Austin
Big Data
AI
Meetup

30
NOV

Thursday, November 30, 2017

Real-time Sporting Event Prediction - DL and ML Lecture Series -2

Hosted by Dustin VanStee and Tuhin Mahmud
From Austin Big Data AI

Presenter : Dustin VanStee
IBM Cognitive Solution Engineer
Twitter : @dustinvanstee
LinkedIn : https://www.linkedin.com/in/dustinvanstee

Meetup Agenda

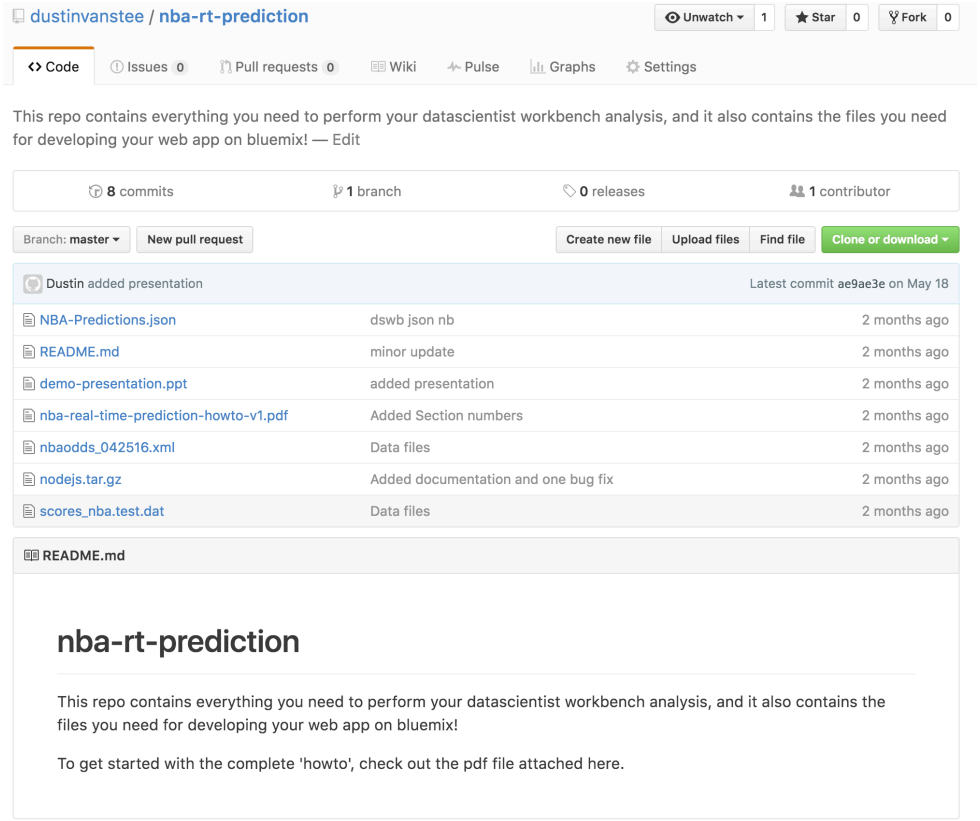
- Use case overview
- Online prediction demonstration
- Data Flow and Architecture
- Tools/Technologies used for this project

- Use Case Deep Dive / Notebook Demo using IBM Data Science Experience and Nimbix Cloud Platform
  - Data Preparation
  - Some Visualizations
  - Logistic Regression
  - Neural Network

- General discussion and questions

NBA Real Time Predictions
All the notebooks from today's session is posted on Github!

https://github.com/dustinvanstee/nba-rt-prediction



dustinvanstee / nba-rt-prediction

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Wiki | Pulse | Graphs | Settings

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

8 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

Dustin added presentation — Latest commit ae9ae3e on May 18

NBA-Predictions.json	dswb json nb	2 months ago
README.md	minor update	2 months ago
demo-presentation.ppt	added presentation	2 months ago
nba-real-time-prediction-howto-v1.pdf	Added Section numbers	2 months ago
nbaodds_042516.xml	Data files	2 months ago
nodejs.tar.gz	Added documentation and one bug fix	2 months ago
scores_nba.test.dat	Data files	2 months ago

README.md

nba-rt-prediction

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!

To get started with the complete 'howto', check out the pdf file attached here.

Use Case

Goal : To generate NBA game outcome predictions while game is active

Method : use historical game data and odds data, and create some machine learning models

Motivation : to apply data science methods to a use case with general applicability

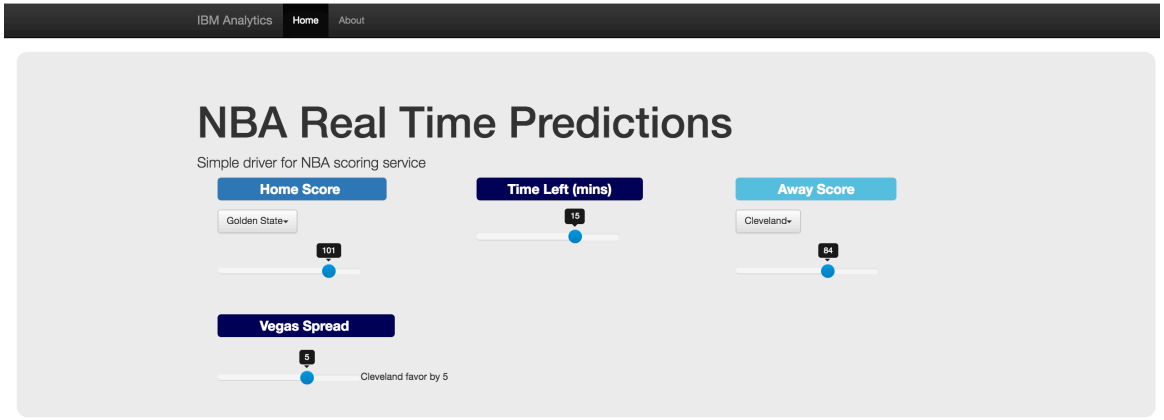
What is the probability the Cavs or Warriors will win during the game ?
Are there other data sources that can help … ?
Are there other analytics we can show ….?

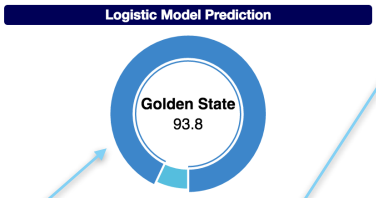
Spark MLlib

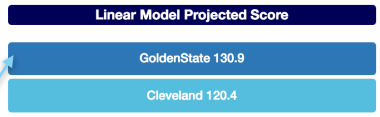
NBA

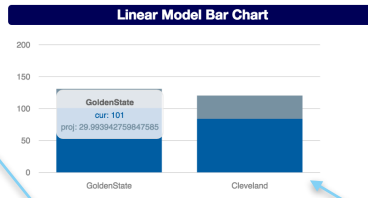
Rather than explain, lets just give it a try first!

https://nba-rt-demo.mybluemix.net









** Logistic / Linear Models built with slightly different methods. There may be some disagreement near 50/50 cases

Winner probability

Logistic Regression

Home Team Final Score

Linear Regression

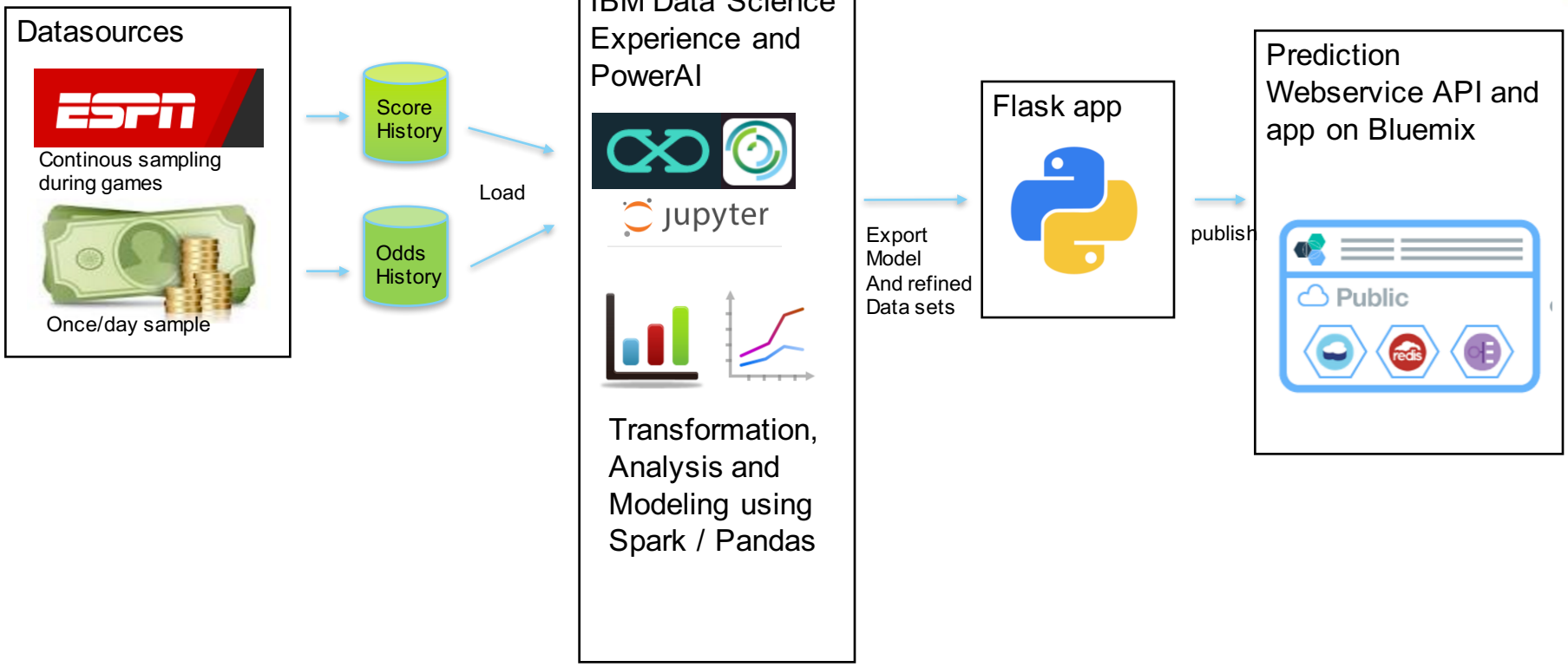
Away Team Final Score

Linear Regression

Final Score Bar Chart

Lets get into the details ….

Architecture and Data Flow



Tooling

Data Acquisition	Analysis / Modelling	Operationalize
Custom scripts running on a server during the month of April	Data Scientist Experience PowerAI Jupyter Notebooks Spark / Pandas Brunel Visualization	Bluemix – Flask app Watson Machine Learning Bootstrap
Python / Bash	Python	Javascript / Python

A few details on the platforms / technologies used during this meetup

Data Science Experience brings all the tools you need into one place

IBM Data Science Experience

Community

- Find tutorials and datasets
- Connect with Data Scientists
- Ask questions
- Read articles and papers
- Fork and share projects

Open Source

- Code in Scala/Python/R/SQL
- Jupyter Notebooks
- RStudio IDE and Shiny
- Apache Spark
- Your favorite libraries

IBM Added Value

- Projects and Version Control
- Spark Service and Remote Spark
- IBM Machine Learning
- Platform Manager

What is Spark?

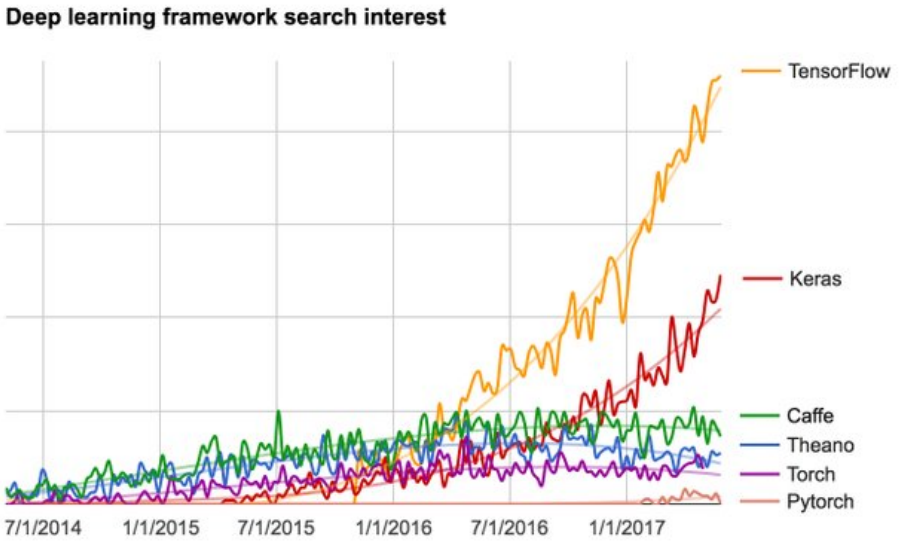
Spark is an **open** source
in-memory
application framework for
distributed data processing and
iterative analysis
on **massive** data volumes

Deep Learning Frameworks

Python Libraries for Data Scientists

new stars from 2017-04-20 to 2017-07-06		
#1:	7929	tensorflow/tensorflow
#2:	2465	fchollet/keras
#3:	1894	caffe2/caffe2
#4:	1526	BVLC/caffe
#5:	1250	pytorch/pytorch
#6:	1233	Microsoft/CNTK
#7:	979	dmlc/mxnet
#8:	709	deepmind/sonnet
#9:	690	tflearn/tflearn
#10:	485	deeplearning4j/deeplearning4j
#11:	458	Theano/Theano
#12:	452	davisking/dlib
#13:	341	torch/torch7
#14:	303	baidu/paddle
#15:	243	pfnet/chainer

Source: github and google



Tensorflow and Caffe remain most popular frameworks combined with wrappers and tools for ease of use, rapid prototyping, reduced/no coding or specialized networks or uses such as Keras, Digits, AI Vision

Keras works with Theano, Tensorflow, CNTK, and MXNet. R users can use Keras without having to switch over to Python. Sharing models and weights is easier if all frameworks unify around same API.

13

IBM PowerAI Platform
August 2017

v4 –

PowerAI Software Distribution

Deep Learning Frameworks

- Caffe
- NVIDIA Caffe
- IBM Caffe
- torch
- TensorFlow
- theano
- Chainer

Supporting Libraries

- DIGITS
- OpenBLAS
- Distributed Frameworks
- Bazel
- NCCL

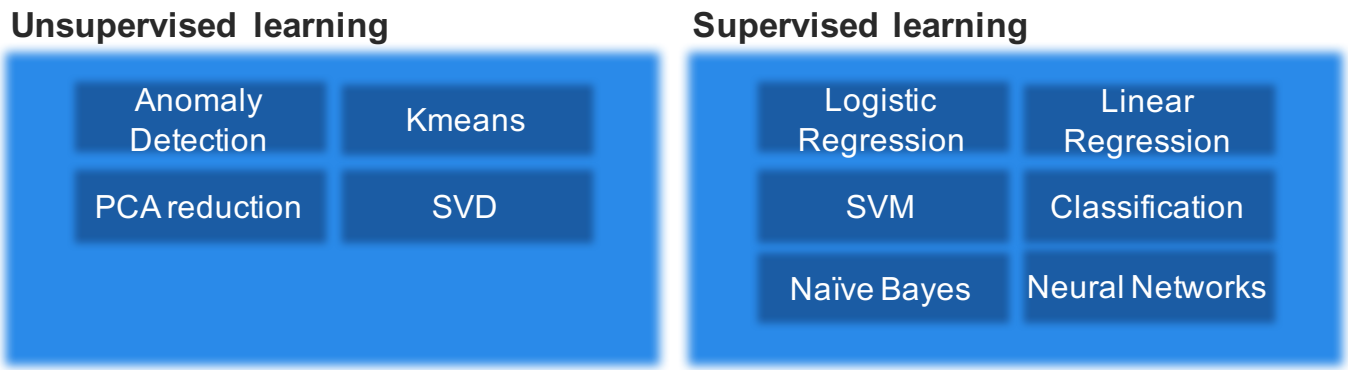
https://public.dhe.ibm.com/software/server/POWER/Linux/mldl/ubuntu/

Supervised and Unsupervised Learning

Machine learning algorithms breakdown into two categories :

Supervised learning : These algorithms use 'labeled' data where there are multiple inputs(features) used with one target or labeled data value that is to be predicted. Classification algorithms fall into this category.

Unsupervised learning : These algorithms try to find structure in the underlying data without a specific target variable. Algorithms in this category include clustering and anomaly detection.



Regression and Classification

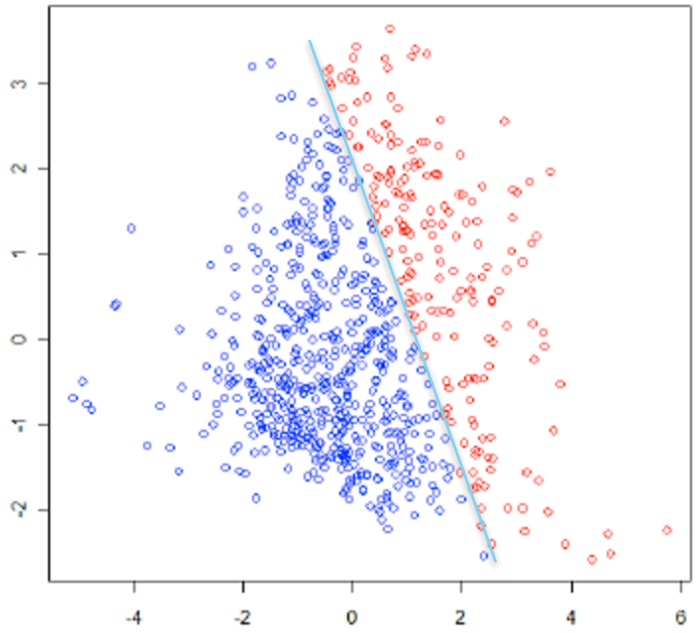
- Regression algorithms are used to try to predict a relationship between a dependent value and one or more independent values
  - Typically used to model continuous outcomes eg. Housing prices, basketball scores

- Classification aims to divide items into categories
  - The most common classification type is binary classification, where there are two categories
  - If there are more than two categories, it is called multiclass classification

- Logistic regression is a popular method to predict a binary response

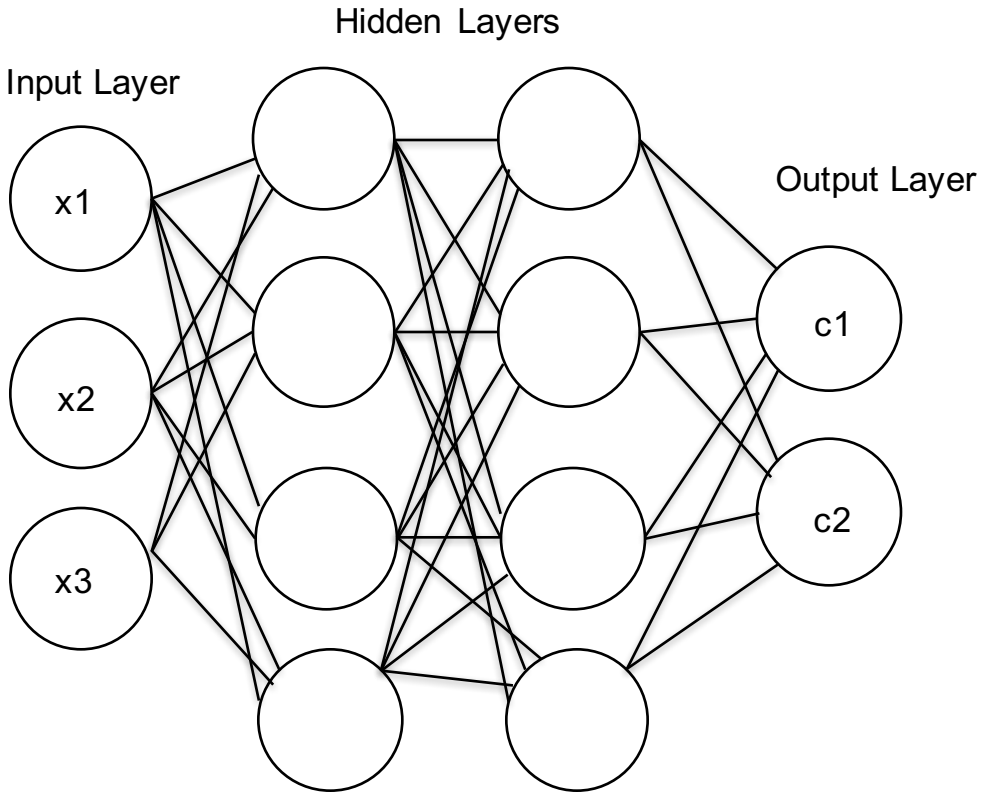
  It is a special case of Generalized Linear models that predict the probability of an outcome

  Binary logistic regression can be generalized into multinomial logistic regression to train and predict multiclass classification problems
  - *The current implementation of logistic regression in spark.ml only supports binary classes. Support for multiclass regression will be added in the future.*

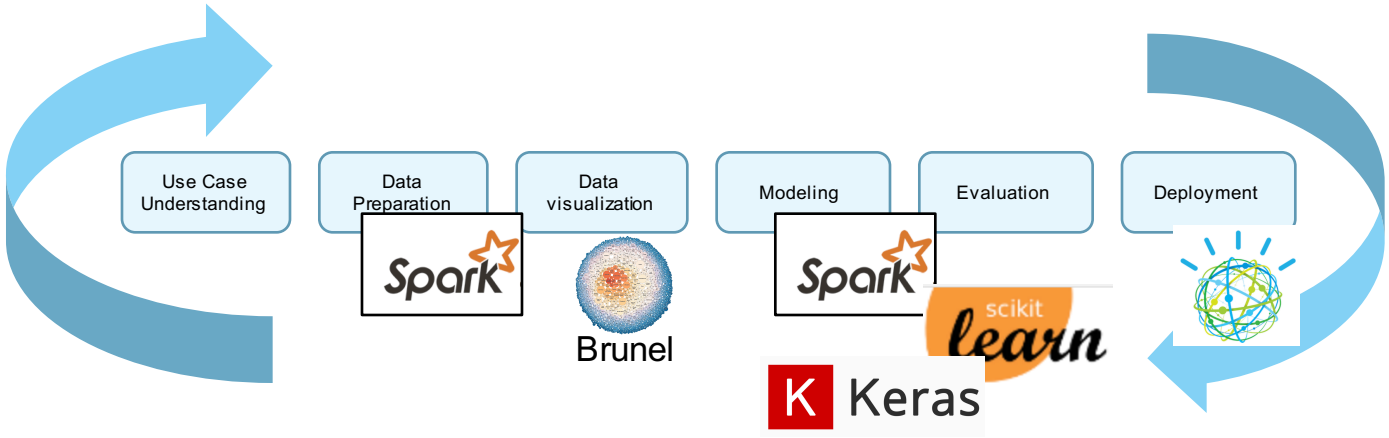


Example of logistic regression

Neural Network



ML Methodology



Data science is an iterative process that includes more than just modeling. It consists of these typical steps highlighted above.

The following demo would built using this workflow in mind

Use Case Input Data – 2 Data Sets

Historical Scores Format

Continous sampling during games

```
2016-04-05,19:23:42,New Orleans,23,Philadelphia,12,(4:39 IN 1ST),40.65,400829041
2016-04-05,19:36:06,New Orleans,31,Philadelphia,23,(11:14 IN 2ND),35.2333333333,400829041
2016-04-05,19:36:21,New Orleans,33,Philadelphia,23,(10:55 IN 2ND),34.9166666667,400829041
2016-04-05,19:36:36,New Orleans,33,Philadelphia,23,(10:52 IN 2ND),34.8666666667,400829041
2016-04-05,19:36:52,New Orleans,33,Philadelphia,25,(10:48 IN 2ND),34.8,400829041
2016-04-05,19:37:38,New Orleans,33,Philadelphia,27,(10:07 IN 2ND),34.1166666667,400829041
2016-04-05,19:38:23,New Orleans,33,Philadelphia,29,(9:53 IN 2ND),33.8833333333,400829041
2016-04-05,19:42:10,New Orleans,33,Philadelphia,32,(9:02 IN 2ND),33.0333333333,400829041

2016-04-05,21:22:09,New Orleans,93,Philadelphia,107,(FINAL),0.0,400829041
```

Script used to scrape scores continuously every 30 secs
Data saved if score changed

Vegas odds format

Once/day sample

```
<title>New Orleans 2.5 O (207.0) 125.0 | Phila. -2.5 U (207.0) -145.0 (Apr 05, 2016 07:10 PM)</title>
<title>Detroit 4.0 O (202.0) 160.0 | Miami -4.0 U (202.0) -190.0 (Apr 05, 2016 08:05 PM)</title>
<title>Charlotte 4.0 O (200.5) 155.0 | Toronto -4.0 U (200.5) -175.0 (Apr 05, 2016 07:40 PM)</title>
<title>Phoenix 14.5 O (207.5) -110.0 | Atlanta -14.5 U (207.5) -110.0 (Apr 05, 2016 08:10 PM)</title>
<title>Chicago -3.0 O (201.5) -150.0 | Memphis 3.0 U (201.5) 130.0 (Apr 05, 2016 08:10 PM)</title>
```

How To Read The Raw Odds Data

FINISHED

How to interpret the odds data ...

```
Example Golden State -12.5 O (207.0) -125.0 | Detroit 12.5 U (207.0) 145.0
The away team is listed first, and the home team is second
Here Golden State is a 12.5 pt favorite to win.  The over under is in parentheses (207) and is the 50/50 line between teams sum of scores
being above/below that line.
Finally the -125 / +145 numbers are whats known at the moneyline odds.
    A negative number means you need to bet 125$ to get a 100$ payout
    A positive number means you need to bet 100$ to get a 145$ payout
```


** Data was sampled for month of April 2016 only **

Data wrangling in notebook .
Step 1. Build score history data set with final score outcomes

Score history

Home Team	Home Score	Away Team	Away Score	Time Status String
Detroit	89	New York	90	5:00 left in the 4th
Detroit	95	New York	97	Final

100+ records per game

Add Extra Features

Home-win	Final score diff
false	2

Replicate this for every in game data point

Data wrangling in notebook .
Step 2. Merge in odds data

Odds History

Home Team	Spread	Away Team
Detroit	-10	New York
Chicago	5	Portland
…		

1 record per game

Data Layout from Step 1

Home Team	Home Score	Away Team	Away Score	Time Status String	Final Score Home	Final Score Away	Winner
Detroit	89	New York	90	5:00 left in the 4th	95	97	Knicks
…							

Approx 100 records per game

Join the Odds data with the Historical Scores

Key will be of the form : date.awayTeam.homeTeam
Score data was loaded into rtscoresAndFinalDF dataframe

Join The Game Dataframe With The Real Time Score Dataframe

```
cleanedDF = rtscoresDF.join(gameDF, rtscoresDF["key"] == gameDF["key"], "inner").drop(gameDF["key"]).cache()
```

Join by key that is equivalent to Date + HomeTeam + awayTeam

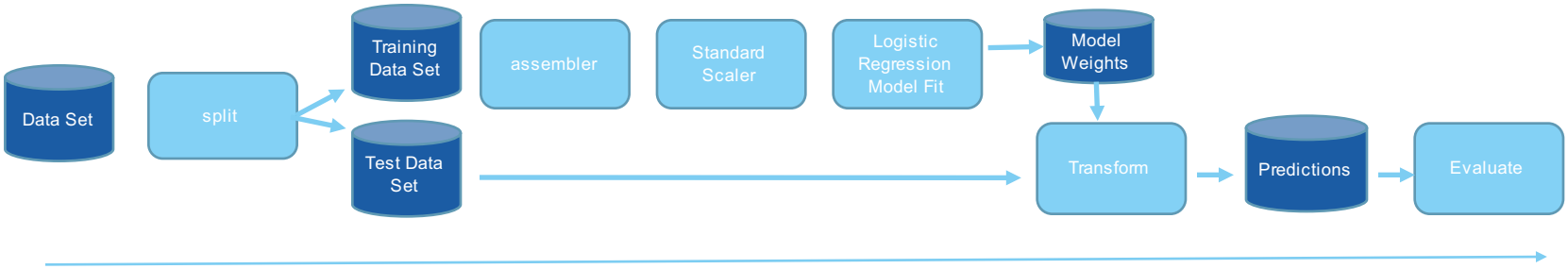
Final Result

```
cleanedDF : Cleaned Data Frame for use with ML algos
Total Data Points in rtscoresDF = 40422
Total Data Points in gameDF = 266
Total Data Points in joined cleanedDF = 30864
```

	dateOrig	ts	away_team_full	away_score	home_team_full	home_score	timestring	timeleft	away_team	home_team	score_diff_an
0	2017-11-01	12:27:48	Phoenix	0.000000	Washington	0.000000	(7:00 PM ET)	48.0	pho	wsh	0.0000
1	2017-11-01	19:23:28	Phoenix	0.727273	Washington	1.545455	(6:30 IN 1ST)	47.5	pho	wsh	-0.8181
2	2017-11-01	19:23:28	Phoenix	1.454545	Washington	3.090909	(6:30 IN 1ST)	47.0	pho	wsh	-1.6363
3	2017-11-01	19:23:28	Phoenix	2.181818	Washington	4.636364	(6:30 IN 1ST)	46.5	pho	wsh	-2.4545
4	2017-11-01	19:23:28	Phoenix	2.909091	Washington	6.181818	(6:30 IN 1ST)	46.0	pho	wsh	-3.2727

away_team_ml	home_team_spread	home_team_ml	dateStr	away_team_vegas_fscore	home_team_vegas_fscore	key
-110	-12.5	-110	2017-11-01	107.5	120.0	2017-11-01.pho.wsh
-110	-12.5	-110	2017-11-01	107.5	120.0	2017-11-01.pho.wsh
-110	-12.5	-110	2017-11-01	107.5	120.0	2017-11-01.pho.wsh
-110	-12.5	-110	2017-11-01	107.5	120.0	2017-11-01.pho.wsh
-110	-12.5	-110	2017-11-01	107.5	120.0	2017-11-01.pho.wsh

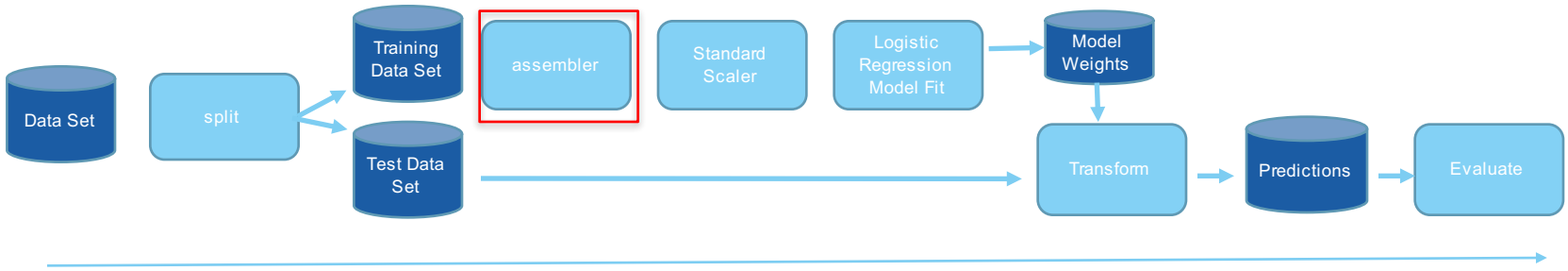
Logistic Regression Example



In the next few slides we review a few key examples

*** For simplicty is an example omits cross validation**

Logistic Regression Example



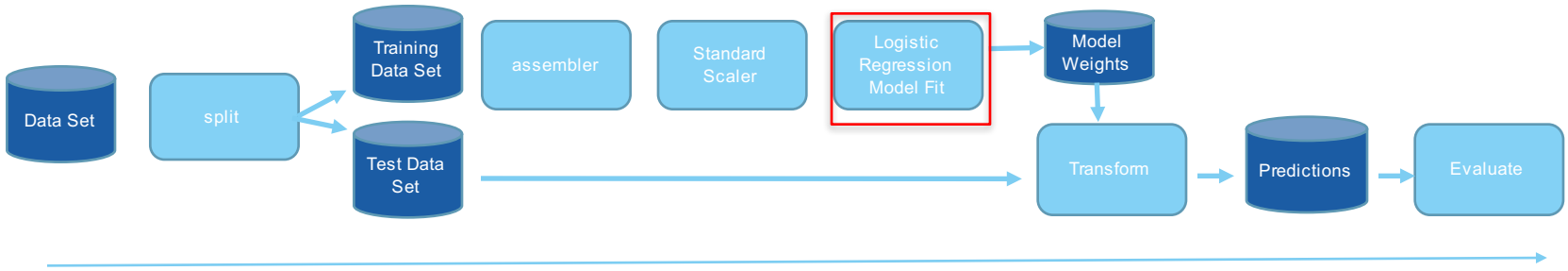
Prior to fitting a model, there must be a features column of type Vector, and a label column. The assembler routine makes this easy by creating a features column, and packing the vector with the specified columns

```
//////////////////////////////
// use a vector assembler to build features for
// logistic model
val assembler = new VectorAssembler()
  .setInputCols(featureAryString)
  .setOutputCol("features")

val trainingdf2  =  assembler.transform(trainingdf)
val testdf2      =  assembler.transform(testdf)
```

Feature Array passed in e.g.

```
Array("scoreb-scorea",  "teamaspread", "cf1", "cf2", "cf3","cf4" )
```

Logistic Regression Example



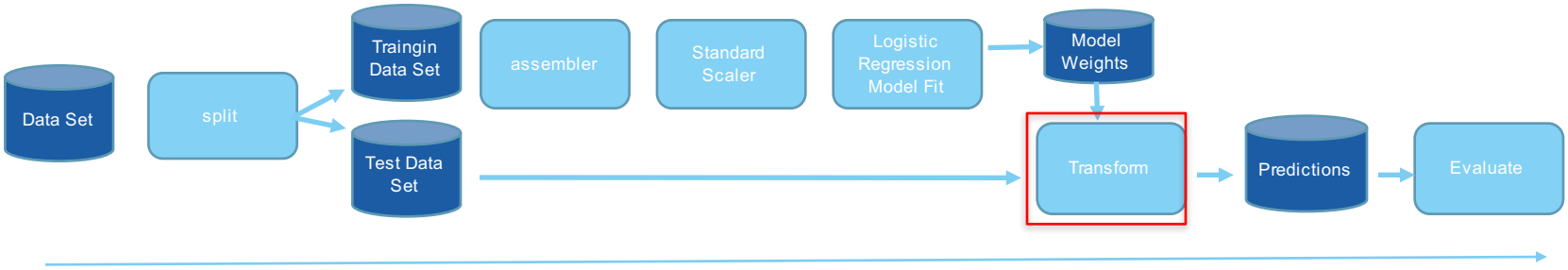
When fitting a model, there are some configuration options like regularization
For this example, I manually tuned this value. Using a cross validator, you can auto-mate this step
Whats key for this example, is extracting the intercept and weights of the model.

```
//////////////////////////////
//Logistic Regression Model Setup
// Setup some of the configurations for the Logistic regression model ..
val lr = new LogisticRegression()
  .setMaxIter(25)
  .setRegParam(0.0001)
  .setElasticNetParam(0.0)
  .setLabelCol("home-win")

// Fit the model
val lrModel    = lr.fit(trainingdf3)

println("Reg Parameter:    =" + lrModel.getRegParam)
println("lrModel.intercept = " + lrModel.intercept)
println("lrModel.weights   = " + lrModel.weights)
```

```
Tranining Samples = 9206
Test      Samples = 4011
Reg Parameter:    =1.0E-4
lrModel.intercept = 0.14919058350290829
lrModel.weights   = [-0.03417700123987033,0.120735775332254,0.23416816388919753,3.9997214418634804E-4]
```

These values are used to implement logistic regression in our web service

Logistic Regression Example



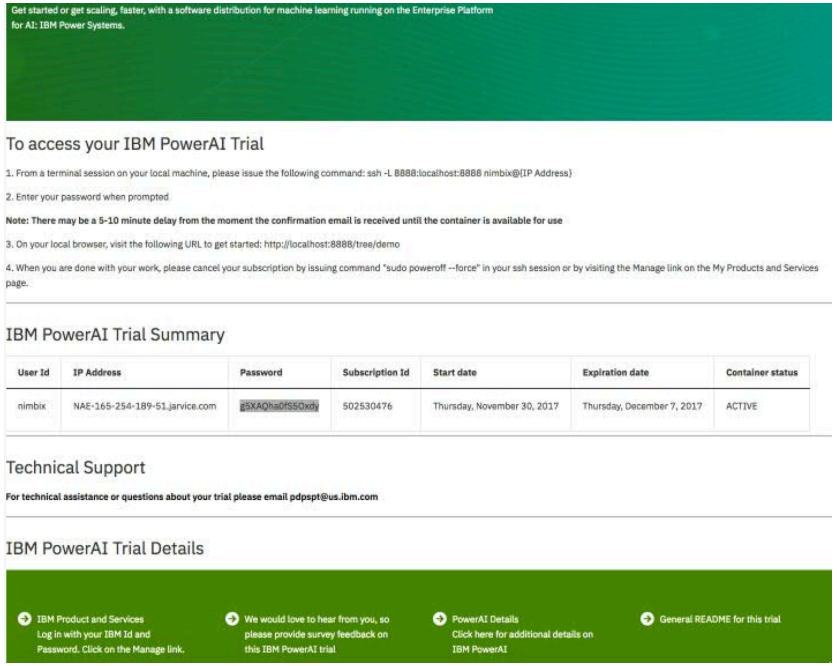
Transform – here we pass the test data frame into the model, and get a new dataframe with a probability/prediction column added

```
val predictions = lrModel.transform(testdf3)
```

```
+-----+------+-----+------+--------------+------------+-----------+-------+-------+--------------------+--------+----------+
|teama|scorea|teamb|scoreb|      timeleft|pct-comp-ceil|teamaspread|fscorea|fscoreb|         probability|home-win|prediction|
+-----+------+-----+------+--------------+------------+-----------+-------+-------+--------------------+--------+----------+
|  cha|    88|  tor|    93|0.433333333333|       100.0|        4.0|   90.0|   96.0|[0.00330369572074...|     1.0|       1.0|
|  lal|    55|  nor|    73| 18.2666666667|        62.0|        2.5|  102.0|  110.0|[0.04601940412997...|     1.0|       1.0|
|  dal|    55|  lac|    64| 16.9333333333|        65.0|        5.5|   91.0|   98.0|[0.10254208163509...|     1.0|       1.0|
|  chi|    55|  nor|    67| 22.6166666667|        53.0|       -4.5|  121.0|  116.0|[0.19810315747617...|     0.0|       1.0|
|  lal|    43|  nor|    42|          28.0|        42.0|        2.5|  102.0|  110.0|[0.42414007608531...|     1.0|       1.0|
+-----+------+-----+------+--------------+------------+-----------+-------+-------+--------------------+--------+----------+
```

Notebooks Demo
DSX and Nimbix

Free 7 day trial available on Nimbix!
https://developer.ibm.com/linuxonpower/cloud-resources/



Engage with us!

IBM has an excellent array of offerings on cloud (freemium, pay-go, monthly) and on premise!

- Data Scientist Experience (DSX)
  - IBM's Data Science platform
  - http://datascience.ibm.com/

Contact : vanstee@us.ibm.com
LinkedIn : https://www.linkedin.com/in/dustinvanstee/