BUS/CSC 386 Homework #2  
Regression Neural Network

Using the housing.csv dataset from california-house-prices (provided), do the following:

1. Import the data using the read.csv function

**Done**  
2. Inspect the data using the glimpse function and document the number of variables and samples

**Observations: 20640**

**Variables: 10**  
3. Move the median-house-value response variable to the first column

**Done**  
4. Split the data into training and test, features and outcomes, respectively using 80% of the  
dataset for training and 20% for test

**Done**  
5. Create new vectors for training and test outcomes, and delete the response variables from the predictor datasets

**Done**  
6. Show the dimension of the training and test predictor sets

7. Show the structure of the training and test predictor sets

8. Create one-hot dummy variables for ocean\_proximity using recipe

**Done**  
9. Center and scale both training and test predictors using recipes

**Done**  
10. Build a keras sequential neural densely connected network with:  
a. 16 input neurons; relu activation; initializer\_he\_normal  
b. 16 hidden layer neurons; same as first layer  
c. 1 output layer neuron, sigmoid activation

**Done**

11. Compile the learning section as follows:  
a. Optimizer = adam  
b. Loss = mse  
c. Metric = c(mae)

**Done**

12. Fit the network to the training data, run for 10 epochs with a batch size of 128 and a validation  
split of 20%.

**Done**  
13. Paste the R code into your homework submission

library (keras)

library (dplyr)

library (rsample)

library (recipes)

Mydata\_raw <- read.csv(file="~/github/DeepLearningR/Homework 2/housing.csv", header=TRUE, sep=",")

Mydata <- Mydata\_raw %>%

 select(median\_house\_value, everything())

glimpse(Mydata)

set.seed(619)

train\_test\_split <- initial\_split(Mydata, prop=.8)

Train\_tbl <- training(train\_test\_split)

Test\_tbl <- testing(train\_test\_split)

dim(Train\_tbl)

dim(Test\_tbl)

rec\_obj <- recipe(median\_house\_value ~ ., data = Train\_tbl) %>%

 step\_bagimpute(all\_predictors(), -all\_outcomes()) %>%

 step\_center(all\_numeric(), -all\_outcomes()) %>%

 step\_scale(all\_numeric(), -all\_outcomes()) %>%

 step\_dummy(all\_nominal(), -all\_outcomes(), one\_hot = TRUE) %>%

 prep(data = Train\_tbl)

rec\_obj

x\_train\_tbl <- bake(rec\_obj, new\_data = Train\_tbl) %>% select(-median\_house\_value)

x\_test\_tbl <- bake(rec\_obj, new\_data = Test\_tbl) %>% select(-median\_house\_value)

y\_train <- Train\_tbl[,1]

y\_test <- Test\_tbl[,1]

model <- keras\_model\_sequential() %>%

 layer\_dense(units = 16, activation = "linear", input\_shape = ncol(x\_train\_tbl)) %>%

 layer\_dense(units = 16, activation = "linear") %>%

 # layer\_dense(units = 128, activation = "linear") %>%

 layer\_dense(units = 1)

model %>% compile(

 optimizer = "Adam",

 loss = "mse",

 metrics = c("mae")

)

history <- model %>% fit(

 as.matrix(x\_train\_tbl), y\_train,

 validation\_split = .2,

 epochs = 10,

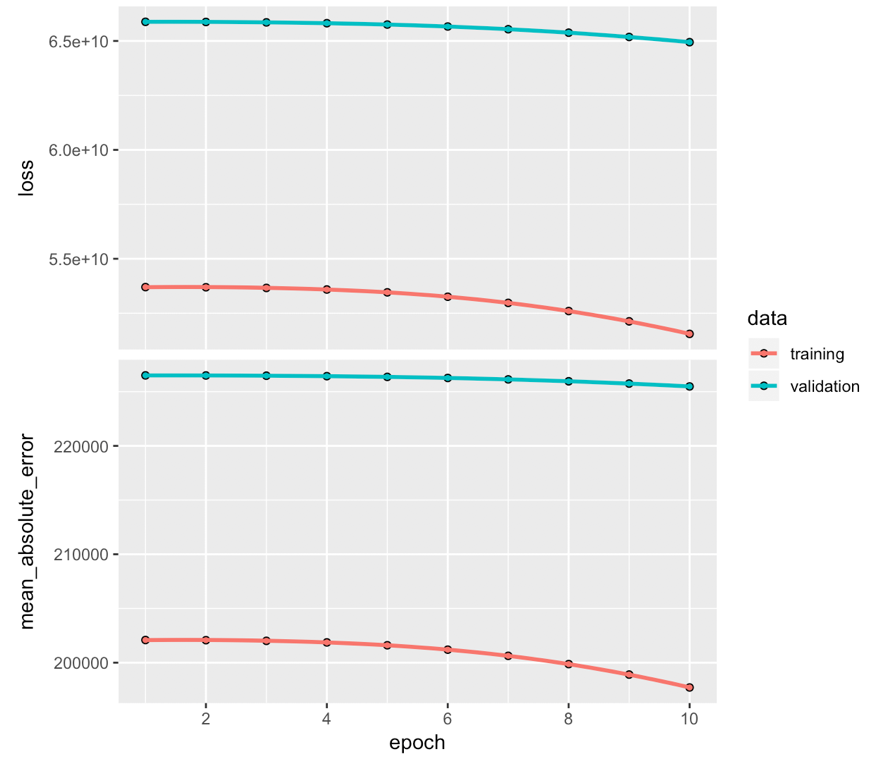
 batch\_size = 128

)

plot(history)

result <- model %>% evaluate(as.matrix(x\_test\_tbl), y\_test)

result

14. Paste the validation accuracy graph into your homework submission

15. Run 5 different permutations of this initial network and keep the best model using the following  
as a guide:  
a. Change the number of layers

**Added one layer with 16 neurons - Large drop in mean absolute error, fitting much faster**b. Change the number of neurons per layer

**Change first layer to 256 neurons - Training slower, down but less than adding layer**

**Changing second layer to 256 neurons - same results as above**

**First layer - 256, second layer - 128 - Incredibly better, much better fitting and training speed**c. Change the activation function of each layer

**Very little change for any combination**d. Change the optimizer and/or the learning rate

**Only able to make minimal changes**e. Change the batch size

**Lowering batch size lowers absolute error by a lot**f. Change the number of epochs

**More epochs lowers the error**16. Document every change event with the corresponding validation accuracy and enter into the  
homework (brief comments and text only)  
17. Paste the R code and validation accuracy graph of your best model into your homework

library (keras)

library (dplyr)

library (rsample)

library (recipes)

Mydata\_raw <- read.csv(file="~/github/DeepLearningR/Homework 2/housing.csv", header=TRUE, sep=",")

Mydata <- Mydata\_raw %>%

 select(median\_house\_value, everything())

glimpse(Mydata)

set.seed(619)

train\_test\_split <- initial\_split(Mydata, prop=.8)

Train\_tbl <- training(train\_test\_split)

Test\_tbl <- testing(train\_test\_split)

dim(Train\_tbl)

dim(Test\_tbl)

rec\_obj <- recipe(median\_house\_value ~ ., data = Train\_tbl) %>%

 step\_bagimpute(all\_predictors(), -all\_outcomes()) %>%

 step\_center(all\_numeric(), -all\_outcomes()) %>%

 step\_scale(all\_numeric(), -all\_outcomes()) %>%

 step\_dummy(all\_nominal(), -all\_outcomes(), one\_hot = TRUE) %>%

 prep(data = Train\_tbl)

rec\_obj

x\_train\_tbl <- bake(rec\_obj, new\_data = Train\_tbl) %>% select(-median\_house\_value)

x\_test\_tbl <- bake(rec\_obj, new\_data = Test\_tbl) %>% select(-median\_house\_value)

y\_train <- Train\_tbl[,1]

y\_test <- Test\_tbl[,1]

model <- keras\_model\_sequential() %>%

 layer\_dense(units = 256, activation = "relu", input\_shape = ncol(x\_train\_tbl)) %>%

 layer\_dense(units = 128, activation = "relu") %>%

 layer\_dense(units = 128, activation = "relu") %>%

 layer\_dense(units = 1)

model %>% compile(

 optimizer = "Adam",

 loss = "mse",

 metrics = c("mae")

)

history <- model %>% fit(

 as.matrix(x\_train\_tbl), y\_train,

 validation\_split = .2,

 epochs = 18,

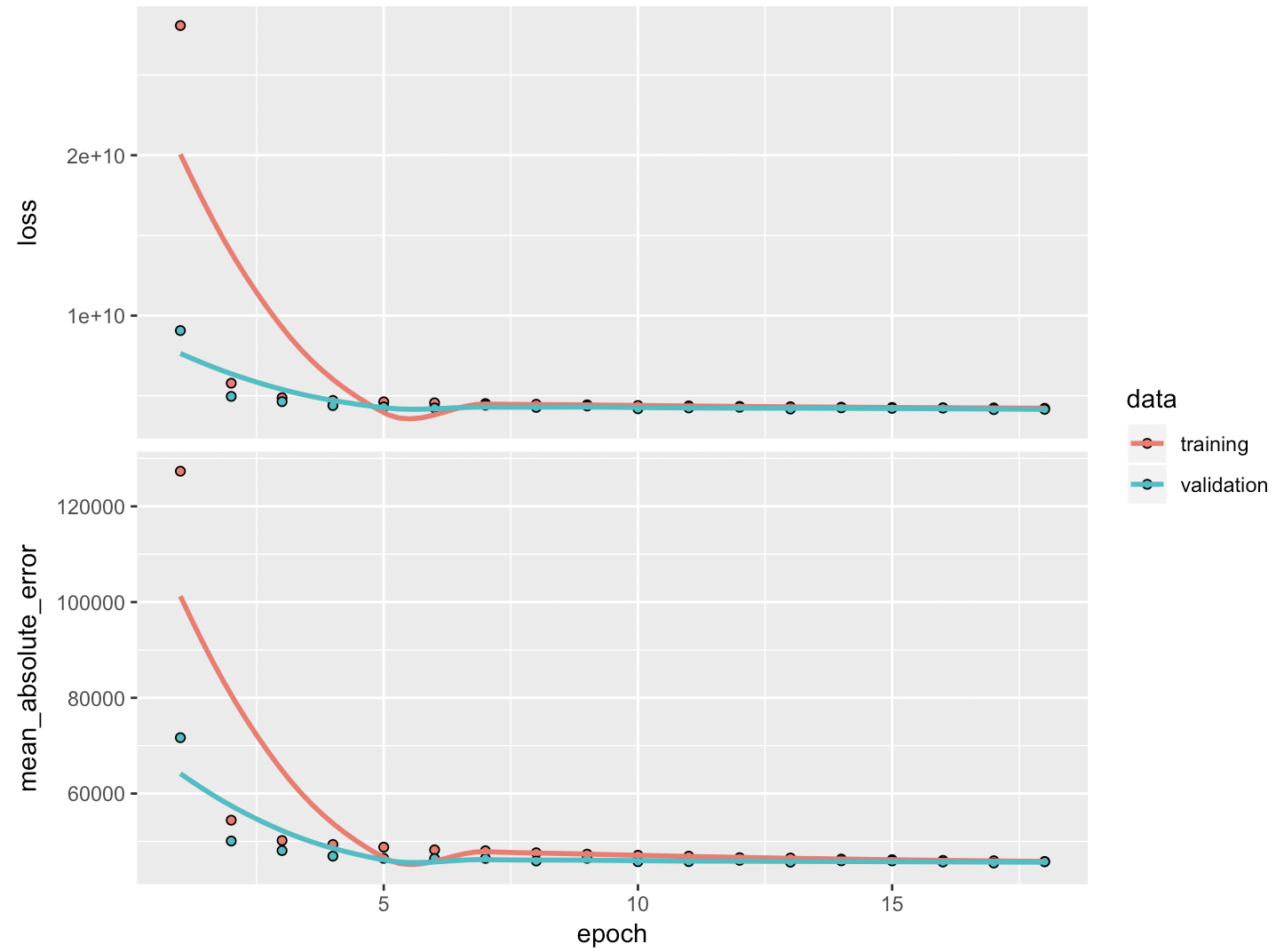
 batch\_size = 32

)

plot(history)

result <- model %>% evaluate(as.matrix(x\_test\_tbl), y\_test)

result



18. Run the best model on the test data ONE TIME ONLY using the evaluation function  
19. Enter the best-model test accuracy into your homework

$loss

[1] 4168879753

$mean\_absolute\_error

[1] 45553.36

20. Submit via Moodle