2017195113_home

12/14/2020

```
library(tidyverse)
library(magrittr)
library(outliers)
library(caret)
library(doParallel)

seed = 323
```

#1. Data Preprocessing —-

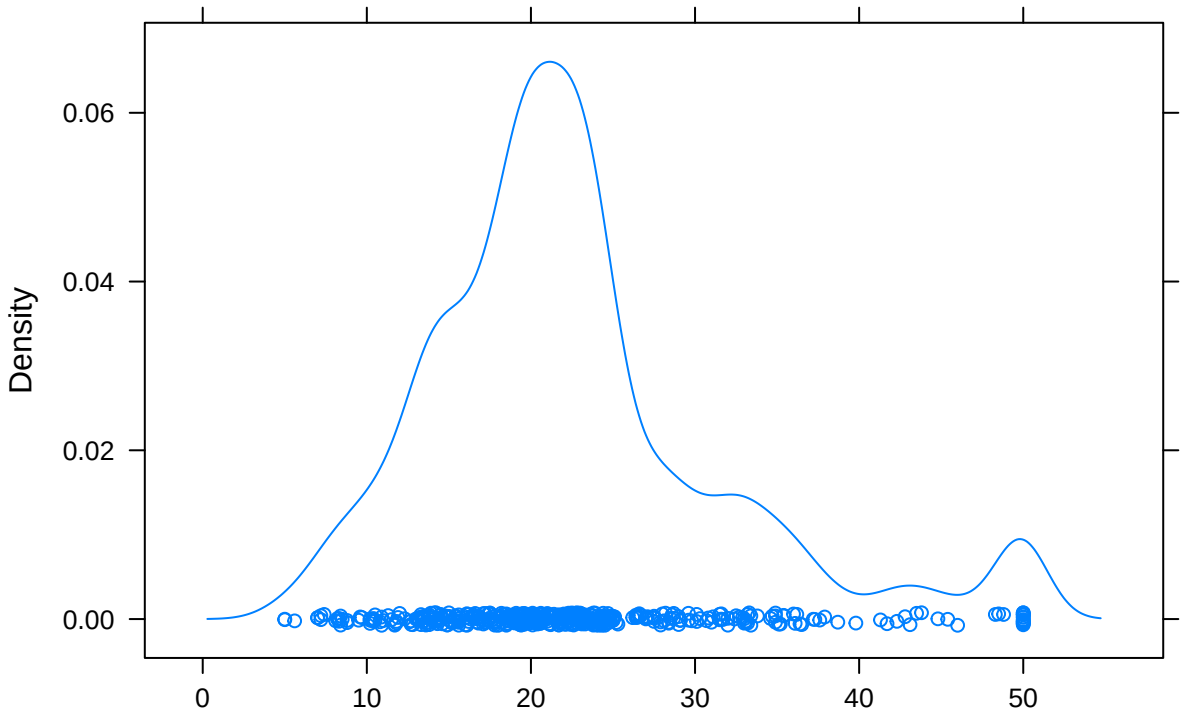
```
# data.raw = read.csv("homevalues.csv")
#
# #1.1 Data overview
# str(data.raw)
# data.raw %>% view()
# data.raw %>% sapply(., class) #rm dummy var
#
# #removing dummy variable
# data.raw %<>% select(-chas)
# data.raw %>% names()
#
# #1.2 check NA / Impute NA
# data.raw %>% anyNA
#
# #1.3 Data Cleaning (outliers, Typos)
# #Descriptive Stats
# summary(data.raw)
# outlier(data.raw)
# data.raw %>% Hmisc::hist.data.frame() #all outliers ok, but should change lon
# data.raw %>% str()
#
# #changing loan
# data.raw$lon %<>%
#   substr(2, 6) %>% as.numeric
# data.raw$lon

#final check
data.raw %>% str()
```

```
## 'data.frame':    506 obs. of  17 variables:
##  $ ID       : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
##  $ homevalue: num  24 21.6 34.7 33.4 36.2 28.7 22.9 22.1 16.5 18.9 ...
##  $ tract    : int  2011 2021 2022 2031 2032 2033 2041 2042 2043 2044 ...
##  $ lon      : num  71 71 70.9 70.9 70.9 ...
##  $ lat      : num  42.3 42.3 42.3 42.3 42.3 ...
##  $ crim     : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn       : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus    : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ nox      : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm       : num  6.58 6.42 7.18 7 7.15 ...
##  $ age      : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis      : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad      : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax      : int  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio  : num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ b        : num  397 397 393 395 397 ...
##  $ lstat    : num  4.98 9.14 4.03 2.94 5.33 ...
```

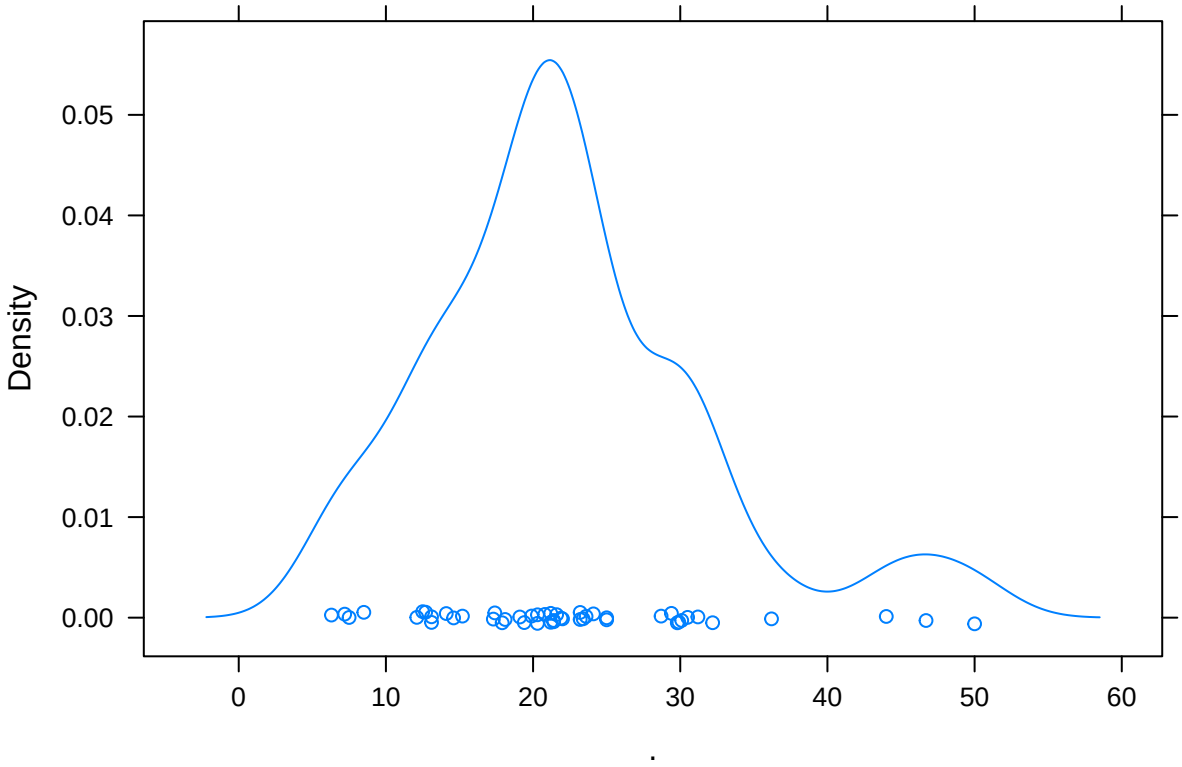
#2. Splitting Data & Configurations —-

```
#2.1 Splitting Data
target.label = "homevalue"
set.seed(seed)
train.index = createDataPartition(data.raw[[target.label]], p = 0.9, list = F)
trainset = data.raw[train.index,]
testset = data.raw[-train.index,]
#checking training & testing data formation
trainset[[target.label]] %>% densityplot()
```



.

```
testset[[target.label]] %>% densityplot() #similar enough
```



.

```
#2.2 Target Selection & Formula Creation
target = trainset[[target.label]]
features.label = trainset %>% select(-target.label) %>% names() %T>% print()
```

```
##  [1] "ID"      "tract"   "lon"     "lat"     "crim"    "zn"      "indus"
##  [8] "nox"     "rm"      "age"     "dis"     "rad"     "tax"     "ptratio"
## [15] "b"       "lstat"
```

```
features = trainset %>% select(features.label) %>% as.data.frame()

formula = features %>%
  names() %>%
  paste(., collapse = " + ") %>%
  paste(target.label, "~ ", .) %>%
  as.formula(env = .GlobalEnv) %T>% print
```

```
## homevalue ~ ID + tract + lon + lat + crim + zn + indus + nox +
##     rm + age + dis + rad + tax + ptratio + b + lstat
```

```
#2.3 trControl configurations
trControl = trainControl(method = "repeatedcv",
                         number = 5,
                         repeats = 3,
```

```
                          search = "random",
                          allowParallel = T)

#2.4 preProc configurations
preProc = c("scale", "center")

#2.5 Metric configuration
metric = "RMSE"
```

#3. 8 Training Models on trainset —-

```
# cl = makePSOCKcluster(5)
# registerDoParallel(cl)
#
# #knn
# set.seed(seed)
# fit.knn = train(formula,
#                  data = trainset,
#                  method = "knn",
#                  preProc = preProc)
# #logistic regression
# set.seed(seed)
# fit.lg = train(formula,
#                data = trainset,
#                method = "glm",
#                family = "binomial",
#                preProc = preProc)
#
# #random forest
# set.seed(seed)
# fit.rf = train(formula,
#                data = trainset,
#                method = "rf",
#                preProc = preProc)
# #gbm
# set.seed(seed)
# fit.gbm = train(formula,
#                  data = trainset,
#                  method = "gbm",
#                  preProc = preProc)
# #svm
# set.seed(seed)
# fit.svm = train(formula,
#                  data = trainset,
#                  method = "svmRadial",
#                  preProc = preProc)
# #nnet
# set.seed(seed)
# fit.nnet = train(formula,
#                   data = trainset,
```

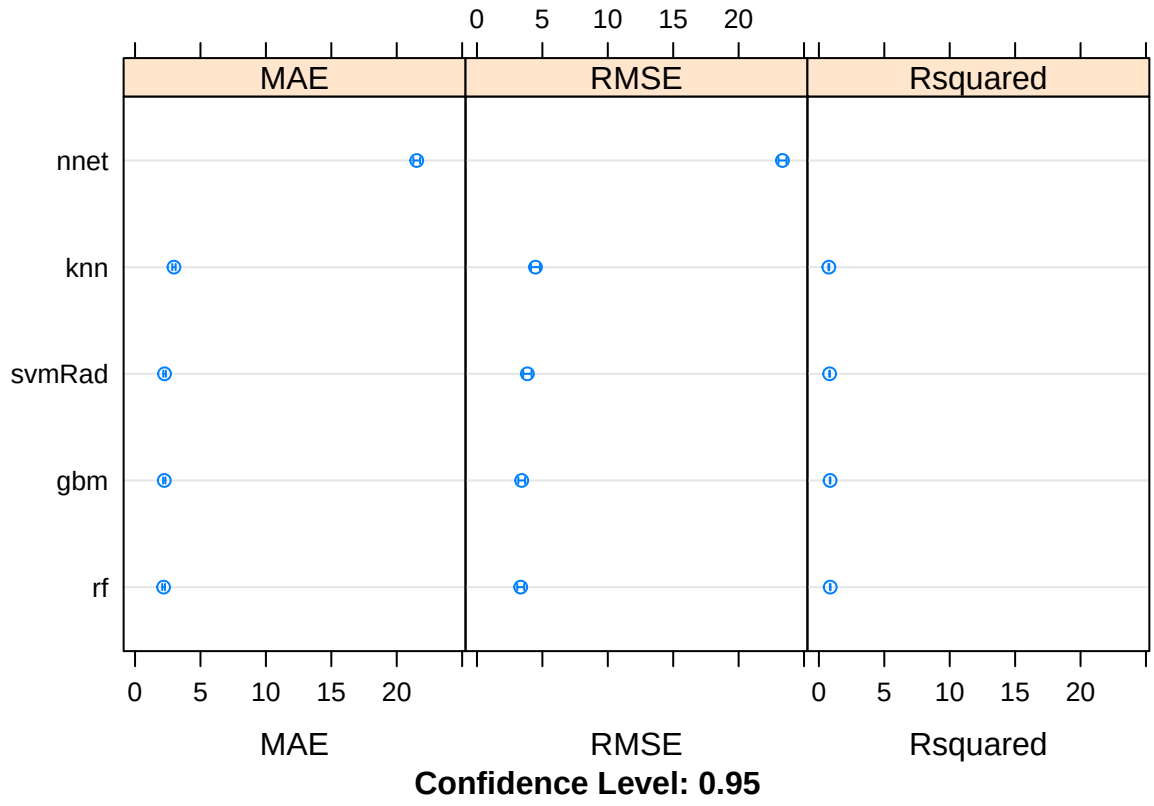
```
#                   method = "nnet",
#                   preProc = preProc)
#
# stopCluster(cl)
```

#4. comparing training performance of all 8 models —-

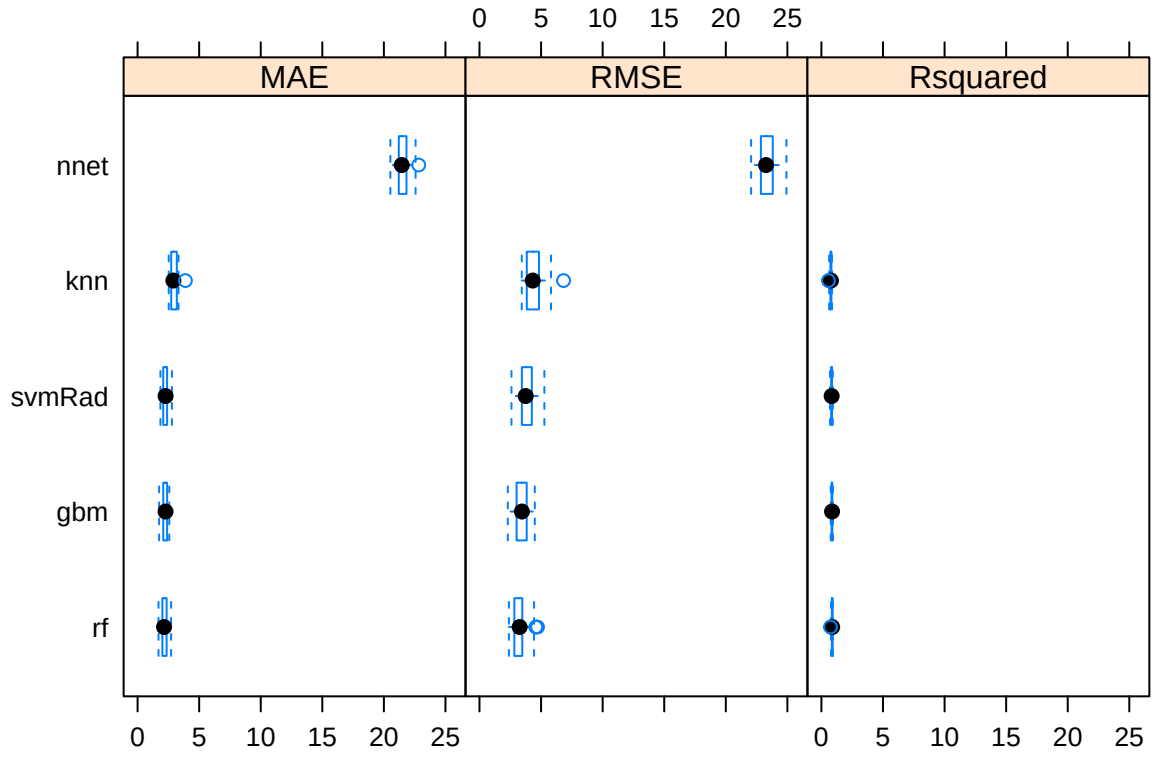
```
results = resamples(list(knn = fit.knn,
                         rf = fit.rf, gbm = fit.gbm, svmRad = fit.svm, nnet = fit.nnet))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: knn, rf, gbm, svmRad, nnet
## Number of resamples: 25
##
## MAE
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## knn     2.526845  2.732004  2.905563  2.973227  3.186324  3.882310    0
## rf      1.700311  2.014383  2.150684  2.181755  2.355459  2.720249    0
## gbm     1.750553  2.084143  2.269813  2.234712  2.399271  2.574545    0
## svmRad  1.853280  2.080642  2.274756  2.254809  2.397516  2.788168    0
## nnet   20.528488 21.210778 21.449133 21.532171 21.832927 22.827778    0
##
## RMSE
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## knn     3.433975  3.839246  4.323562  4.470945  4.829097  6.826664    0
## rf      2.387819  2.828329  3.261338  3.341789  3.473348  4.713870    0
## gbm     2.298377  3.010928  3.441758  3.417931  3.834050  4.488596    0
## svmRad  2.597430  3.439723  3.751159  3.852667  4.249401  5.267112    0
## nnet   22.055763 22.849738 23.270250 23.348063 23.813875 24.924029    0
##
## Rsquared
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## knn    0.5703995 0.7321722 0.7748597 0.7596123 0.8135766 0.8636371    0
## rf     0.7370750 0.8519513 0.8747975 0.8662457 0.9122570 0.9263591    0
## gbm    0.7612610 0.8235173 0.8624987 0.8559566 0.8885566 0.9335257    0
## svmRad 0.7027640 0.7925418 0.8307896 0.8206894 0.8624776 0.9254783    0
## nnet          NA        NA        NA       NaN        NA        NA   25
```

```
dot.plot=dotplot(results) %T>% print()
```




```
bwplot = bwplot(results) %T>% print()
```



```
# I will tune nnet and svmRad (instead of knn, bc svm better performs when tuned than knn)
```

#5. Tuning 2 models using train function —-

```
#5.3 svm
#Tune grid
# print(fit.svm) #used sigma = 0.05878653 and C = 1
# getModelInfo("svmRadial")
# tunegrid.svm = expand.grid(
#   .sigma = c(0.04 : 0.07),
#   .C = c(0.5, 1, 1.5)
# )
# #Training tuned model
# set.seed(seed)
#
# cl = makePSOCKcluster(5)
# registerDoParallel(cl)
# tune.svm = train(formula,
#                  data = trainset,
#                  method = "svmRadial",
#                  linout = T,
#                  metric = metric,  preProc = preProc,
#                  trControl = trControl, tuneGrid = tunegrid.svm)
# stopCluster(cl)
# #Checking train performance
# print(tune.svm)
# tune.svm %>% getTrainPerf() #RMSE 3.572557
#
#
# #5.4 nnet
# #Tune grid
# print(fit.nnet) #size = 1 and decay = 0
# getModelInfo("nnet")
# tunegrid.nnet = expand.grid(.size = c(0.5, 1, 1.5),
#                             .decay = c(0, 0.05, 0.1))
# set.seed(seed)
# cl = makePSOCKcluster(5)
# registerDoParallel(cl)
#
# tune.nnet = train(formula,
#                   data = trainset,
#                   method= "nnet",
#                   metric = metric, preProc = preProc,
#                   linout = T,
#                   trControl = trControl, tuneGrid = tunegrid.nnet)
# stopCluster(cl)
#
# #Checking train performance
# print(tune.nnet)
# tune.nnet %>% getTrainPerf() #RMSE  4.330002
```

#6. Final Analysis —-

```
predict(tune.svm, testset)
```

```
##  [1] 22.459414 30.052680 24.568231 13.978605 14.857961 17.747408 26.425018
##  [8] 24.280707 22.898530 20.896543 28.648687 17.276011 17.851091 16.720787
## [15] 14.558989 29.805151 30.408308 41.963966 28.810521 28.981416 23.737710
## [22] 41.192596 39.393730 32.541278 23.865433 20.932195 20.146047 20.145819
## [29] 23.171652 27.238088 24.624805 19.961488 23.409490 20.897811 15.277448
## [36] 10.401466 12.732692 13.767655  5.632443 11.448980 10.417196 16.132860
## [43] 13.510784 19.951786 16.331190 21.058688 15.987552 20.894734
```

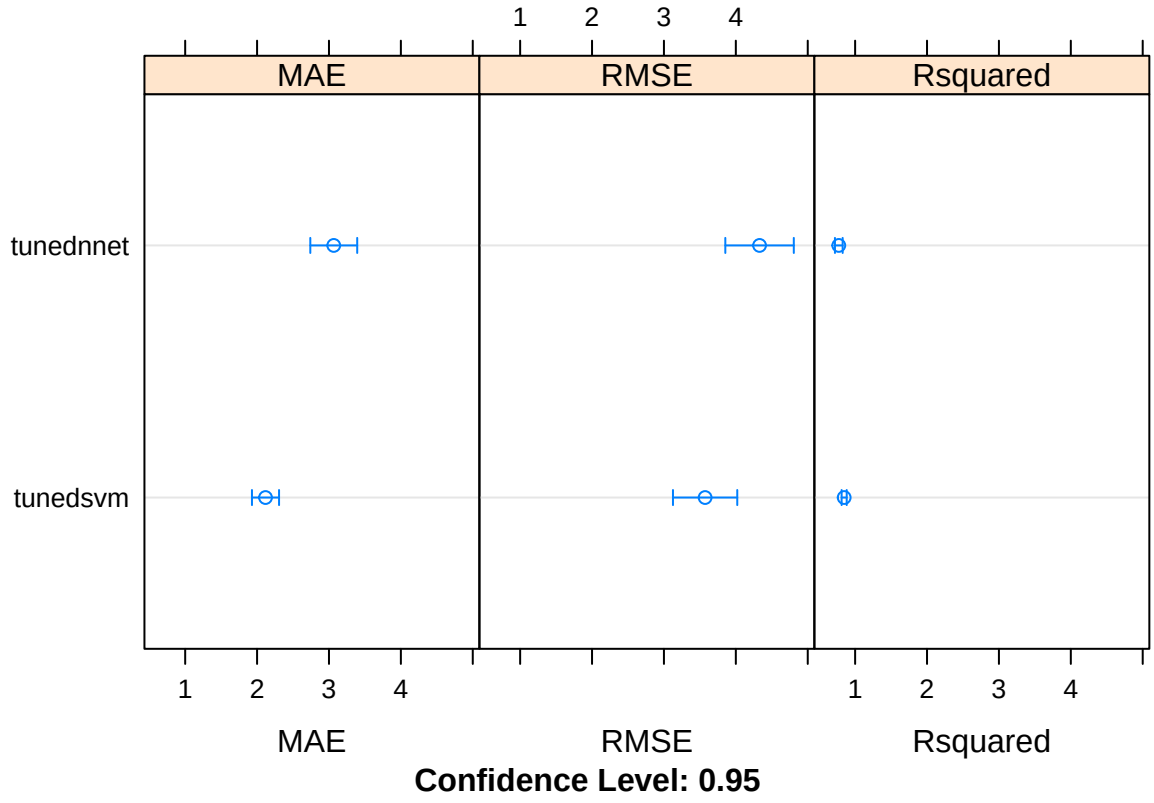
```
predict(tune.nnet, testset)
```

```
##        2        5        6       31       34       50       64       74
## 27.28162 32.39481 28.89920 17.55289 17.77552 18.40899 25.34134 22.75405
##       76       79       89      124      136      137      157      176
## 23.08352 20.06810 30.93673 17.78679 18.48022 18.24996 17.54749 32.10503
##      179      187      189      192      202      229      257      285
## 32.27559 35.81689 33.46109 31.68698 29.60148 36.68348 37.07256 32.68351
##      286      288      298      310      325      345      354      362
## 26.83671 29.87233 18.94484 21.30172 24.82825 28.85874 26.77726 26.18538
##      363      367      397      400      402      403      405      413
## 27.51113 21.04372 18.82636 17.42706 17.42623 17.50298 17.42301 17.42301
##      417      452      456      465      469      473      479      499
## 17.42506 19.29324 17.94721 18.10746 17.42369 25.44064 17.46220 22.90551
```

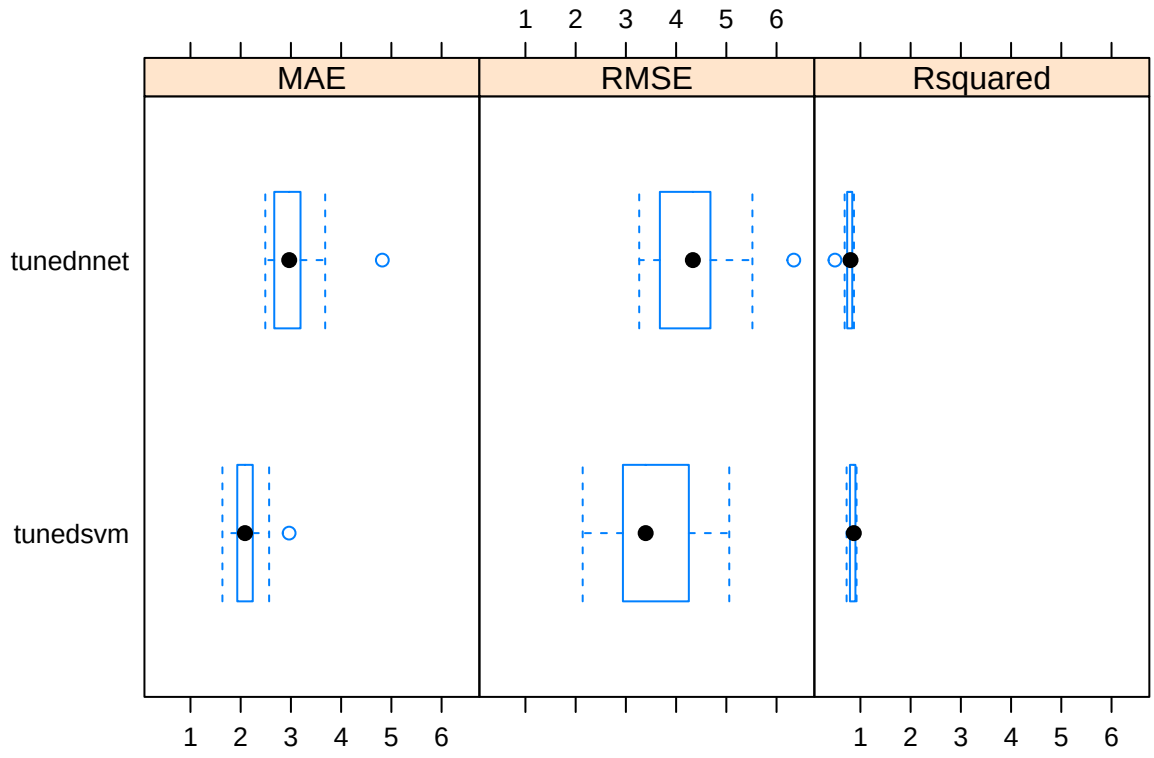
```
results1 = resamples(list(tunedsvm = tune.svm, tunednnet = tune.nnet))
summary(results1)
```

```
##
## Call:
## summary.resamples(object = results1)
##
## Models: tunedsvm, tunednnet
## Number of resamples: 15
##
## MAE
##               Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## tunedsvm  1.636270 1.932008 2.086096 2.117357 2.240259 2.964745    0
## tunednnet 2.489779 2.668153 2.966409 3.066477 3.191298 4.820081    0
##
## RMSE
##               Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## tunedsvm  2.139759 2.941029 3.393944 3.572557 4.253546 5.058446    0
## tunednnet 3.265491 3.676922 4.333660 4.330002 4.683607 6.343496    0
##
## Rsquared
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## tunedsvm  0.7249278 0.7915283 0.8685364 0.8477156 0.9010480 0.9229168    0
## tunednnet 0.4931157 0.7362415 0.8030617 0.7729814 0.8378647 0.8716280    0
```

```
dot.plot=dotplot(results1) %T>% print()
```




```
bwplot = bwplot(results1) %T>% print()
```



```
#the svm model has a lower RMSE than nnet. Thus, svm predicts house values better.
```