Lecture 25 - Into Deep Learning

This is an R Markdown Notebook

Lecture 25 - Your project. Deep Learning with H2O. P.1 Install and explore by example

Deep Learning… In this lecture I would like to esplore with you the DeepLearning we can do with H2O package in R…

Learning to do Deep Learning

Used example from: (https://dzone.com/articles/anomaly-detection-with-deep-learning-in-r-with-h2o)[https://dzone.com/articles/anomaly-detection-with-deep-learning-in-r-with-h2o]

More reading: (https://dzone.com/articles/the-basics-of-deep-learning-how-to-apply-it-to-pre?fromrel=true)[https://dzone.com/articles/the-basics-of-deep-learning-how-to-apply-it-to-pre?fromrel=true]

And: (https://shiring.github.io/machine_learning/2017/05/01/fraud)[https://shiring.github.io/machine_learning/2017/05/01/fraud]

In this lecture we would explore the 'technology' on the sample and try to do this in 10 min lecture!

Understanding architecture

The 'Thing' will be working on your computer. In fact you will install another 'computer' inside your 'computer'…

Installing from R and starting it up

These instructions I got from: (http://h2o.ai/download/)[http://h2o.ai/download/]. Simply use 'Install from R' option

```r
# The following two commands will remove any previously installed H2O packages for R.
if ("package:h2o" %in% search()) { detach("package:h2o", unload=TRUE) }
if ("h2o" %in% rownames(installed.packages())) { remove.packages("h2o") }

# Next, we download packages that H2O depends on.
pkgs <- c("statmod","RCurl","jsonlite")
for (pkg in pkgs) {
if (! (pkg %in% rownames(installed.packages()))) { install.packages(pkg) }
}

# Now we download, install and initialize the H2O package for R.
install.packages("h2o", type="source", repos="http://h2o-release.s3.amazonaws.com/h2o/rel-weierstrass/7

# Finally, let's load H2O and start up an H2O cluster
library(h2o)
h2o.init()
```

As you see this will actually start the 'cluster' on our machine! We will now can look on `Localhost:54321` to see the 'interface' of our cluster…

As a trial let's learn how to shut down the machine…

```r
# we can shut down the 'machine' like this...
h2o.shutdown(prompt= FALSE)
```

In this case I have installed H2O Machine Learning Platform on my PC, but in the real world you may install it on more powerfull computer.

Explore the 'thing' on example

We will now run the demo from H2O. Our goal will be to teach system what is `normal` by using ECG dataset. After that we will use another dataset that will contain `anomaly` and use our Deep Learning model to detect that

Start Virtual Machine h2o

First thing we will launch the machine again...

```r
# to load the library
library(h2o)

# to initialize the 'machine'
h2o.init()
```

```
##
## H2O is not running yet, starting it now...
##
## Note:  In case of errors look at the following log files:
##     C:\Users\fxtrams\AppData\Local\Temp\RtmpmqBNE9/h2o_fxtrams_started_from_r.out
##     C:\Users\fxtrams\AppData\Local\Temp\RtmpmqBNE9/h2o_fxtrams_started_from_r.err
##
##
## Starting H2O JVM and connecting: . Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         2 seconds 170 milliseconds
##     H2O cluster version:        3.14.0.7
##     H2O cluster version age:    22 days
##     H2O cluster name:           H2O_started_from_R_fxtrams_bxj389
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.77 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  4
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Algos, AutoML, Core V3, Core V4
##     R Version:                  R version 3.2.5 (2016-04-14)
```

Load datasets from H2O

Then we will download the datasets from h2o.

```r
# Import ECG train and test data into the H2O cluster
train_ecg <- h2o.importFile(
 path = "http://h2o-public-test-data.s3.amazonaws.com/smalldata/anomaly/ecg_discord_train.csv",
 header = FALSE,
 sep = ",")
```

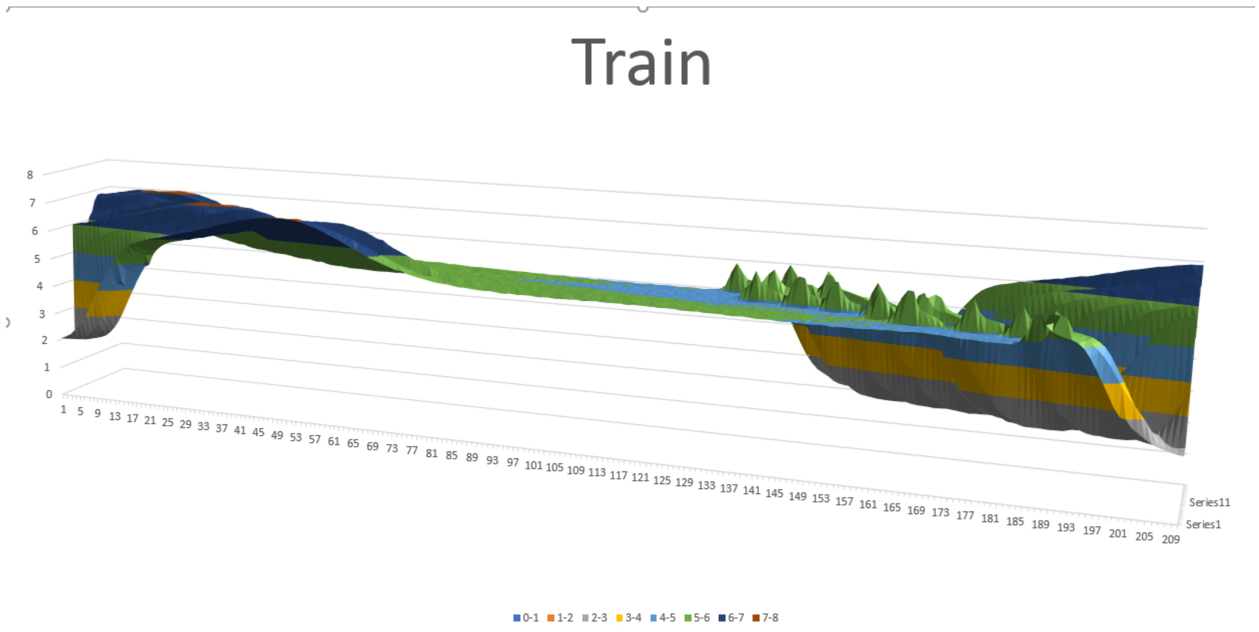
```
##
  |
  |                                                                  |   0%
  |
  |==================================================================| 100%
```

```r
test_ecg <- h2o.importFile(
 path = "http://h2o-public-test-data.s3.amazonaws.com/smalldata/anomaly/ecg_discord_test.csv",
 header = FALSE,
 sep = ",")
```

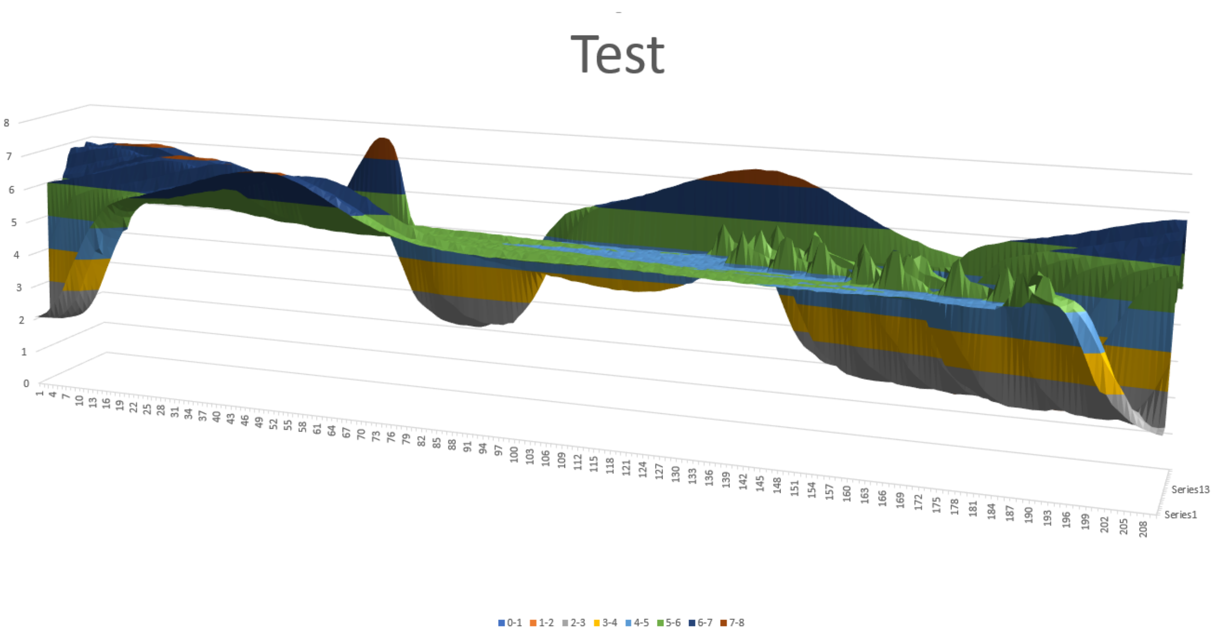
```
##
  |
  |                                                                  |   0%
  |
  |==================================================================| 100%
```

Understanding dataset

Personally I like to know what is in the data and how it look like!!! I will just use this link and put it into the browser. This will download the files with raw data. If I open this dataset it would not tell me much! I will plot this data in excel…



Train data set



Test data set

Or, I can also pull the data from h2o into R and make some 3D visualizations…

```
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr

## Conflicts with tidy packages ----------------------------------------------

## filter(): dplyr, stats
## lag():    dplyr, stats
```

```
# data frame matrix for training dataset
matrix_train <- train_ecg %>% as.data.frame() %>% as.matrix.data.frame()
# data frame matrix for test dataset
matrix_test <- test_ecg %>% as.data.frame() %>% as.matrix.data.frame()
```

from there we can see that the difference between both are in the three new rows 21-23

```
# using library plotly to plot 3D surface
library(plotly)
plot_ly(z = matrix_train, type = "surface")
```

```
# using library plotly to plot 3D surface
plot_ly(z = matrix_test, type = "surface")
```

Building Deep Learning model with autoencoder

Now, once we know how our data looks like we can start to do our Anomaly Model

```
# Train deep autoencoder learning model on "normal"
# training data, y ignored
anomaly_model <- h2o.deeplearning(
```

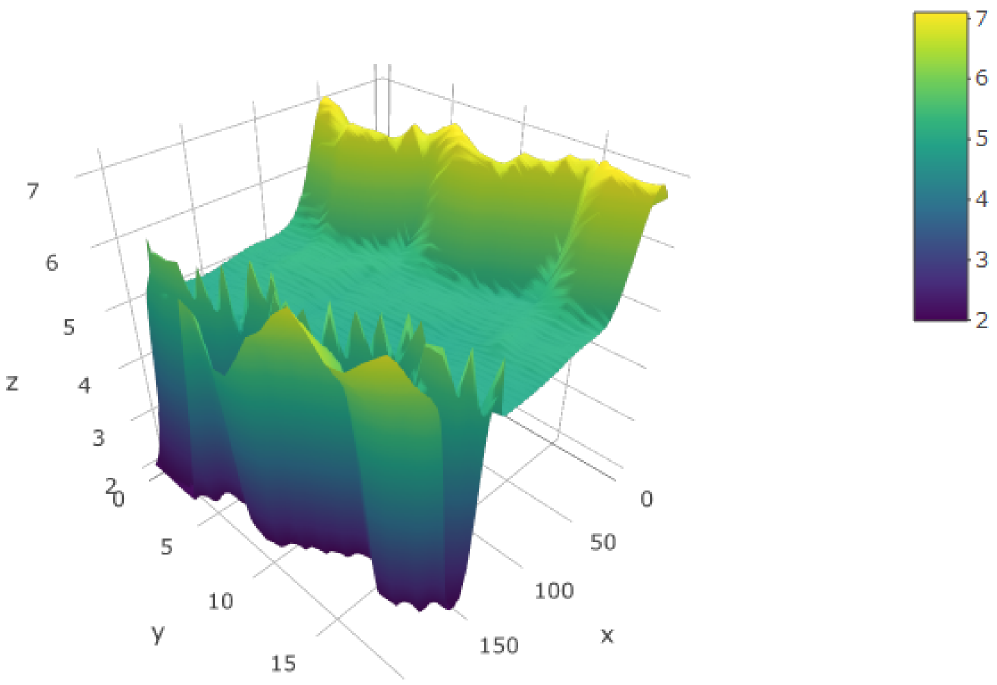


Figure 1: Train Dataset with plotly

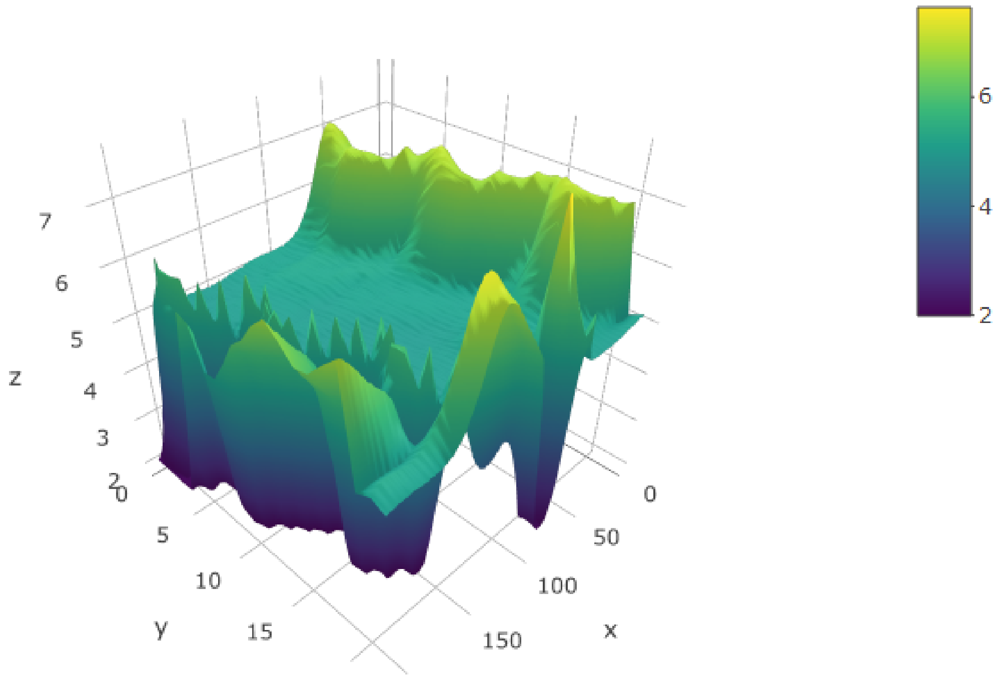


Figure 2: Test Dataset with plotly

```
  x = names(train_ecg),
  training_frame = train_ecg,
  activation = "Tanh",
  autoencoder = TRUE,
  hidden = c(50,20,50),
  sparse = TRUE,
  l1 = 1e-4,
  epochs = 100)
```

```
##
  |
  |                                                                      |   0%
  |
  |===============================================================       |  90%
  |
  |======================================================================| 100%
```

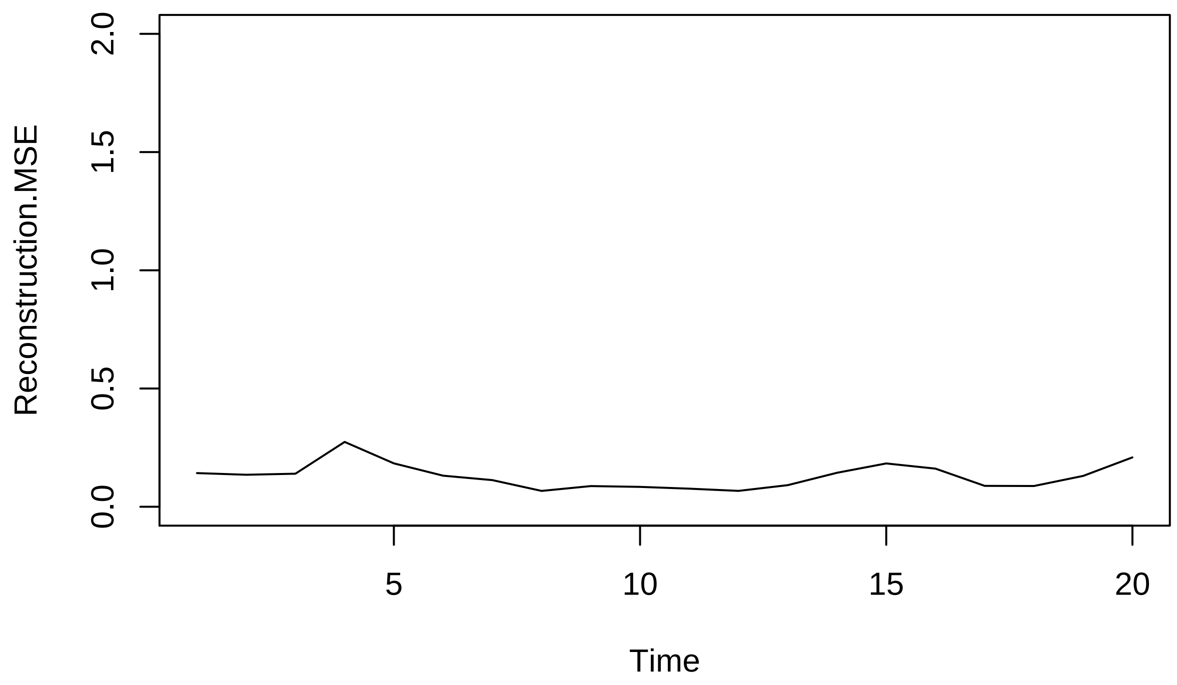
Calculate MSE from the train dataset

Let's use this model on our training dataset…

```
# computer error of the model
mod_error <- h2o.anomaly(anomaly_model, train_ecg)
```

get it as a plot and see that the value is very low

```
# visually see it
h2o.anomaly(anomaly_model, train_ecg) %>%
  as.data.frame() %>% plot.ts(ylim = c(0, 2))
```



Detect Anomaly using MSE value

Once our model is made, we can use it to detect anomalies in our **test** dataset

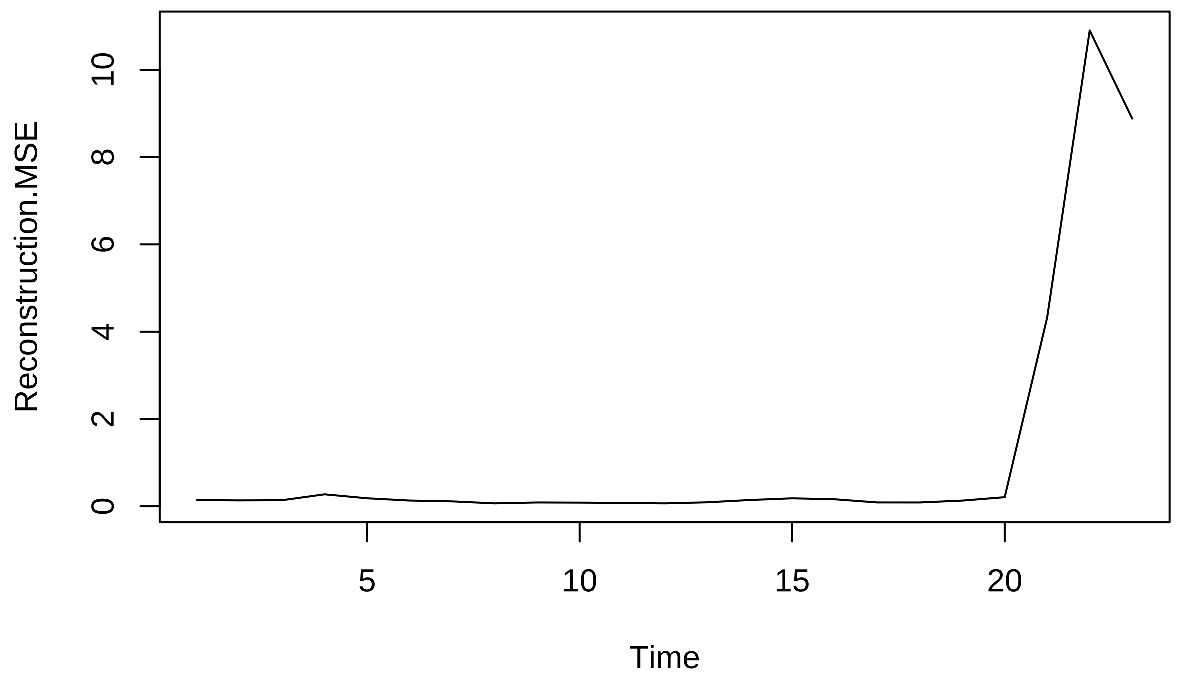
```
# Compute reconstruction error with the Anomaly
# detection app (MSE between output and input layers)
recon_error <- h2o.anomaly(anomaly_model, test_ecg)

# Pull reconstruction error data into R and
# plot to find outliers (last 3 heartbeats)
df_recon_error <- as.data.frame(recon_error)
tail(df_recon_error, 9)
```

```
##    Reconstruction.MSE
## 15         0.18308837
## 16         0.16113917
## 17         0.08831310
## 18         0.08771082
## 19         0.13044050
## 20         0.20868673
## 21         4.32765892
## 22        10.90042969
## 23         8.88141754
```

We can plot this as well

```
plot.ts(df_recon_error)
```



What it is telling us is that based on the new data we have anomaly in elements 21, 22, 23

Use Anomaly model to reconstruct Test Dataset

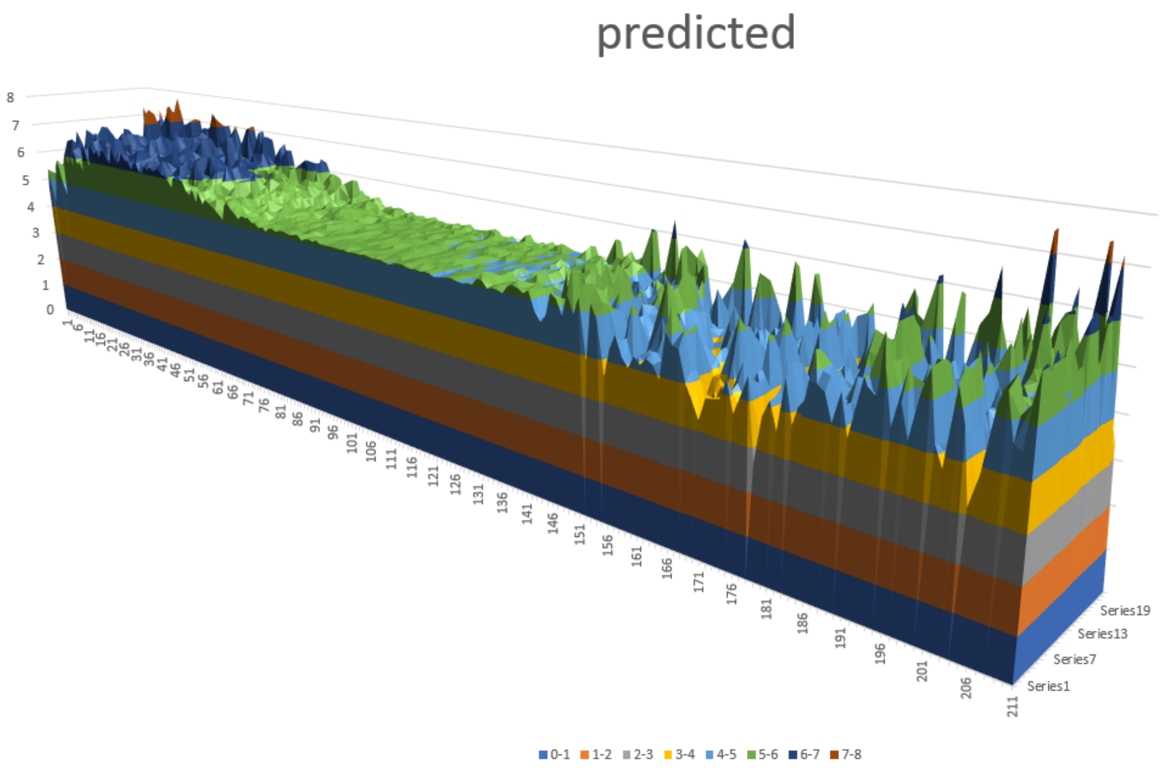
Now we can obtain predictions, or physical values using our model. We should provide the model and test dataset

```
# Note: Testing = Reconstructing the test dataset
test_recon <- h2o.predict(anomaly_model, test_ecg)
```

```
##
  |
  |                                                                 |   0%
  |
  |=================================================================| 100%
```

In order to make things visible. Once again I ask excel to help… Here I simply write teh dataframe to csv and create graph in excel

```
# write to csv to use it in excel
test_recon %>% as.data.frame() %>%
write.csv("test_predicted.csv")
```



Predicted data set

Visualize as 3D

Or we can visualize in 3D directly in R

```
# making a matrix dataframe
recon_matrix <- test_recon %>% as.data.frame() %>% as.matrix.data.frame()

# make 3D plot
plot_ly(z = recon_matrix, type = "surface")
```

Saving Deep Learning Model for the future use

To use our model in our ShinyApp we will save it…

```
h2o.saveModel(anomaly_model, "C:/Users/fxtrams/Downloads/tmp/anomaly_model.bin")
h2o.download_pojo(anomaly_model, "C:/Users/fxtrams/Downloads/tmp", get_jar = TRUE)
```

And let's not forget to switch off our cluster!

```
h2o.shutdown(prompt= FALSE)
```

```
## [1] TRUE
```

Conclusion

In this example the Anomaly Detection model was able to output the anomaly in rows 21-23.

It learned on the pattern of many vectors and was able to distinguish the anomaly coming on new dataset

Practical use of this model can be to us function **h2o.anomaly**. In case the MSE value will be high - the anomaly is detected!

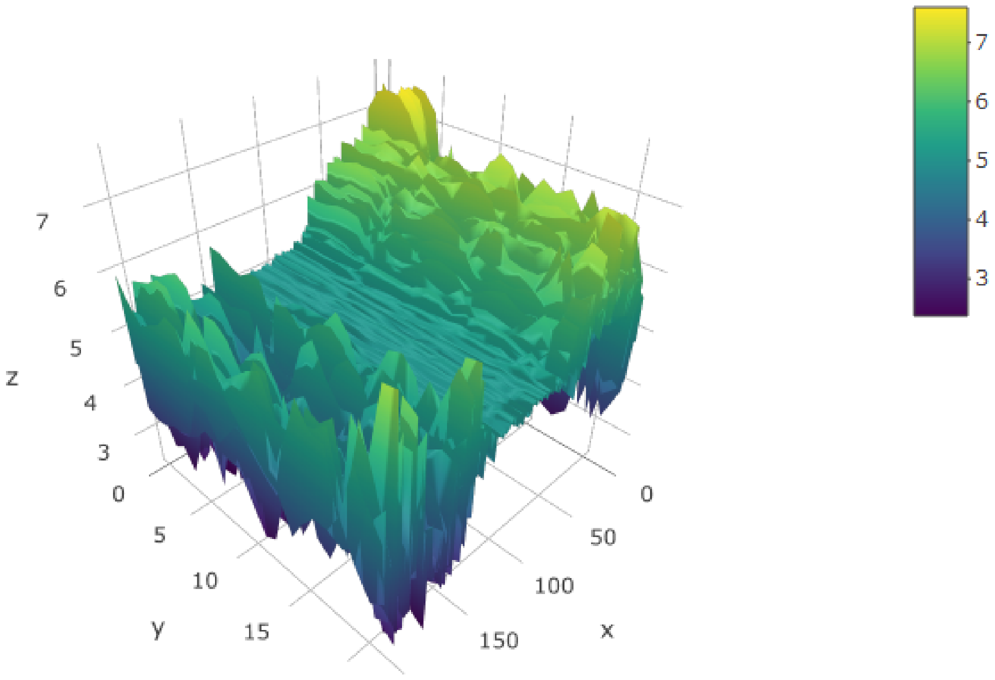


Figure 3: Predicted Dataset with plotly

Next step

our next step will be to repeat the procedure but on our machine data.

- re-arranging data as matrix
- fitting deep learning models
- testing the models
- saving models
- implementation in our ShinyApp…