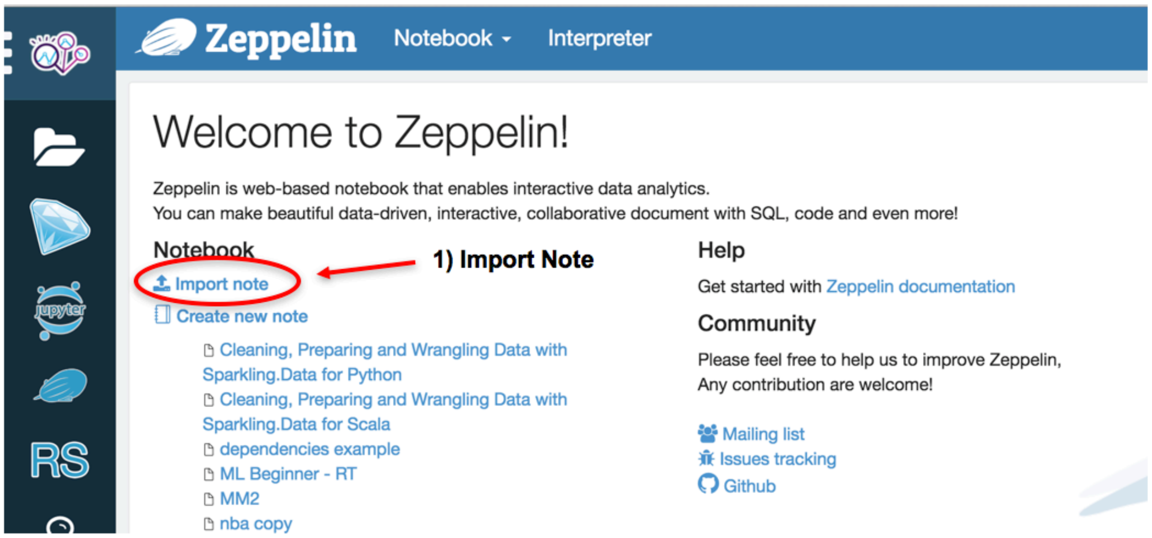
2 Setup the DSWB Environment

The following steps will guide you through the process of setting up datascientist workbench. At the end of this section, you will be able to run your zeppelin notebook and create a logistic regression model in Spark.

1. download https://github.com/dustinvanstee/nba-rt-prediction project
2. Setup DSWB environment
   a. create userid if you have not already done so
   b. For the zeppelin notebook in datascientistworkbench, you will need to upload 3 notebooks and 2 data files into the environment.
      i. Copy nbaodds.xml into /data/resources
      ii. Copy scores_nba.test.dat into /data/resources



iii. Import the 3 json notebooks into your zeppelin Environment


Zeppelin
Notebook
Interpreter
Welcome to Zeppelin!
Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!
Notebook
Import note
1) Import Note
Create new note
Cleaning, Preparing and Wrangling Data with Sparkling.Data for Python
Cleaning, Preparing and Wrangling Data with Sparkling.Data for Scala
dependencies example
ML Beginner - RT
MM2
nba copy
Help
Get started with Zeppelin documentation
Community
Please feel free to help us to improve Zeppelin, Any contribution are welcome!
Mailing list
Issues tracking
Github

Congratulations, you can now run the rest of the demo from within the notebook!

The notebooks are organized into 3 files :

NBA-Predictions-1-DataWrangling.json	: data preparation notebook
NBA-Predictions-2-LinearRegression.json	: Linear regression model
NBA-Predictions-3-LogisticRegression.json	: Logistic regression model

```
Note, you need to run the entire data wrangling notebook prior
to running the modelling notebooks
```

Click on the notebook you just added and run each of the cells. There is more documentation within the notebook to guide you through that portion of the demo.

```
INFO: This completes the Analytics Setup portion of the
document
```

3 Reproduce the Bluemix Web Application Environment

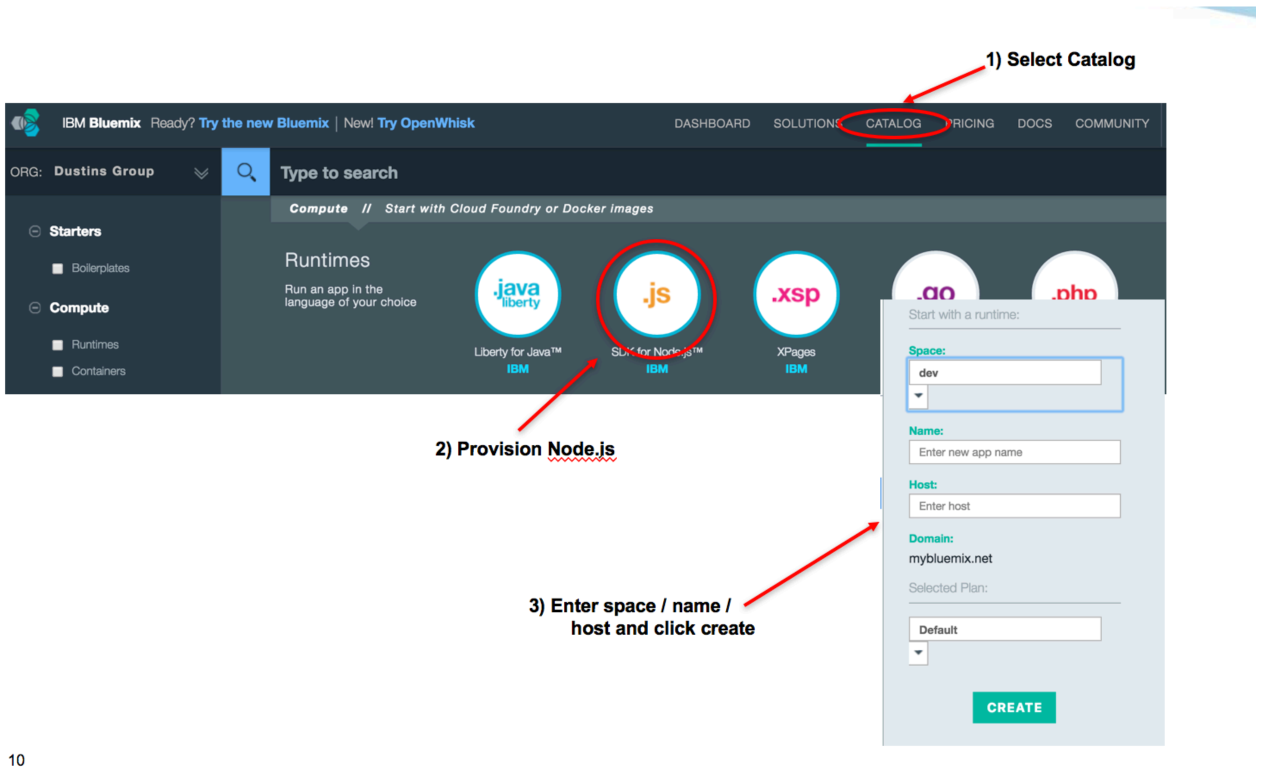
3.1 Software pre-requisites

- Install Node Package Manager (npm) - https://docs.npmjs.com/getting-started/installing-node
- Setup a Bluemix account - https://console.ng.bluemix.net/registration

3.2 Bluemix Configuration and Setup

To setup the web application portion of your demo, we will use the node.js framework in Bluemix. This framework provides convenient method to develop and test our application on your local machine, and then deploy the finished product to Bluemix. The first step will be to deploy the Node.js framework from within Bluemix as shown below

3.2.1 Install Node.js Framework


1) Select Catalog
IBM Bluemix Ready? Try the new Bluemix | New! Try OpenWhisk
DASHBOARD SOLUTIONS CATALOG PRICING DOCS COMMUNITY
ORG: Dustins Group
Type to search
Compute // Start with Cloud Foundry or Docker images
Starters
Boilerplates
Compute
Runtimes
Containers
Runtimes
Run an app in the language of your choice
Liberty for Java™ IBM
SDK for Node.js™ IBM
XPages IBM
2) Provision Node.js
3) Enter space / name / host and click create
CREATE
10

Start with a runtime:

Space: dev

Name: Enter new app name

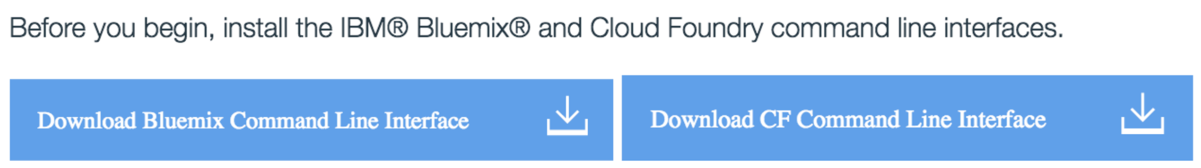
Host: Enter host

Domain: mybluemix.net

Selected Plan: Default

3.2.2 Customize our Node.js app

While the service is being deployed, your screen will display the instructions for the next steps. You will need to install the following tools so that you can push your web application to Bluemix. Follow the instructions to install those 2 applications to your computer.


Before you begin, install the IBM® Bluemix® and Cloud Foundry command line interfaces.
Download Bluemix Command Line Interface
Download CF Command Line Interface

From here, we will deviate slightly from the directions in Step 1 of the getting started. Instead of downloading the starter app, you will use the app downloaded earlier from github.

```
tar —zxvf nodejs.tar.gz
mv nodejs/ <directoryOfYourChoice>
```

Unzip the Node.js web application and move the directory to the desired location on your computer

Next, we will follow the directions as laid out in the getting started guide for node.js in Bluemix. Here are the commands I typed to get my temporary node.js environment working….

```
bluemix api https://api.ng.bluemix.net
```

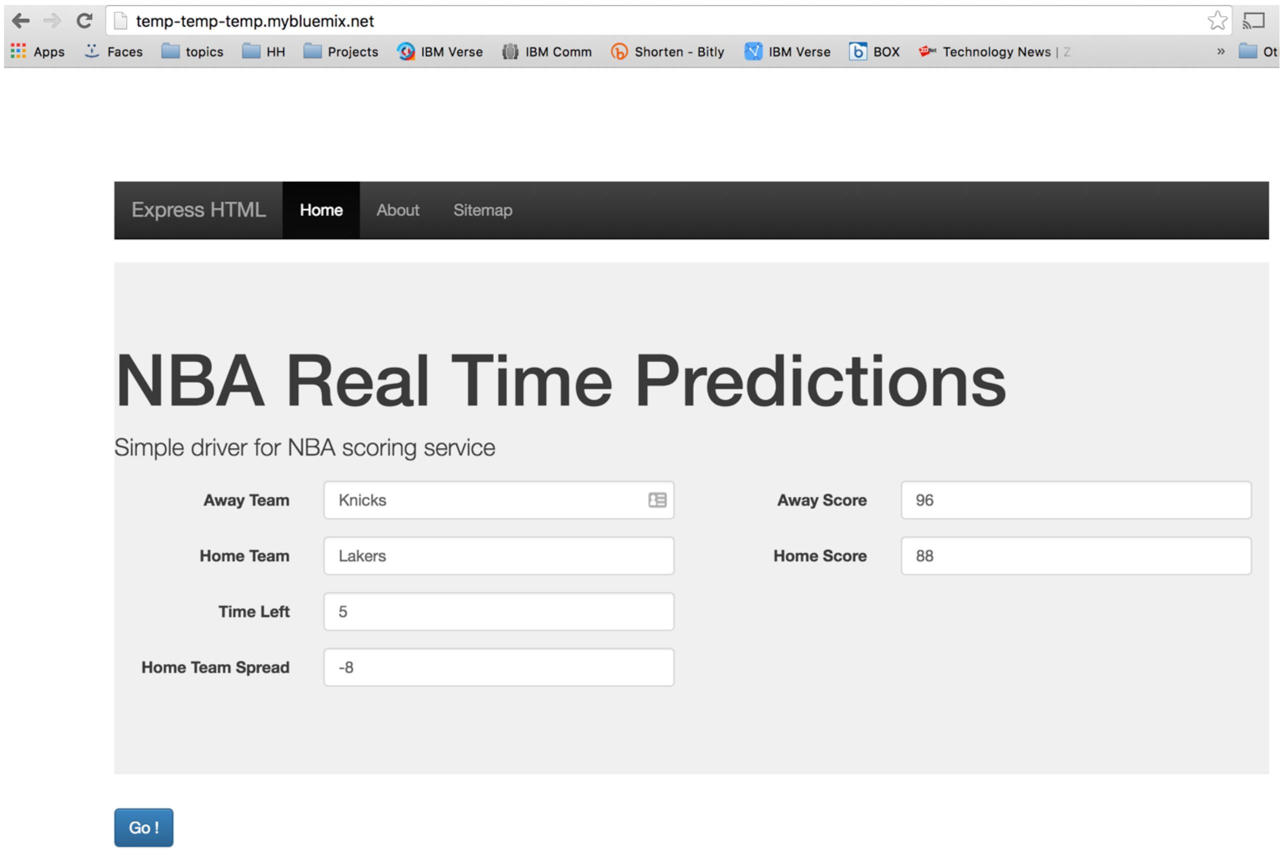
```
Invoke 'cf api https://api.ng.bluemix.net'...

Setting api endpoint to https://api.ng.bluemix.net...
OK


API endpoint:   https://api.ng.bluemix.net (API version: 2.44.0)
User:           vanstee@us.ibm.com
Org:            vanstee@us.ibm.com
Space:          dev
```

```
cd /tmp/nodejs/
# use cloud foundary to push my application to my Node.js
# instance name in bluemix (named temp-temp-temp)
cf push temp-temp-temp
```

```
Updating app temp-temp-temp in org Dustins Group / space dev as vanstee@us.ibm.com... cf
push temp-temp-temp

OK

Uploading temp-temp-temp...
Uploading app files from: /private/tmp/nodejs
Uploading 4.8M, 1271 files
Done uploading
OK

Stopping app temp-temp-temp in org Dustins Group / space dev as vanstee@us.ibm.com...
OK

Starting app temp-temp-temp in org Dustins Group / space dev as vanstee@us.ibm.com...
-----> Downloaded app package (7.2M)
-----> Downloaded app buildpack cache (456K)

-----> IBM SDK for Node.js Buildpack v3.3-20160428-1409
       Based on Cloud Foundry Node.js Bui
ldpack v1.5.4
-----> Creating runtime environment
```

3.2.3 Test our Node.js app

Once the code push iscomplete, you have now deployed your web application ! You can test this by pointing your browser to your apps URL and you should see the screen below.



NBA Real Time Predictions

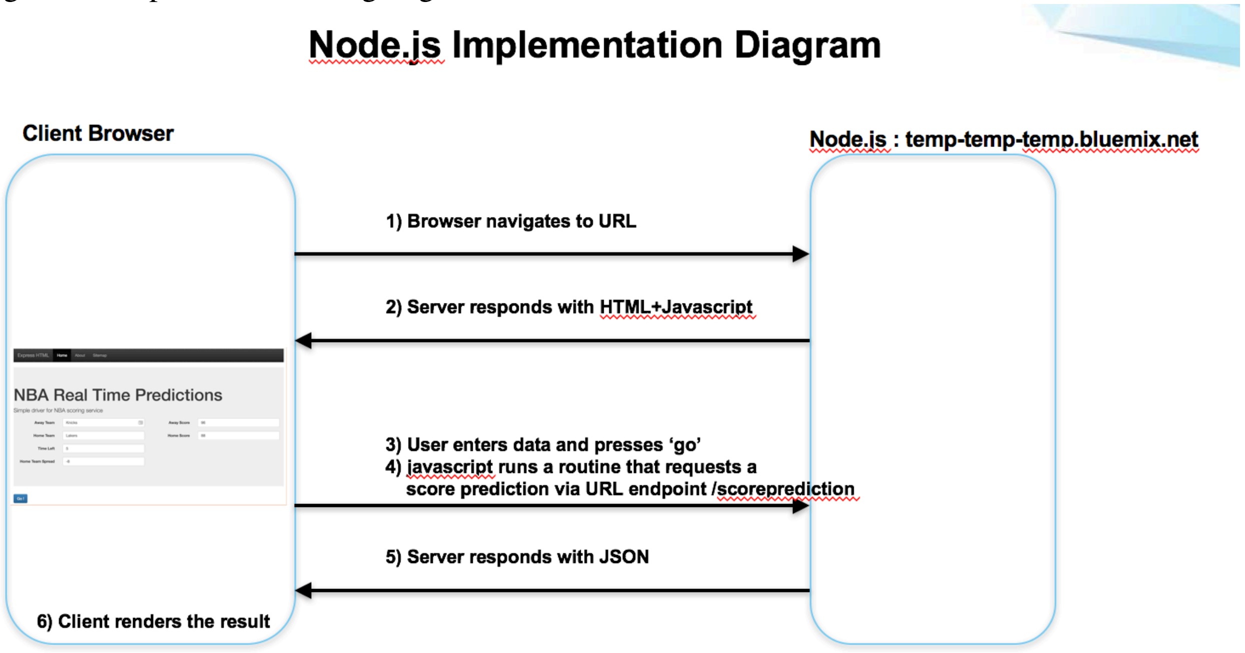
Simple driver for NBA scoring service

Away Team	Knicks	Away Score	96
Home Team	Lakers	Home Score	88
Time Left	5		
Home Team Spread	-8		

3.2.4 Node.js app implementation details

When you point your browser to this URL, the node.js server that is listening at that URL responds with this webpage (coded with Bootstrap). You can enter some data into the cells and click 'go'.

Once you click to, some javascript is run on the browser client to send a request to the node.js service to score this outcome based on the point spread, time left, and home/away score. Some JSON is returned from the node.js service and formatted in a simple tabular view. Here is a diagram the represents what is going on 'under the hood'

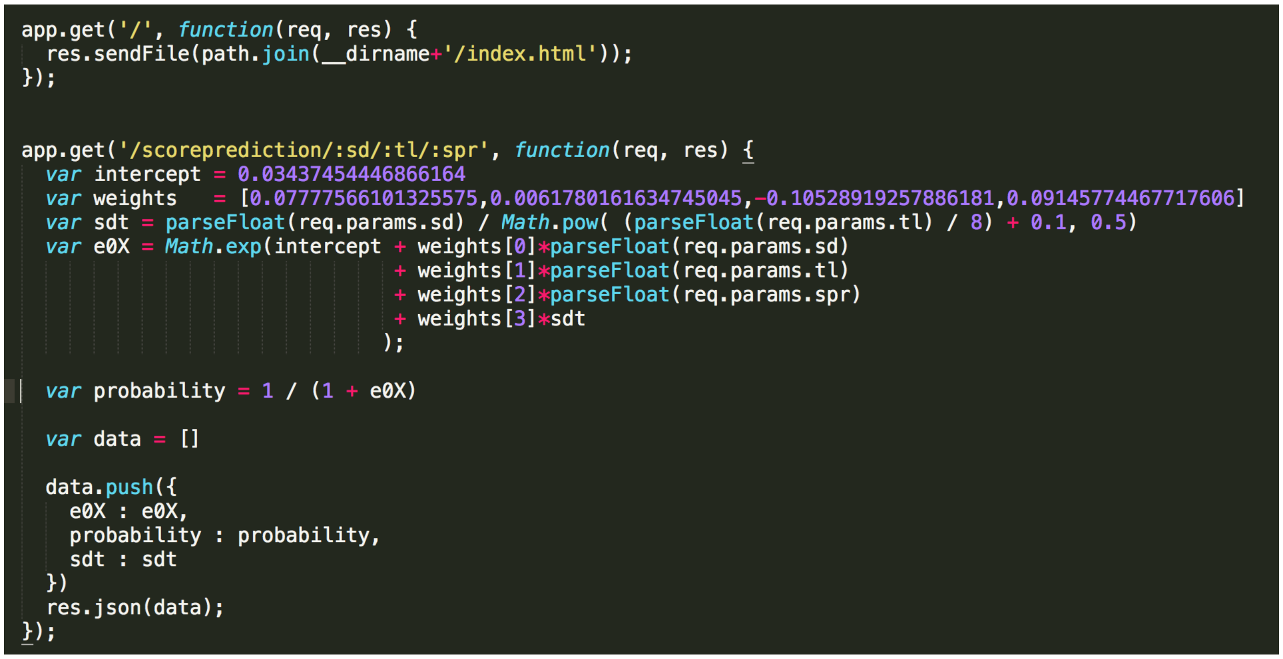

Node.js Implementation Diagram
Client Browser
Node.js : temp-temp-temp.bluemix.net
1) Browser navigates to URL
2) Server responds with HTML+Javascript
NBA Real Time Predictions
3) User enters data and presses 'go'
4) javascript runs a routine that requests a score prediction via URL endpoint /scoreprediction
5) Server responds with JSON
6) Client renders the result

3.2.5 Code layout

There are 3 main files that contain most of the custom code for this project

app.js	: node.js file that uses express library to create a rest API
public/index.html	: NBA real time prediction web page
public/rtp.js	: javascript function that is invoked by clicking 'go'

I will briefly cover the the app.js file. This file make use of a few libraries, with the main library being express. ExpressJs provides all the web server functions you need to make a web server without have to know all the details of the underlying framework. The custom part of the code is this code below. Here you can see that I have 2 endpoints under my base URL defined. When a browser navigates to the base URL + / , the main index.html page is return. If a user were to browse to the base URL + /scoreprediction/ , the scoreprediction service is run with 3 inputs that are passed as part of the URL. This code executes the logistic regression scoring model (more details on that in the next section), and returns an answer back to the client in JSON format.



```
app.get('/', function(req, res) {
  res.sendFile(path.join(__dirname+'/index.html'));
});


app.get('/scoreprediction/:sd/:tl/:spr', function(req, res) {
  var intercept = 0.03437454446866164
  var weights   = [0.07777566101325575,0.0061780161634745045,-0.10528919257886181,0.09145774467717606]
  var sdt = parseFloat(req.params.sd) / Math.pow( (parseFloat(req.params.tl) / 8) + 0.1, 0.5)
  var e0X = Math.exp(intercept + weights[0]*parseFloat(req.params.sd)
                               + weights[1]*parseFloat(req.params.tl)
                               + weights[2]*parseFloat(req.params.spr)
                               + weights[3]*sdt
                              );

  var probability = 1 / (1 + e0X)

  var data = []

  data.push({
    e0X : e0X,
    probability : probability,
    sdt : sdt
  })
  res.json(data);
});
```

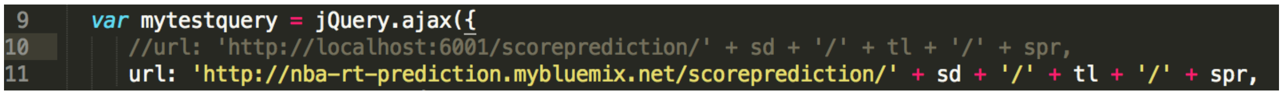
The webpage was built using a nice web framework called bootstrap. I like it because the formatting and look of the pages is very moderation and fairly easy to manipulate. I won't go into the gory detail for the webpage and javascript, but will just highlight the one line of code that is activated when the 'go' button is pushed

```
<form class="form-horizontal container" role="form"
onSubmit="return
handleClick(this.homescorertp.value,this.awayscorertp.value,this.t
imeleftrtp.value,this.spreadrtp.value,this.homertp.value,this.away
rtp.value)">
```

What happens is that a handleClick function gets called with the values that are in the form of the webpage. This function is coded in the rtp.js. The rtp.js:handleClick function calls the /scoreprediction service on the server side and then waits (asynchronously) for the JSON data to be return. The result values are shown in the results table.

3.2.6 Code Modification

There is one dependency that needs to get updated in the public/rtp.js file so that you can use your own score prediction service instead of the one I setup (I have a hardcoded URL in the code). To fix this modify this line of code to point to your web URL (this is the value you typed in for the name of your Node.js project when you created it in Bluemix).



```
  var mytestquery = jQuery.ajax({
    //url: 'http://localhost:6001/scoreprediction/' + sd + '/' + tl + '/' + spr,
    url: 'http://nba-rt-prediction.mybluemix.net/scoreprediction/' + sd + '/' + tl + '/' + spr,
```

Once you have done this, you can push this change to Bluemix by running this command in your node.js project directory.

```
cf push nba-rt-prediction
```

This will copy the entire project back up to Bluemix and redeploy your app. You can also run the app locally for debugging purposes if you type this

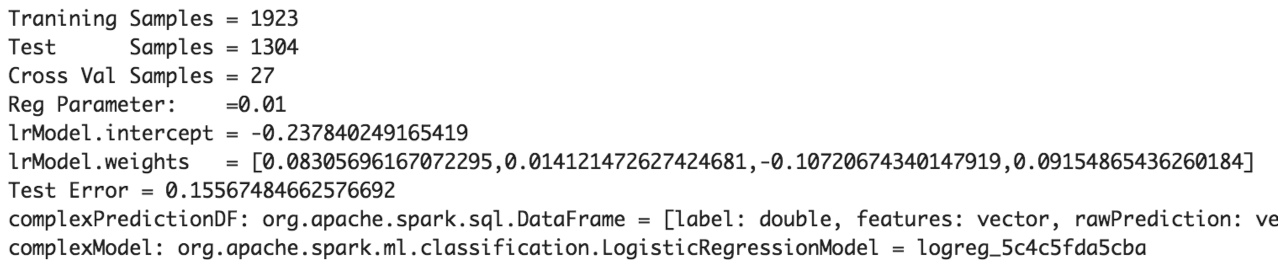
```
 node app.js
```

This will setup the webserver locally on your laptop and you can point your browser to localhost:6001 for local debugging.

```
INFO: This completes the Bluemix Setup portion of the document
```

4 How to 'Operationalize' Machine Learning Insights

So far, we have been able to reproduce the environment, but let's have a deeper look at what is happening in this demo environment. In the Zeppelin workbook, we went through a number of data refinement steps to produce an input data set. We then took that input data set and split it into a training and test partition that we fed into the logistic regression modelling functions within Spark MLLIB. Simply stated, the logistic regression model returns a set of weights that transform the inputs into the predicted output. In our test we had just 4 'features'. Here is what the logistic regression output summary looks like in the zeppelin notebook.



```
Tranining Samples = 1923
Test      Samples = 1304
Cross Val Samples = 27
Reg Parameter:    =0.01
lrModel.intercept = -0.237840249165419
lrModel.weights   = [0.08305696167072295,0.014121472627424681,-0.10720674340147919,0.09154865436260184]
Test Error = 0.15567484662576692
complexPredictionDF: org.apache.spark.sql.DataFrame = [label: double, features: vector, rawPrediction: ve
complexModel: org.apache.spark.ml.classification.LogisticRegressionModel = logreg_5c4c5fda5cba
```

For the model, the important values to focus on are the intercept and the weights. For this specific example the weights will scale to the following input variables [score_difference, time_left, spread, custom_score_diff].

To implement simple logistic regression in our Node.js web application, we just need to know how to apply these weights to the logistic regression formula. We implement the formula using this equation

$$f(X;\theta) = sigmoid(\theta^T X)$$

where

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

Explicitly in this example the inputs from the web form (X) are multiplied by the weights (theta) as follows

$$\theta^T X = -0.23 + 0.08 * X_1 + 0.014 * X_2 + -0.107 * X_3 + 0.09 * X_4$$

We then take the number apply the sigmoid function, and out comes a probability. The higher the number the more likely the home team will win. A value near 0.50 means that our model has no more predictive value than flipping a coin. The zeppelin notebook has some commentary about the weights, so I will omit the discussion here.

In practice, a pattern like this works well enough to get the job done when the number of inputs to the model are small, but for models that are more complex(neural nets, svm) or updated frequently this manual coding approach doesn't work so well. We benefitted from the fact that the logistic equation can be coded in a few lines of code, but we should not always expect that to be the case. There are some other alternatives I will discuss in the next section ….

4.1 Extensions to this demo …

Extensions to what has been shown here include methods to connect our Node.js framework directly to spark for execution of the scoring. Frameworks like EclairJS look very interesting from this perspective. For the next revision of this demo I plan try to implement something like that. Here is a brief list of some other improvements ….

Implement machine learning pipeline to select best hyper parameters for model
Experiment with linear regression to predict final score
Assess prediction sensitivity and margin of error (variance)