Deep learning helps in identifying true positive cases which are essential in finding abnormalities in medical diagnostic problems, fraud analytics, and attacks in communication networks. It is critical to predict accurately and isolate true positive cases, compared to true negative cases. It has been shown here one such example in finding attacks in a communication network. The data is taken from <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>. Deep Autoencoder is used as the deep neural network for the implementation of the problem .

There are 60839 rows and 43 columns. All columns are of numeric data type.

The id column is removed as it does not influence the results.

There are two levels ’yes’ and ‘no’ in the outlier column, which are

converted to numeric type of data.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| > trial<-read.arff("D:/literature1/literature/KDDCup99/KDDCup99\_norm\_idf.arff")  > dim(trial)  [1] 60839 43  str(trial)  'data.frame': 60839 obs. of 43 variables:  $ id : num 1 2 3 7 8 10 12 13 15 16 ...  $ duration : num 0 0 0 0 0 ...  $ protocol\_type : num 0.222 0.222 0.222 0.222 0.222 ...  $ service : num 0.105 0.105 0.105 0.238 0.105 ...  $ flag : num 0 0 0 0 0 0 0 0 0 0 ...  $ src\_bytes : num 1.67e-05 1.67e-05 1.67e-05 4.61e-06 1.67e-05 ...  $ dst\_bytes : num 2.81e-05 2.81e-05 2.81e-05 0.00 2.81e-05 ...  $ land : num 0 0 0 0 0 0 0 0 0 0 ...  $ wrong\_fragment : num 0 0 0 0 0 0 0 0 0 0 ...  $ urgent : num 0 0 0 0 0 0 0 0 0 0 ...  $ hot : num 0 0 0 0 0 0 0 0 0 0 ...  $ num\_failed\_logins : num 0 0 0 0 0 0 0 0 0 0 ...  $ logged\_in : num 0 0 0 0 0 1 1 0 1 1 ...  $ num\_compromissed : num 0 0 0 0 0 0 0 0 0 0 ...  $ root\_shell : num 0 0 0 0 0 0 0 0 0 0 ...  $ su\_attempted : num 0 0 0 0 0 0 0 0 0 0 ...  $ num\_root : num 0 0 0 0 0 0 0 0 0 0 ...  $ num\_file\_creations : num 0 0 0 0 0 0 0 0 0 0 ...  $ num\_shells : num 0 0 0 0 0 0 0 0 0 0 ...  $ num\_access\_files : num 0 0 0 0 0 0 0 0 0 0 ...  $ num\_outbounds\_cmds : num 0 0 0 0 0 0 0 0 0 0 ...  $ is\_host\_login : num 0 0 0 0 0 0 0 0 0 0 ...  $ is\_guest\_login : num 0 0 0 0 0 0 0 0 0 0 ...  $ count : num 0.00196 0.00196 0.00196 0.00391 0.00196 ...  $ srv\_count : num 0.00196 0.00196 0.00196 0.00196 0.00196 ...  $ serror\_rate : num 0 0 0 0 0 0 0 0 0 0 ...  $ srv\_serror\_rate : num 0 0 0 0 0 0 0 0 0 0 ...  $ rerror\_rate : num 0 0 0 0 0 0 0 0 0 0 ...  $ srv\_rerror\_rate : num 0 0 0 0 0 0 0 0 0 0 ...  $ same\_srv\_rate : num 1 1 1 0.5 1 1 1 1 1 1 ...  $ diff\_srv\_rate : num 0 0 0 1 0 0 0 0 0 0 ...  $ srv\_diff\_host\_rate : num 0 0 0 0 0 0 0.11 0 1 0 ...  $ dst\_host\_count : num 1 1 1 0.0392 1 ...  $ dst\_host\_srv\_count : num 0.9961 0.9961 0.9961 0.0118 0.9922 ...  $ dst\_host\_same\_srv\_rate : num 1 1 1 0.3 0.99 1 1 1 0.72 1 ...  $ dst\_host\_diff\_srv\_rate : num 0.01 0.01 0.01 0.3 0.01 0 0 0.01 0.11 0 ...  $ dst\_host\_same\_src\_port\_rate: num 0 0 0 0.3 0 0.01 0.33 0.01 0.02 0.01 ...  $ dst\_host\_srv\_diff\_host\_rate: num 0 0 0 0 0 0.01 0.07 0 0 0.01 ...  $ dst\_host\_serror\_rate : num 0 0 0 0 0 0 0.33 0 0.02 0 ...  $ dst\_host\_srv\_serror\_rate : num 0 0 0 0 0 0 0 0 0 0 ...  $ dst\_host\_rerror\_rate : num 0 0 0 0 0 0 0 0 0.09 0 ...  $ dst\_host\_srv\_rerror\_rate : num 0 0 0 0 0 0 0 0 0.13 0 ...  $ outlier : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...  > trial %>%  + ggplot(aes(x = outlier)) +  + geom\_bar(color = "blue", fill = "lightgrey") +  + theme\_bw()  >  From the figure it can be shown that the positive examples are only a fraction of the entire examples.  trial1<-trial[,-1]  > trial1$outlier<-as.numeric(trial1$outlier)  > str(trial1$outlier)  num [1:60839] 1 1 1 1 1 1 1 1 1 1 ...   |  | | --- | | > trial1$outlier[trial1$outlier==1]<-0  > trial1$outlier[trial1$outlier==2]<-1  > str(trial1$outlier)  num [1:60839] 0 0 0 0 0 0 0 0 0 0 ...  > trial1$outlier<-as.factor(trial1$outlier)  > str(trial1$outlier)  Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  > head(trial1$outlier)  [1] 0 0 0 0 0 0  Levels: 0 1  > table(trial1$outlier)  0 1  60593 246 | |  | | |  | | --- | | #Initialize the h2o library which is for the implementation of deep learning neural networks. | | | > h2o.init()  H2O is not running yet, starting it now...  Note: In case of errors look at the following log files:  C:\Users\malat\AppData\Local\Temp\RtmpMBiIxT/h2o\_malat\_started\_from\_r.out  C:\Users\malat\AppData\Local\Temp\RtmpMBiIxT/h2o\_malat\_started\_from\_r.err  java version "1.8.0\_71"  Java(TM) SE Runtime Environment (build 1.8.0\_71-b15)  Java HotSpot(TM) 64-Bit Server VM (build 25.71-b15, mixed mode)  Starting H2O JVM and connecting: ...... Connection successful!  R is connected to the H2O cluster:  H2O cluster uptime: 20 seconds 345 milliseconds  H2O cluster version: 3.10.3.6  H2O cluster version age: 6 months and 19 days !!!  H2O cluster name: H2O\_started\_from\_R\_malat\_prl645  H2O cluster total nodes: 1  H2O cluster total memory: 0.87 GB  H2O cluster total cores: 2  H2O cluster allowed cores: 2  H2O cluster healthy: TRUE  H2O Connection ip: localhost  H2O Connection port: 54321  H2O Connection proxy: NA  R Version: R version 3.3.2 (2016-10-31)  Note: As started, H2O is limited to the CRAN default of 2 CPUs.  Shut down and restart H2O as shown below to use all your CPUs.  > h2o.shutdown()  > h2o.init(nthreads = -1)  Warning message:  In h2o.clusterInfo() :  Your H2O cluster version is too old (6 months and 19 days)!  Please download and install the latest version from http://h2o.ai/download/  The data frame is converted to h2o data frame. | |  | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | | trial1\_hf <- as.h2o(trial1)  |============================================================================================================================| 100%  The data is split in the ratios 60% (unsupervised training)  20% (supervised training) and the remaining percentage as test set.  > splits <- h2o.splitFrame(trial1\_hf,  + ratios = c(0.6, 0.2),  + seed = 42)  > train\_unsupervised <- splits[[1]]  > train\_supervised <- splits[[2]]  > test <- splits[[3]]  To implement One class classification, | |  | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | > # move class 1 instances to second training set...  > train\_supervised <- rbind(as.data.frame(train\_supervised), as.data.frame(train\_unsupervised[train\_unsupervised$outlier == "1", ])) %>%  + as.h2o()  |============================================================================================================================| 100%  >  > # ... and remove from first training set  > train\_unsupervised <- train\_unsupervised[train\_unsupervised$outlier == "0", ]  > NROW(train\_unsupervised)  [1] 36332  > NROW(train\_supervised)  [1] 12342  > NROW(test)  [1] 12165   |  | | --- | | > sum(train\_supervised=='0')  [1] 12150  > sum(train\_supervised=='1')  [1] 192  > sum(train\_unsupervised=='1')  [1] 0  > sum(train\_unsupervised=='0')  [1] 36332  > sum(test=='0')  [1] 12111  > sum(test=='1')  [1] 54 | |  | | |  | | --- | |  | |   The outlier is the dependent variable that depends on all 41 features. Outlier is stored as response variable, for convenience. There are 5 layers int his autoencoder neural network. The structure of the hidden layer is 20, 4, 20. So this structure says 4 nodes in the middle layer 20 nodes in the second and fourth layer. The first layer and fifth one has 41 nodes which are the features in the problem. | | > response <- "outlier"  > features <- setdiff(colnames(train\_unsupervised), response)  > str(features)  chr [1:41] "duration" "protocol\_type" "service" "flag" "src\_bytes"  "dst\_bytes" "land" "wrong\_fragment" ...   |  | | --- | | > model\_nn <- h2o.deeplearning(x = features, training\_frame = train\_unsupervised,  + model\_id = "model\_nn",  + autoencoder = TRUE,  + reproducible = TRUE, #slow - turn off for real problems  + ignore\_const\_cols = FALSE,  + seed = 42,  + hidden = c(20, 4, 20),  + epochs = 100,  + activation = "Tanh")  |============================================================================================================================| 100% | |  | | > summary(model\_nn)  Model Details:  ==============  H2OAutoEncoderModel: deeplearning  Model Key: model\_nn  Status of Neuron Layers: auto-encoder, gaussian distribution, Quadratic loss, 1,885 weights/biases, 31.3 KB, 1,562,276 training samples, mini-batch size 1  layer units type dropout l1 l2 mean\_rate rate\_rms momentum mean\_weight weight\_rms mean\_bias bias\_rms  1 1 41 Input 0.00 %  2 2 20 Tanh 0.00 % 0.000000 0.000000 0.600504 0.424541 0.000000 0.010328 0.378554 -0.194525 1.507428  3 3 4 Tanh 0.00 % 0.000000 0.000000 0.005045 0.002342 0.000000 0.032318 0.299479 0.009820 0.346375  4 4 20 Tanh 0.00 % 0.000000 0.000000 0.011280 0.008331 0.000000 -0.078779 0.987728 -0.200818 0.885592  5 5 41 Tanh 0.000000 0.000000 0.062106 0.071803 0.000000 0.003673 0.253734 0.021328 0.389590  H2OAutoEncoderMetrics: deeplearning  \*\* Reported on training data. \*\*  Training Set Metrics:  =====================  MSE: (Extract with `h2o.mse`) 0.0006185575  RMSE: (Extract with `h2o.rmse`) 0.02487082  Scoring History:  timestamp duration training\_speed epochs iterations samples training\_rmse training\_mse  1 2017-09-10 12:40:01 2.259 sec 0.00000 obs/sec 0.00000 0 0.000000 0.15181 0.02305  2 2017-09-10 12:40:09 9.512 sec 5172 obs/sec 1.00000 1 36332.000000 0.04360 0.00190  3 2017-09-10 12:40:17 17.207 sec 5030 obs/sec 2.00000 2 72664.000000 0.03825 0.00146  4 2017-09-10 12:40:25 25.139 sec 4931 obs/sec 3.00000 3 108996.000000 0.03960 0.00157  5 2017-09-10 12:40:31 31.981 sec 5059 obs/sec 4.00000 4 145328.000000 0.03307 0.00109  ---  timestamp duration training\_speed epochs iterations samples training\_rmse training\_mse  40 2017-09-10 12:44:31 4 min 31.238 sec 5476 obs/sec 39.00000 39 1416948.000000 0.02611 0.00068  41 2017-09-10 12:44:37 4 min 37.564 sec 5487 obs/sec 40.00000 40 1453280.000000 0.03207 0.00103  42 2017-09-10 12:44:43 4 min 43.745 sec 5500 obs/sec 41.00000 41 1489612.000000 0.02566 0.00066  43 2017-09-10 12:44:49 4 min 49.976 sec 5512 obs/sec 42.00000 42 1525944.000000 0.02487 0.00062  44 2017-09-10 12:44:56 4 min 56.225 sec 5523 obs/sec 43.00000 43 1562276.000000 0.02512 0.00063  45 2017-09-10 12:44:56 4 min 56.474 sec 5522 obs/sec 43.00000 43 1562276.000000 0.02487 0.00062 | |  | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | > > model\_nn\_2 <- h2o.deeplearning(y = response,  + x = features,  + training\_frame = train\_supervised,  + pretrained\_autoencoder = "model\_nn",  + reproducible = TRUE, #slow - turn off for real problems  + balance\_classes = TRUE,  + ignore\_const\_cols = FALSE,  + seed = 42,  + hidden = c(20, 4, 20),  + epochs = 100,  + activation = "Tanh")  |===================================================================  ==========================================================================| 100%   |  | | --- | | > summary(model\_nn\_2)  Model Details:  ==============  H2OBinomialModel: deeplearning  Model Key: DeepLearning\_model\_R\_1505023176513\_1  Status of Neuron Layers: predicting outlier, 2-class classification, bernoulli distribution, CrossEntropy loss, 1,066 weights/biases, 22.7 KB, 534,666 training samples, mini-batch size 1  layer units type dropout l1 l2 mean\_rate rate\_rms momentum mean\_weight weight\_rms mean\_bias bias\_rms  1 1 41 Input 0.00 %  2 2 20 Tanh 0.00 % 0.000000 0.000000 0.565703 0.397267 0.000000 0.012117 0.463028 -0.206532 1.552544  3 3 4 Tanh 0.00 % 0.000000 0.000000 0.005055 0.005185 0.000000 -0.031622 0.312858 -0.345607 0.410989  4 4 20 Tanh 0.00 % 0.000000 0.000000 0.104177 0.229819 0.000000 -0.119495 1.111441 0.080546 0.899968  5 5 2 Softmax 0.000000 0.000000 0.013429 0.005842 0.000000 0.037002 1.380809 -0.000000 0.692770  H2OBinomialMetrics: deeplearning  \*\* Reported on training data. \*\*  \*\* Metrics reported on temporary training frame with 10026 samples \*\*  MSE: 0.01199219  RMSE: 0.1095089  LogLoss: 0.05150088  Mean Per-Class Error: 0.003898441  AUC: 0.9990908  Gini: 0.9981816  Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:  0 1 Error Rate  0 4963 39 0.007797 =39/5002  1 0 5024 0.000000 =0/5024  Totals 4963 5063 0.003890 =39/10026  Maximum Metrics: Maximum metrics at their respective thresholds  metric threshold value idx  1 max f1 0.180478 0.996134 180  2 max f2 0.180478 0.998450 180  3 max f0point5 0.216049 0.994038 172  4 max accuracy 0.180478 0.996110 180  5 max precision 0.997144 1.000000 0  6 max recall 0.180478 1.000000 180  7 max specificity 0.997144 1.000000 0  8 max absolute\_mcc 0.180478 0.992250 180  9 max min\_per\_class\_accuracy 0.216049 0.993603 172  10 max mean\_per\_class\_accuracy 0.180478 0.996102 180  Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`  Scoring History:  timestamp duration training\_speed epochs iterations samples training\_rmse training\_logloss training\_auc training\_lift  1 2017-09-10 12:51:30 0.000 sec 0.00000 0 0.000000  2 2017-09-10 12:51:33 4.481 sec 10207 obs/sec 1.00000 1 24303.000000 0.29781 0.28256 0.99629 1.97913  3 2017-09-10 12:51:36 7.126 sec 10486 obs/sec 2.00000 2 48606.000000 0.20913 0.16474 0.99746 1.99562  4 2017-09-10 12:51:39 9.782 sec 10516 obs/sec 3.00000 3 72909.000000 0.22275 0.18737 0.99802 1.97966  5 2017-09-10 12:51:41 12.503 sec 10588 obs/sec 4.00000 4 97212.000000 0.18348 0.12794 0.99870 1.99562  training\_classification\_error  1  2 0.01666  3 0.00858  4 0.00768  5 0.00658  ---  timestamp duration training\_speed epochs iterations samples training\_rmse training\_logloss training\_auc training\_lift  19 2017-09-10 12:52:16 47.484 sec 11116 obs/sec 18.00000 18 437454.000000 0.22658 0.17173 0.99916 1.99562  20 2017-09-10 12:52:19 49.880 sec 11151 obs/sec 19.00000 19 461757.000000 0.12099 0.05757 0.99914 1.99562  21 2017-09-10 12:52:21 52.306 sec 11184 obs/sec 20.00000 20 486060.000000 0.23030 0.18133 0.99876 1.99562  22 2017-09-10 12:52:24 54.708 sec 11213 obs/sec 21.00000 21 510363.000000 0.17102 0.10889 0.99859 1.99562  23 2017-09-10 12:52:26 57.102 sec 11245 obs/sec 22.00000 22 534666.000000 0.19094 0.13159 0.99876 1.97748  24 2017-09-10 12:52:26 57.456 sec 11240 obs/sec 22.00000 22 534666.000000 0.10951 0.05150 0.99909 1.99562  training\_classification\_error  19 0.00349  20 0.00439  21 0.00379  22 0.00369  23 0.00419  24 0.00389 | |  | | |  | | --- | |  | |   pred <- as.data.frame(h2o.predict(object = model\_nn\_2, newdata = test)) %>%  + mutate(actual = as.vector(test[, 42]))  |=============================================================================================================================================| 100%  > pred %>%  + group\_by(actual, predict) %>%  + summarise(n = n()) %>%  + mutate(freq = n / sum(n))  Source: local data frame [4 x 4]  Groups: actual [2]  actual predict n freq  <chr> <fctr> <int> <dbl>  1 0 0 11975 0.98877054  2 0 1 136 0.01122946  3 1 0 1 0.01851852  4 1 1 53 0.98148148  The result shows out of 54 true positive cases 53 are correctly classified which is pretty impressive.  The data set division is 40,40,20, and the hidden layer structure is 15,3,15. 51 true positive cases are isolated among 54.  actual predict n freq  <chr> <fctr> <int> <dbl>  1 0 0 11978 0.98901825  2 0 1 133 0.01098175  3 1 0 3 0.05555556  4 1 1 51 0.94444444  The data set division is 40,40,20, and the hidden layer structure is 10,2,10.49 true positive cases are isolated among 54.  ac*tual predict n freq*  *<chr> <*fctr> <int> <dbl>  1 0 0 11479 0.94781604  2 0 1 632 0.05218396  3 1 0 5 0.09259259  4 1 1 49 0.907407   |  | | --- | | The data is divided in to 60% as training, 10% as validation and 30% as testing set.  Among the 79 true positive cases, 73 are isolated.  trial1\_hf <- as.h2o(trial1)  |=============================================================================================================================================| 100%  > splits <- h2o.splitFrame(trial1\_hf,  + ratios = c(0.6, 0.1),  + seed = 42)  > train\_unsupervised <- splits[[1]]  > train\_supervised <- splits[[2]]  > test <- splits[[3]]  > train\_supervised <- rbind(as.data.frame(train\_supervised), as.data.frame(train\_unsupervised[train\_unsupervised$outlier == "1", ])) %>%  + as.h2o()  |=============================================================================================================================================| 100%  >  > train\_unsupervised <- train\_unsupervised[train\_unsupervised$outlier == "0", ]  > sum(train\_supervised=='0')  [1] 6098  > nrow(train\_supervised)  [1] 6265  > nrow(train\_unsupervised)  [1] 36332  > nrow(train\_unsupervised=='0')  [1] 36332  > nrow(train\_unsupervised=='1')  [1] 36332  > sum(train\_unsupervised=='1')  [1] 0  > sum(train\_unsupervised=='0')  [1] 36332  > nrow(test)  [1] 18242  > sum(test=='0')  [1] 18163  > sum(test=='1')  [1] 79  > sum(train\_supervised=='1')  [1] 167 | |  | | |  | | --- | | > | | | response <- "outlier"  > features <- setdiff(colnames(train\_unsupervised), response)  >  >  > dim(features)  NULL  > model\_nn <- h2o.deeplearning(x = features,training\_frame = train\_unsupervised,  + model\_id = "model\_nn",  + autoencoder = TRUE,  + reproducible = TRUE, #slow - turn off for real problems  +  ignore\_const\_cols = FALSE,  + seed = 42,  + hidden = c(20, 4, 20),  + epochs = 100,  + activation = "Tanh")  |=============================================================================================================================================| 100%  >  summary(model\_nn)  Model Details:  ==============  H2OAutoEncoderModel: deeplearning  Model Key: model\_nn  Status of Neuron Layers: auto-encoder, gaussian distribution, Quadratic loss, 1,885 weights/biases, 31.3 KB, 1,562,276 training samples, mini-batch size 1  layer units type dropout l1 l2 mean\_rate rate\_rms momentum mean\_weight weight\_rms mean\_bias bias\_rms  1 1 41 Input 0.00 %  2 2 20 Tanh 0.00 % 0.000000 0.000000 0.600504 0.424541 0.000000 0.010328 0.378554 -0.194525 1.507428  3 3 4 Tanh 0.00 % 0.000000 0.000000 0.005045 0.002342 0.000000 0.032318 0.299479 0.009820 0.346375  4 4 20 Tanh 0.00 % 0.000000 0.000000 0.011280 0.008331 0.000000 -0.078779 0.987728 -0.200818 0.885592  5 5 41 Tanh 0.000000 0.000000 0.062106 0.071803 0.000000 0.003673 0.253734 0.021328 0.389590  H2OAutoEncoderMetrics: deeplearning  \*\* Reported on training data. \*\*  Training Set Metrics:  =====================  MSE: (Extract with `h2o.mse`) 0.0006185575  RMSE: (Extract with `h2o.rmse`) 0.02487082  Scoring History:  timestamp duration training\_speed epochs iterations samples training\_rmse training\_mse  1 2017-09-10 16:53:23 0.805 sec 0.00000 obs/sec 0.00000 0 0.000000 0.15181 0.02305  2 2017-09-10 16:53:29 7.082 sec 5991 obs/sec 1.00000 1 36332.000000 0.04360 0.00190  3 2017-09-10 16:53:35 13.510 sec 5929 obs/sec 2.00000 2 72664.000000 0.03825 0.00146  4 2017-09-10 16:53:42 19.918 sec 5928 obs/sec 3.00000 3 108996.000000 0.03960 0.00157  5 2017-09-10 16:53:48 26.320 sec 5916 obs/sec 4.00000 4 145328.000000 0.03307 0.00109  ---  timestamp duration training\_speed epochs iterations samples training\_rmse training\_mse  40 2017-09-10 16:57:31 4 min 8.799 sec 5961 obs/sec 39.00000 39 1416948.000000 0.02611 0.00068  41 2017-09-10 16:57:37 4 min 15.085 sec 5962 obs/sec 40.00000 40 1453280.000000 0.03207 0.00103  42 2017-09-10 16:57:43 4 min 21.513 sec 5959 obs/sec 41.00000 41 1489612.000000 0.02566 0.00066  43 2017-09-10 16:57:50 4 min 27.692 sec 5962 obs/sec 42.00000 42 1525944.000000 0.02487 0.00062  44 2017-09-10 16:57:56 4 min 34.005 sec 5965 obs/sec 43.00000 43 1562276.000000 0.02512 0.00063  45 2017-09-10 16:57:56 4 min 34.288 sec 5964 obs/sec 43.00000 43 1562276.000000 0.02487 0.00062 | |  | | |  | | --- | | > | | | > model\_nn\_2 <- h2o.deeplearning(y = response, x = features,  + training\_frame = train\_supervised,  + pretrained\_autoencoder = "model\_nn",  + reproducible = TRUE, #slow - turn off for real problems  + balance\_classes = TRUE,  + ignore\_const\_cols = FALSE,  + seed = 42,  + hidden = c(20, 4, 20),  + epochs = 100,  + activation = "Tanh")  |=============================================================================================================================================| 100% | |  | | |  | | --- | | > > pred <- as.data.frame(h2o.predict(object = model\_nn\_2, newdata = test)) %>%  + mutate(actual = as.vector(test[, 42]))  |=============================================================================================================================================| 100% | | | | | | |  | | --- | |  | | | > pred %>%  + group\_by(actual, predict) %>%  + summarise(n = n()) %>%  + mutate(freq = n / sum(n))  Source: local data frame [4 x 4]  Groups: actual [2]  actual predict n freq  <chr> <fctr> <int> <dbl>  1 0 0 18037 0.99306282  2 0 1 126 0.00693718  3 1 0 6 0.07594937  4 1 1 73 0.92405063 | |  | |  | | | | | |
|  |
| |  | | --- | |  | |