OCR letter analysis

Lei Deng, STAT 6620, ti7597

May 25, 2017

## Step 1 - Colleting data

Read the data to the R

# load the necessary library  
library(kernlab)  
library(h2o)

##   
## ----------------------------------------------------------------------  
##   
## Your next step is to start H2O:  
## > h2o.init()  
##   
## For H2O package documentation, ask for help:  
## > ??h2o  
##   
## After starting H2O, you can use the Web UI at http://localhost:54321  
## For more information visit http://docs.h2o.ai  
##   
## ----------------------------------------------------------------------

##   
## Attaching package: 'h2o'

## The following objects are masked from 'package:stats':  
##   
## cor, sd, var

## The following objects are masked from 'package:base':  
##   
## %\*%, %in%, &&, ||, apply, as.factor, as.numeric, colnames,  
## colnames<-, ifelse, is.character, is.factor, is.numeric, log,  
## log10, log1p, log2, round, signif, trunc

letters <- read.csv("letterdata.csv")

## Step 2 - Exploring and preparing the data

str(letters)

## 'data.frame': 20000 obs. of 17 variables:  
## $ letter: Factor w/ 26 levels "A","B","C","D",..: 20 9 4 14 7 19 2 1 10 13 ...  
## $ xbox : int 2 5 4 7 2 4 4 1 2 11 ...  
## $ ybox : int 8 12 11 11 1 11 2 1 2 15 ...  
## $ width : int 3 3 6 6 3 5 5 3 4 13 ...  
## $ height: int 5 7 8 6 1 8 4 2 4 9 ...  
## $ onpix : int 1 2 6 3 1 3 4 1 2 7 ...  
## $ xbar : int 8 10 10 5 8 8 8 8 10 13 ...  
## $ ybar : int 13 5 6 9 6 8 7 2 6 2 ...  
## $ x2bar : int 0 5 2 4 6 6 6 2 2 6 ...  
## $ y2bar : int 6 4 6 6 6 9 6 2 6 2 ...  
## $ xybar : int 6 13 10 4 6 5 7 8 12 12 ...  
## $ x2ybar: int 10 3 3 4 5 6 6 2 4 1 ...  
## $ xy2bar: int 8 9 7 10 9 6 6 8 8 9 ...  
## $ xedge : int 0 2 3 6 1 0 2 1 1 8 ...  
## $ xedgey: int 8 8 7 10 7 8 8 6 6 1 ...  
## $ yedge : int 0 4 3 2 5 9 7 2 1 1 ...  
## $ yedgex: int 8 10 9 8 10 7 10 7 7 8 ...

# divide into training and test data  
letters\_train <- letters[1:16000, ]  
letters\_test <- letters[16001:20000, ]

## Step3 - Training a model on the data

# begin by training a simple linear SVM  
  
letter\_classifier <- ksvm(letter ~ ., data = letters\_train,  
 kernel = "vanilladot")

## Setting default kernel parameters

# look at basic information about the model  
letter\_classifier

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 1   
##   
## Linear (vanilla) kernel function.   
##   
## Number of Support Vectors : 7037   
##   
## Objective Function Value : -14.1746 -20.0072 -23.5628 -6.2009 -7.5524 -32.7694 -49.9786 -18.1824 -62.1111 -32.7284 -16.2209 -32.2837 -28.9777 -51.2195 -13.276 -35.6217 -30.8612 -16.5256 -14.6811 -32.7475 -30.3219 -7.7956 -11.8138 -32.3463 -13.1262 -9.2692 -153.1654 -52.9678 -76.7744 -119.2067 -165.4437 -54.6237 -41.9809 -67.2688 -25.1959 -27.6371 -26.4102 -35.5583 -41.2597 -122.164 -187.9178 -222.0856 -21.4765 -10.3752 -56.3684 -12.2277 -49.4899 -9.3372 -19.2092 -11.1776 -100.2186 -29.1397 -238.0516 -77.1985 -8.3339 -4.5308 -139.8534 -80.8854 -20.3642 -13.0245 -82.5151 -14.5032 -26.7509 -18.5713 -23.9511 -27.3034 -53.2731 -11.4773 -5.12 -13.9504 -4.4982 -3.5755 -8.4914 -40.9716 -49.8182 -190.0269 -43.8594 -44.8667 -45.2596 -13.5561 -17.7664 -87.4105 -107.1056 -37.0245 -30.7133 -112.3218 -32.9619 -27.2971 -35.5836 -17.8586 -5.1391 -43.4094 -7.7843 -16.6785 -58.5103 -159.9936 -49.0782 -37.8426 -32.8002 -74.5249 -133.3423 -11.1638 -5.3575 -12.438 -30.9907 -141.6924 -54.2953 -179.0114 -99.8896 -10.288 -15.1553 -3.7815 -67.6123 -7.696 -88.9304 -47.6448 -94.3718 -70.2733 -71.5057 -21.7854 -12.7657 -7.4383 -23.502 -13.1055 -239.9708 -30.4193 -25.2113 -136.2795 -140.9565 -9.8122 -34.4584 -6.3039 -60.8421 -66.5793 -27.2816 -214.3225 -34.7796 -16.7631 -135.7821 -160.6279 -45.2949 -25.1023 -144.9059 -82.2352 -327.7154 -142.0613 -158.8821 -32.2181 -32.8887 -52.9641 -25.4937 -47.9936 -6.8991 -9.7293 -36.436 -70.3907 -187.7611 -46.9371 -89.8103 -143.4213 -624.3645 -119.2204 -145.4435 -327.7748 -33.3255 -64.0607 -145.4831 -116.5903 -36.2977 -66.3762 -44.8248 -7.5088 -217.9246 -12.9699 -30.504 -2.0369 -6.126 -14.4448 -21.6337 -57.3084 -20.6915 -184.3625 -20.1052 -4.1484 -4.5344 -0.828 -121.4411 -7.9486 -58.5604 -21.4878 -13.5476 -5.646 -15.629 -28.9576 -20.5959 -76.7111 -27.0119 -94.7101 -15.1713 -10.0222 -7.6394 -1.5784 -87.6952 -6.2239 -99.3711 -101.0906 -45.6639 -24.0725 -61.7702 -24.1583 -52.2368 -234.3264 -39.9749 -48.8556 -34.1464 -20.9664 -11.4525 -123.0277 -6.4903 -5.1865 -8.8016 -9.4618 -21.7742 -24.2361 -123.3984 -31.4404 -88.3901 -30.0924 -13.8198 -9.2701 -3.0823 -87.9624 -6.3845 -13.968 -65.0702 -105.523 -13.7403 -13.7625 -50.4223 -2.933 -8.4289 -80.3381 -36.4147 -112.7485 -4.1711 -7.8989 -1.2676 -90.8037 -21.4919 -7.2235 -47.9557 -3.383 -20.433 -64.6138 -45.5781 -56.1309 -6.1345 -18.6307 -2.374 -72.2553 -111.1885 -106.7664 -23.1323 -19.3765 -54.9819 -34.2953 -64.4756 -20.4115 -6.689 -4.378 -59.141 -34.2468 -58.1509 -33.8665 -10.6902 -53.1387 -13.7478 -20.1987 -55.0923 -3.8058 -60.0382 -235.4841 -12.6837 -11.7407 -17.3058 -9.7167 -65.8498 -17.1051 -42.8131 -53.1054 -25.0437 -15.302 -44.0749 -16.9582 -62.9773 -5.204 -5.2963 -86.1704 -3.7209 -6.3445 -1.1264 -122.5771 -23.9041 -355.0145 -31.1013 -32.619 -4.9664 -84.1048 -134.5957 -72.8371 -23.9002 -35.3077 -11.7119 -22.2889 -1.8598 -59.2174 -8.8994 -150.742 -1.8533 -1.9711 -9.9676 -0.5207 -26.9229 -30.429 -5.6289   
## Training error : 0.130062

## step4 - Evaluating model performance

# predictions on testing dataset  
letter\_predictions <- predict(letter\_classifier, letters\_test)  
  
head(letter\_predictions)

## [1] U N V X N H  
## Levels: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

table(letters\_test$letter, letter\_predictions)

## letter\_predictions  
## A B C D E F G H I J K L M N O P Q  
## A 144 0 0 2 0 0 1 0 0 0 1 0 0 0 1 0 0  
## B 0 121 0 2 0 0 1 0 1 1 1 0 0 0 0 0 0  
## C 0 0 120 0 5 0 2 0 0 0 9 0 1 0 2 0 0  
## D 0 5 0 156 0 0 1 1 0 0 0 0 1 0 1 1 0  
## E 0 2 4 0 127 0 9 0 0 0 0 2 0 0 0 0 0  
## F 0 0 0 1 3 138 2 1 1 1 0 0 0 1 0 2 0  
## G 0 1 10 3 1 2 123 0 0 0 2 1 1 0 1 1 8  
## H 0 2 2 10 1 2 2 102 0 2 5 1 1 1 2 0 2  
## I 0 0 2 4 0 6 0 0 141 5 0 0 0 0 0 0 0  
## J 1 0 0 3 0 0 0 2 8 128 0 0 0 0 1 0 0  
## K 0 1 1 4 3 0 1 3 0 0 118 0 0 0 0 0 0  
## L 0 0 3 3 4 0 2 2 0 0 0 133 0 0 0 0 3  
## M 1 1 0 0 0 0 1 3 0 0 0 0 135 0 0 0 0  
## N 2 0 0 5 0 0 0 4 0 0 2 0 4 145 1 0 0  
## O 2 0 2 5 0 0 1 20 0 1 0 0 0 0 99 2 3  
## P 0 2 0 3 0 16 2 0 1 1 1 0 0 0 3 130 1  
## Q 5 2 0 1 2 0 8 2 0 3 0 1 0 0 3 0 124  
## R 0 3 0 4 0 0 2 3 0 0 7 0 0 3 0 0 0  
## S 1 5 0 0 10 3 4 0 3 2 0 5 0 0 0 0 5  
## T 1 0 0 0 0 0 3 3 0 0 1 0 0 0 0 0 0  
## U 1 0 0 0 0 0 0 0 0 0 3 0 3 1 3 0 0  
## V 0 2 0 0 0 1 0 2 0 0 0 0 0 0 0 0 0  
## W 1 0 0 0 0 0 0 0 0 0 0 0 8 2 0 0 0  
## X 0 1 0 3 2 1 1 0 5 1 5 0 0 0 0 0 0  
## Y 0 0 0 3 0 2 0 1 1 0 0 0 0 0 0 1 2  
## Z 1 0 0 1 3 0 0 0 1 6 0 1 0 0 0 0 0  
## letter\_predictions  
## R S T U V W X Y Z  
## A 0 1 0 1 0 0 0 3 2  
## B 7 1 0 0 0 0 1 0 0  
## C 0 0 0 3 0 0 0 0 0  
## D 0 0 0 1 0 0 0 0 0  
## E 1 1 3 0 0 0 2 0 1  
## F 0 0 2 0 1 0 0 0 0  
## G 3 3 0 0 3 1 0 0 0  
## H 8 0 0 2 4 0 1 1 0  
## I 0 1 0 0 0 0 3 0 3  
## J 0 1 0 0 0 0 0 0 4  
## K 13 0 1 0 0 0 1 0 0  
## L 0 1 0 0 0 0 6 0 0  
## M 0 0 0 0 1 2 0 0 0  
## N 1 0 0 0 2 0 0 0 0  
## O 1 0 0 1 1 0 1 0 0  
## P 1 0 0 0 0 0 0 7 0  
## Q 0 14 0 0 3 0 0 0 0  
## R 138 0 0 0 1 0 0 0 0  
## S 0 101 3 0 0 0 1 0 18  
## T 1 3 133 0 0 0 0 3 3  
## U 0 0 1 152 0 4 0 0 0  
## V 1 0 0 0 126 4 0 0 0  
## W 0 0 0 0 1 127 0 0 0  
## X 0 2 0 1 0 0 137 0 0  
## Y 0 0 2 1 4 0 1 127 0  
## Z 0 10 2 0 0 0 1 0 132

# look only at agreement vs. non-agreement  
# construct a vector of TRUE/FALSE indicating correct/incorrect predictions  
agreement <- letter\_predictions == letters\_test$letter  
table(agreement)

## agreement  
## FALSE TRUE   
## 643 3357

prop.table(table(agreement))

## agreement  
## FALSE TRUE   
## 0.16075 0.83925

## step5 - Improving model performance

set.seed(12345)  
letter\_classifier\_rbf <- ksvm(letter ~ ., data = letters\_train, kernel = "rbfdot")  
letter\_predictions\_rbf <- predict(letter\_classifier\_rbf, letters\_test)  
  
table(letters\_test$letter, letter\_predictions\_rbf)

## letter\_predictions\_rbf  
## A B C D E F G H I J K L M N O P Q  
## A 151 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0  
## B 0 128 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0  
## C 0 0 133 0 2 0 2 0 0 0 1 0 0 0 2 0 0  
## D 0 3 0 161 0 0 0 1 0 0 0 0 0 0 0 0 0  
## E 0 0 3 0 137 0 8 0 0 0 0 0 0 0 0 0 0  
## F 0 1 0 0 2 148 0 0 0 0 0 0 0 2 0 0 0  
## G 0 0 1 2 0 0 154 2 0 0 0 1 1 0 0 0 0  
## H 0 2 0 8 0 0 2 126 0 0 4 0 1 0 0 0 1  
## I 0 0 2 2 0 3 0 0 151 3 0 0 0 0 0 1 0  
## J 0 0 0 3 1 0 0 1 3 136 0 0 0 0 1 0 0  
## K 0 0 0 1 0 0 0 2 0 0 132 0 0 0 0 0 0  
## L 0 1 1 0 4 0 2 1 0 0 0 142 0 0 0 0 0  
## M 0 2 0 0 0 0 2 1 0 0 0 0 138 0 0 0 0  
## N 0 1 0 1 0 0 0 3 0 0 1 0 1 150 5 0 0  
## O 0 0 0 1 0 0 2 0 0 0 0 0 0 0 129 0 3  
## P 0 2 0 3 1 11 1 1 0 0 0 0 0 0 2 141 3  
## Q 3 1 0 1 0 0 0 1 0 0 0 0 0 0 4 0 158  
## R 0 3 0 3 0 0 0 0 0 0 3 0 0 2 0 0 0  
## S 0 3 0 0 2 1 0 0 0 0 0 1 0 0 0 0 0  
## T 1 0 0 2 1 0 2 2 0 0 0 0 0 0 0 0 0  
## U 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0  
## V 0 3 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0  
## W 0 1 0 0 0 0 0 0 0 0 0 0 2 1 0 0 0  
## X 0 1 0 2 0 0 0 0 1 0 2 0 0 0 0 0 0  
## Y 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0  
## Z 0 0 0 0 2 0 0 0 0 3 0 0 0 0 0 0 0  
## letter\_predictions\_rbf  
## R S T U V W X Y Z  
## A 0 0 0 0 0 0 0 4 0  
## B 3 2 0 0 0 0 1 0 0  
## C 1 0 0 1 0 0 0 0 0  
## D 1 0 0 1 0 0 0 0 0  
## E 0 0 0 0 0 0 1 0 3  
## F 0 0 0 0 0 0 0 0 0  
## G 2 0 0 0 0 1 0 0 0  
## H 5 0 0 1 0 0 0 1 0  
## I 0 1 0 0 0 0 0 0 2  
## J 0 2 0 0 0 0 0 0 1  
## K 9 0 0 0 0 0 2 0 0  
## L 1 1 0 0 0 0 4 0 0  
## M 0 0 0 0 0 1 0 0 0  
## N 3 0 0 0 1 0 0 0 0  
## O 2 0 0 0 0 2 0 0 0  
## P 1 0 0 0 0 0 0 2 0  
## Q 0 0 0 0 0 0 0 0 0  
## R 150 0 0 0 0 0 0 0 0  
## S 0 152 0 0 0 0 1 0 1  
## T 1 0 140 0 0 0 1 1 0  
## U 0 0 0 161 2 3 0 0 0  
## V 0 0 0 0 131 0 0 0 0  
## W 0 0 0 0 0 135 0 0 0  
## X 0 0 0 0 0 0 153 0 0  
## Y 0 0 1 1 1 0 1 138 0  
## Z 0 2 0 0 0 0 1 0 150

agreement\_rbf <- letter\_predictions\_rbf == letters\_test$letter  
table(agreement\_rbf)

## agreement\_rbf  
## FALSE TRUE   
## 275 3725

prop.table(table(agreement\_rbf))

## agreement\_rbf  
## FALSE TRUE   
## 0.06875 0.93125

# using h2o deeplearning  
  
  
  
h2o.init()

##   
## H2O is not running yet, starting it now...  
##   
## Note: In case of errors look at the following log files:  
## C:\Users\LEIDEN~1\AppData\Local\Temp\RtmpEF6sQH/h2o\_Lei\_Deng\_started\_from\_r.out  
## C:\Users\LEIDEN~1\AppData\Local\Temp\RtmpEF6sQH/h2o\_Lei\_Deng\_started\_from\_r.err  
##   
##   
## Starting H2O JVM and connecting: ..... Connection successful!  
##   
## R is connected to the H2O cluster:   
## H2O cluster uptime: 10 seconds 486 milliseconds   
## H2O cluster version: 3.10.4.6   
## H2O cluster version age: 28 days, 16 hours and 11 minutes   
## H2O cluster name: H2O\_started\_from\_R\_Lei\_Deng\_bta090   
## H2O cluster total nodes: 1   
## H2O cluster total memory: 3.54 GB   
## H2O cluster total cores: 8   
## H2O cluster allowed cores: 2   
## H2O cluster healthy: TRUE   
## H2O Connection ip: localhost   
## H2O Connection port: 54321   
## H2O Connection proxy: NA   
## H2O Internal Security: FALSE   
## R Version: R version 3.4.0 (2017-04-21)   
##   
## 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)

letterdata.hex <- h2o.importFile("letterdata.csv")

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

summary(letterdata.hex)

## Warning in summary.H2OFrame(letterdata.hex): Approximated quantiles  
## computed! If you are interested in exact quantiles, please pass the  
## `exact\_quantiles=TRUE` parameter.

## letter xbox ybox width   
## U:813 Min. : 0.000 Min. : 0.000 Min. : 0.000   
## D:805 1st Qu.: 3.000 1st Qu.: 5.000 1st Qu.: 4.000   
## P:803 Median : 4.000 Median : 7.000 Median : 5.000   
## T:796 Mean : 4.024 Mean : 7.035 Mean : 5.122   
## M:792 3rd Qu.: 5.000 3rd Qu.: 9.000 3rd Qu.: 6.000   
## A:789 Max. :15.000 Max. :15.000 Max. :15.000   
## height onpix xbar ybar   
## Min. : 0.000 Min. : 0.000 Min. : 0.000 Min. : 0.0   
## 1st Qu.: 4.000 1st Qu.: 2.000 1st Qu.: 6.000 1st Qu.: 6.0   
## Median : 6.000 Median : 3.000 Median : 7.000 Median : 7.0   
## Mean : 5.372 Mean : 3.506 Mean : 6.898 Mean : 7.5   
## 3rd Qu.: 7.000 3rd Qu.: 5.000 3rd Qu.: 8.000 3rd Qu.: 9.0   
## Max. :15.000 Max. :15.000 Max. :15.000 Max. :15.0   
## x2bar y2bar xybar x2ybar   
## Min. : 0.000 Min. : 0.000 Min. : 0.000 Min. : 0.000   
## 1st Qu.: 3.000 1st Qu.: 4.000 1st Qu.: 7.000 1st Qu.: 5.000   
## Median : 4.000 Median : 5.000 Median : 8.000 Median : 6.000   
## Mean : 4.629 Mean : 5.179 Mean : 8.282 Mean : 6.454   
## 3rd Qu.: 6.000 3rd Qu.: 7.000 3rd Qu.:10.000 3rd Qu.: 8.000   
## Max. :15.000 Max. :15.000 Max. :15.000 Max. :15.000   
## xy2bar xedge xedgey yedge   
## Min. : 0.000 Min. : 0.000 Min. : 0.000 Min. : 0.000   
## 1st Qu.: 7.000 1st Qu.: 1.000 1st Qu.: 8.000 1st Qu.: 2.000   
## Median : 8.000 Median : 3.000 Median : 8.000 Median : 3.000   
## Mean : 7.929 Mean : 3.046 Mean : 8.339 Mean : 3.692   
## 3rd Qu.: 9.000 3rd Qu.: 4.000 3rd Qu.: 9.000 3rd Qu.: 5.000   
## Max. :15.000 Max. :15.000 Max. :15.000 Max. :15.000   
## yedgex   
## Min. : 0.000   
## 1st Qu.: 7.000   
## Median : 8.000   
## Mean : 7.801   
## 3rd Qu.: 9.000   
## Max. :15.000

splits <- h2o.splitFrame(letterdata.hex, 0.80, seed=1234)  
  
dl <- h2o.deeplearning(x=2:17,y="letter",training\_frame=splits[[1]],activation = "RectifierWithDropout",   
 hidden = c(16,16,16), distribution = "multinomial",input\_dropout\_ratio=0.2,  
 epochs = 10,nfold=5,variable\_importances = TRUE)

##   
 |   
 | | 0%  
 |   
 |=============================== | 48%  
 |   
 |======================================================== | 87%  
 |   
 |=================================================================| 100%

dl.predict <- h2o.predict (dl, splits[[2]])

##   
 |   
 | | 0%  
 |   
 |========================== | 41%  
 |   
 |=================================================================| 100%

dl@parameters

## $model\_id  
## [1] "DeepLearning\_model\_R\_1495747374794\_1"  
##   
## $training\_frame  
## [1] "RTMP\_sid\_938d\_2"  
##   
## $nfolds  
## [1] 5  
##   
## $overwrite\_with\_best\_model  
## [1] FALSE  
##   
## $activation  
## [1] "RectifierWithDropout"  
##   
## $hidden  
## [1] 16 16 16  
##   
## $epochs  
## [1] 10.40997  
##   
## $seed  
## [1] 5.296962e+18  
##   
## $input\_dropout\_ratio  
## [1] 0.2  
##   
## $distribution  
## [1] "multinomial"  
##   
## $stopping\_rounds  
## [1] 0  
##   
## $variable\_importances  
## [1] TRUE  
##   
## $x  
## [1] "xbox" "ybox" "width" "height" "onpix" "xbar" "ybar"   
## [8] "x2bar" "y2bar" "xybar" "x2ybar" "xy2bar" "xedge" "xedgey"  
## [15] "yedge" "yedgex"  
##   
## $y  
## [1] "letter"

h2o.performance(dl)

## H2OMultinomialMetrics: deeplearning  
## \*\* Reported on training data. \*\*  
## \*\* Metrics reported on temporary training frame with 10060 samples \*\*  
##   
## Training Set Metrics:   
## =====================  
##   
## MSE: (Extract with `h2o.mse`) 0.8823238  
## RMSE: (Extract with `h2o.rmse`) 0.9393209  
## Logloss: (Extract with `h2o.logloss`) 3.005762  
## Mean Per-Class Error: 0.9048364  
## Confusion Matrix: Extract with `h2o.confusionMatrix(<model>,train = TRUE)`)  
## =========================================================================  
## Confusion Matrix: vertical: actual; across: predicted  
## A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Error  
## A 0 0 0 0 0 0 40 0 267 0 0 70 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1.0000  
## B 0 0 0 0 92 0 213 0 79 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1.0000  
## C 0 0 0 0 151 0 46 0 160 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 3 1.0000  
## D 0 0 0 0 6 0 290 0 116 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 1.0000  
## E 0 0 0 0 127 0 48 0 193 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0.6622  
## Rate  
## A = 378 / 378  
## B = 386 / 386  
## C = 363 / 363  
## D = 419 / 419  
## E = 249 / 376  
##   
## ---  
## A B C D E F G H I J K L M N O P Q R S T U V W X Y  
## V 0 0 0 0 0 0 270 0 0 0 0 0 18 0 0 0 0 0 0 0 0 121 0 0 0  
## W 0 0 0 0 0 0 56 0 1 0 0 0 144 0 0 0 0 0 0 0 0 187 0 0 0  
## X 0 0 0 0 73 0 53 0 257 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## Y 0 0 0 0 4 0 347 0 2 0 0 0 3 0 0 0 0 0 0 0 0 16 0 0 0  
## Z 0 0 0 0 144 0 40 0 182 0 0 5 0 0 0 0 0 0 0 0 0 1 0 0 0  
## Totals 54 0 0 0 1034 0 4515 0 2883 1 0 481 303 0 0 0 0 0 0 0 0 761 0 0 0  
## Z Error Rate  
## V 0 0.7042 = 288 / 409  
## W 0 1.0000 = 388 / 388  
## X 2 1.0000 = 385 / 385  
## Y 0 1.0000 = 372 / 372  
## Z 0 1.0000 = 372 / 372  
## Totals 28 0.9048 = 9,102 / 10,060  
##   
## Hit Ratio Table: Extract with `h2o.hit\_ratio\_table(<model>,train = TRUE)`  
## =======================================================================  
## Top-10 Hit Ratios:   
## k hit\_ratio  
## 1 1 0.095229  
## 2 2 0.168986  
## 3 3 0.239264  
## 4 4 0.278032  
## 5 5 0.317893  
## 6 6 0.362922  
## 7 7 0.401392  
## 8 8 0.441551  
## 9 9 0.491849  
## 10 10 0.519881

h2o.varimp(dl)

## Variable Importances:   
## variable relative\_importance scaled\_importance percentage  
## 1 ybar 1.000000 1.000000 0.089730  
## 2 x2ybar 0.916170 0.916170 0.082208  
## 3 xedge 0.828784 0.828784 0.074367  
## 4 x2bar 0.796075 0.796075 0.071432  
## 5 y2bar 0.777035 0.777035 0.069724  
## 6 xy2bar 0.776719 0.776719 0.069695  
## 7 yedge 0.744857 0.744857 0.066836  
## 8 yedgex 0.650059 0.650059 0.058330  
## 9 xybar 0.648241 0.648241 0.058167  
## 10 onpix 0.638589 0.638589 0.057301  
## 11 xbar 0.627785 0.627785 0.056331  
## 12 ybox 0.615188 0.615188 0.055201  
## 13 xedgey 0.574634 0.574634 0.051562  
## 14 height 0.541802 0.541802 0.048616  
## 15 xbox 0.505162 0.505162 0.045328  
## 16 width 0.503398 0.503398 0.045170

h2o.shutdown()

## Are you sure you want to shutdown the H2O instance running at http://localhost:54321/ (Y/N)?

## [1] TRUE