Model 1

July 1, 2023

Installing Libaries

[95]:
```
install.packages("keras")
install.packages("tensorflow")
install.packages("mlbench")
install.packages("magrittr")
install.packages("dplyr")
install.packages("neuralnet")

library(reticulate)
library(keras)
library(tensorflow)
library(mlbench)
library(dplyr)
library(magrittr)
library(neuralnet)
library(here)

reticulate::conda_install(packages = "graphviz")
reticulate::py_install("pydot", pip = TRUE)
prev_model = load_model_tf(filepath = here("model_out"))
test_predictions = predict(prev_model, test)
print(paste("The test R^2 value was: ", cor(testtarget, test_predictions) ^ 2))
```

```
Warning message:
"package 'keras' is in use and will not be installed"
Warning message:
"package 'tensorflow' is in use and will not be installed"
Warning message:
"package 'mlbench' is in use and will not be installed"
Warning message:
"package 'magrittr' is in use and will not be installed"
Warning message:
"package 'dplyr' is in use and will not be installed"
Warning message:
"package 'neuralnet' is in use and will not be installed"
+ "C:/Users/ASUS/anaconda3/condabin/conda.bat" "install" "--yes" "--name"
"r-reticulate" "-c" "conda-forge" "graphviz"
```

Importing Data and Setting Up Traning Datatset

[96]:
```
dat <- read.csv('https://skewthescript.org/s/four_year_colleges.csv')
colnames(dat)

# set training data to be 80% of all colleges
train_size <- floor(0.8 * nrow(dat))

## sample row indeces
set.seed(123)
train_ind <- sample(seq_len(nrow(dat)), size = train_size)

train <- dat[train_ind, ]
test <- dat[-train_ind, ]
head(train)
```

1. 'OPEID' 2. 'name' 3. 'city' 4. 'state' 5. 'region' 6. 'median_debt' 7. 'default_rate' 8. 'highest_degree' 9. 'ownership' 10. 'locale' 11. 'hbcu' 12. 'admit_rate' 13. 'SAT_avg' 14. 'online_only' 15. 'enrollment' 16. 'net_price' 17. 'avg_cost' 18. 'net_tuition' 19. 'ed_spending_per_student' 20. 'avg_faculty_salary' 21. 'pct_PELL' 22. 'pct_fed_loan' 23. 'grad_rate' 24. 'pct_firstgen' 25. 'med_fam_income' 26. 'med_alum_earnings'

A data.frame: 6 × 26

	OPEID <int>	name <chr>	city <chr>	state <chr>	region <chr>
415	231400	Saginaw Valley State University	University Center	MI	Midwest
463	908900	Hannibal-LaGrange University	Hannibal	MO	Midwest
179	160400	Young Harris College	Young Harris	GA	South
526	264200	The College of New Jersey	Ewing	NJ	Northeast
195	169400	Chicago State University	Chicago	IL	Midwest
938	370200	Averett University	Danville	VA	South

[97]:
```
dat <- train
dat_test <- test
```

Pre-processing

[98]:
```
chr_cols <- c(colnames(dat[,sapply(dat,is.character)]))[0:-2] #gets all columns␣
 ↪with datatype "character", excludes "name" and "city" columns

#For traning datatset
for (i in chr_cols){
    chr_dat <- dat[i]
    chr_dat <- chr_dat %>% group_by_at(i) %>% mutate(id=cur_group_id())
    dat[paste(i,"_id", sep = "")] <- chr_dat["id"]
}

#For test
for (i in chr_cols){
    chr_test <- dat_test[i]
```

```
    chr_test <- chr_test %>% group_by_at(i) %>% mutate(id=cur_group_id())
    dat_test[paste(i,"_id", sep = "")] <- chr_test["id"]
}
```

[99]:
```
print('Training:')
str(dat)
print('Test:')
str(dat_test)
```

```
[1] "Training:"
'data.frame':   842 obs. of  33 variables:
 $ OPEID                   : int  231400 908900 160400 264200 169400 370200
3070900 301400 679100 1026600 …
 $ name                    : chr  "Saginaw Valley State University" "Hannibal-
LaGrange University" "Young Harris College" "The College of New Jersey" …
 $ city                    : chr  "University Center" "Hannibal" "Young Harris"
"Ewing" …
 $ state                   : chr  "MI" "MO" "GA" "NJ" …
 $ region                  : chr  "Midwest" "Midwest" "South" "Northeast" …
 $ median_debt             : num  18.1 15 12 21 22 …
 $ default_rate            : num  4.8 6.9 3.5 1.3 8.7 5.3 2 3.8 7.2 13.7 …
 $ highest_degree          : chr  "Graduate" "Graduate" "Graduate" "Graduate" …
 $ ownership               : chr  "Public" "Private nonprofit" "Private
nonprofit" "Public" …
 $ locale                  : chr  "Suburb" "Town" "Rural" "Suburb" …
 $ hbcu                    : chr  "No" "No" "No" "No" …
 $ admit_rate              : num  89.5 64.6 65 51.2 46.4 …
 $ SAT_avg                 : int  1086 1120 1065 1240 887 984 1120 1176 1213 1057
…
 $ online_only             : chr  "No" "No" "No" "No" …
 $ enrollment              : int  6953 559 923 7039 1683 881 193 2843 3528 282
…
 $ net_price               : num  14.3 20.9 20.8 28.2 12.7 …
 $ avg_cost                : num  22.4 36.1 44.2 35.5 21.9 …
 $ net_tuition             : num  9.52 9.16 8.39 12.92 7.09 …
 $ ed_spending_per_student: num  7.7 5.35 8.12 10.56 20.86 …
 $ avg_faculty_salary      : num  8.43 4.99 5.56 10.9 8.12 …
 $ pct_PELL                : num  34.6 31.8 22.8 17.8 61.7 …
 $ pct_fed_loan            : num  57.4 47.9 39.7 50.8 80.5 …
 $ grad_rate               : num  47.9 42.9 43.8 86.5 16.2 …
 $ pct_firstgen            : num  32.9 45.7 28.2 20.5 42 …
 $ med_fam_income          : num  52.7 39.9 55.7 106.4 15.1 …
 $ med_alum_earnings       : num  46.2 37.5 40.5 65.5 40.5 …
 $ state_id                : int  23 25 11 32 15 47 25 36 35 37 …
 $ region_id               : int  2 2 5 3 2 5 2 2 3 4 …
 $ highest_degree_id       : int  2 2 2 2 2 1 1 2 2 2 …
 $ ownership_id            : int  3 2 2 3 3 2 2 2 3 2 …
 $ locale_id               : int  4 5 2 4 1 5 3 4 4 4 …
```

```
 $ hbcu_id                : int  1 1 1 1 1 1 1 1 1 1 …
 $ online_only_id         : int  1 1 1 1 1 1 1 1 1 1 …
[1] "Test:"
'data.frame':   211 obs. of  33 variables:
 $ OPEID                  : int  100200 105500 100900 102400 103600 105700
147902 110800 108600 109700 …
 $ name                   : chr  "Alabama A & M University" "University of
Alabama in Huntsville" "Auburn University" "University of West Alabama" …
 $ city                   : chr  "Normal" "Huntsville" "Auburn" "Livingston" …
 $ state                  : chr  "AL" "AL" "AL" "AL" …
 $ region                 : chr  "South" "South" "South" "South" …
 $ median_debt            : num  15.2 14 17.5 12.5 15.9 …
 $ default_rate           : num  12.1 4.7 2.6 5.8 2 6.3 4.1 4.1 13.6 3.1 …
 $ highest_degree         : chr  "Graduate" "Graduate" "Graduate" "Graduate" …
 $ ownership              : chr  "Public" "Public" "Public" "Public" …
 $ locale                 : chr  "Small City" "Small City" "Small City" "Rural"
…
 $ hbcu                   : chr  "Yes" "No" "No" "No" …
 $ admit_rate             : num  89.7 77.1 85.1 93.1 84 …
 $ SAT_avg                : int  959 1300 1302 1035 1219 1162 1238 1236 900 1207
…
 $ online_only            : chr  "No" "No" "No" "No" …
 $ enrollment             : int  5090 7825 24368 2247 3573 8787 2959 22624 2440
3492 …
 $ net_price              : num  15.5 17.2 24 16.7 30.1 …
 $ avg_cost               : num  23.4 24.9 32.2 22.6 49.6 …
 $ net_tuition            : num  8.1 8.28 16.86 8.29 20.06 …
 $ ed_spending_per_student: num  4.84 8.32 8.36 6.69 15.37 …
 $ avg_faculty_salary     : num  7.6 9.7 10.72 6.55 9.01 …
 $ pct_PELL               : num  71 24 13.4 53.5 11.4 …
 $ pct_fed_loan           : num  75 38.5 29.8 67.3 30.8 …
 $ grad_rate              : num  28.7 57.1 78.7 40.1 77.2 …
 $ pct_firstgen           : num  36.6 31 17.3 40.3 13.8 …
 $ med_fam_income         : num  23.6 44.8 72 25.6 87.7 …
 $ med_alum_earnings      : num  36.3 54.4 56.9 35.8 53.2 …
 $ state_id               : int  1 1 1 1 1 1 3 2 2 2 …
 $ region_id              : int  5 5 5 5 5 5 4 5 5 5 …
 $ highest_degree_id      : int  2 2 2 2 2 2 2 2 2 2 …
 $ ownership_id           : int  2 2 2 2 1 2 1 2 2 1 …
 $ locale_id              : int  3 3 3 2 4 3 2 3 3 5 …
 $ hbcu_id                : int  2 1 1 1 1 1 1 1 2 1 …
 $ online_only_id         : int  1 1 1 1 1 1 1 1 1 1 …
```

Set Up Traning Dataset (part 2)

[100]:
```
cols <- colnames(dat[,sapply(dat,is.numeric)]) #selects all columns with␣
 ↪numeric data
```

```
cols <- c(cols[c(-1, -3, -length(cols))]) #excludes OPEID (index 1),␣
 ↪default_rate(index 3), and online_only_id (index = length(cols)).
                                          #These are all the response variable␣
 ↪columns
cols
```

1. 'median_debt' 2. 'admit_rate' 3. 'SAT_avg' 4. 'enrollment' 5. 'net_price' 6. 'avg_cost' 7. 'net_tuition' 8. 'ed_spending_per_student' 9. 'avg_faculty_salary' 10. 'pct_PELL' 11. 'pct_fed_loan' 12. 'grad_rate' 13. 'pct_firstgen' 14. 'med_fam_income' 15. 'med_alum_earnings' 16. 'state_id' 17. 'region_id' 18. 'highest_degree_id' 19. 'ownership_id' 20. 'locale_id' 21. 'hbcu_id'

[101]:
```
training <- dat[, cols] #store as traning for all rows where 1 was picked, 70%␣
 ↪of dat
test <- dat_test[, cols] #store as test for all rows where 2 was picked, 30% of␣
 ↪dat
trainingtarget <- dat[, "default_rate"] #gets "answers" for test and train sets
testtarget <- dat_test[, "default_rate"]

#convert data into matrices
training <- as.matrix(training)
dimnames(training) <- NULL
test <- as.matrix(test)
dimnames(test) <- NULL
trainingtarget <- as.matrix(trainingtarget)
dimnames(trainingtarget) <- NULL
testtarget <- as.matrix(testtarget)
dimnames(testtarget) <- NULL
```

Normalizing Data

[102]:
```
m <- colMeans(training)
s <- apply(training, 2, sd)
training <- scale(training, center = m, scale = s)
test <- scale(test, center = m, scale = s)
summary(training)
```

```
       V1                 V2                 V3                 V4
 Min.   :-3.30717   Min.   :-3.3179   Min.   :-2.3270   Min.   :-0.70835
 1st Qu.:-0.66040   1st Qu.:-0.4901   1st Qu.:-0.6597   1st Qu.:-0.57209
 Median :-0.04799   Median : 0.2001   Median :-0.1783   Median :-0.41927
 Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000
 3rd Qu.: 0.75127   3rd Qu.: 0.6883   3rd Qu.: 0.4909   3rd Qu.: 0.08554
 Max.   : 2.67628   Max.   : 1.5206   Max.   : 3.2148   Max.   : 6.76383
       V5                 V6                 V7                 V8
 Min.   :-2.3826   Min.   :-1.6025   Min.   :-1.7114   Min.   :-1.0013
 1st Qu.:-0.6667   1st Qu.:-0.8799   1st Qu.:-0.7367   1st Qu.:-0.4567
 Median :-0.1062   Median :-0.1498   Median :-0.1773   Median :-0.2362
```

```
Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
3rd Qu.: 0.5596   3rd Qu.: 0.6702   3rd Qu.: 0.4336   3rd Qu.: 0.1018
Max.   : 3.5677   Max.   : 2.4982   Max.   : 3.7836   Max.   :13.0655
      V9                V10               V11               V12
Min.   :-2.1495   Min.   :-1.78316   Min.   :-2.88420   Min.   :-3.01005
1st Qu.:-0.6971   1st Qu.:-0.74794   1st Qu.:-0.65940   1st Qu.:-0.72366
Median :-0.2277   Median :-0.05665   Median : 0.02626   Median :-0.03814
Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.00000
3rd Qu.: 0.5293   3rd Qu.: 0.54793   3rd Qu.: 0.74568   3rd Qu.: 0.65665
Max.   : 4.8891   Max.   : 3.61232   Max.   : 2.34807   Max.   : 2.41408
     V13               V14               V15               V16
Min.   :-2.2517   Min.   :-1.9564   Min.   :-2.1147   Min.   :-1.87489
1st Qu.:-0.7744   1st Qu.:-0.7634   1st Qu.:-0.6781   1st Qu.:-0.87778
Median : 0.1027   Median :-0.2061   Median :-0.2119   Median : 0.01438
Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000
3rd Qu.: 0.7058   3rd Qu.: 0.6304   3rd Qu.: 0.3962   3rd Qu.: 0.78408
Max.   : 3.1778   Max.   : 3.2229   Max.   : 5.1818   Max.   : 1.76370
     V17               V18               V19               V20
Min.   :-1.6250   Min.   :-2.5128   Min.   :-2.8003   Min.   :-1.5691
1st Qu.:-0.8898   1st Qu.: 0.3975   1st Qu.:-0.7987   1st Qu.:-0.8741
Median :-0.1546   Median : 0.3975   Median :-0.7987   Median :-0.1791
Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
3rd Qu.: 1.3159   3rd Qu.: 0.3975   3rd Qu.: 1.2028   3rd Qu.: 1.0371
Max.   : 2.0511   Max.   : 0.3975   Max.   : 1.2028   Max.   : 1.2109
     V21
Min.   :-0.2261
1st Qu.:-0.2261
Median :-0.2261
Mean   : 0.0000
3rd Qu.:-0.2261
Max.   : 4.4174
```

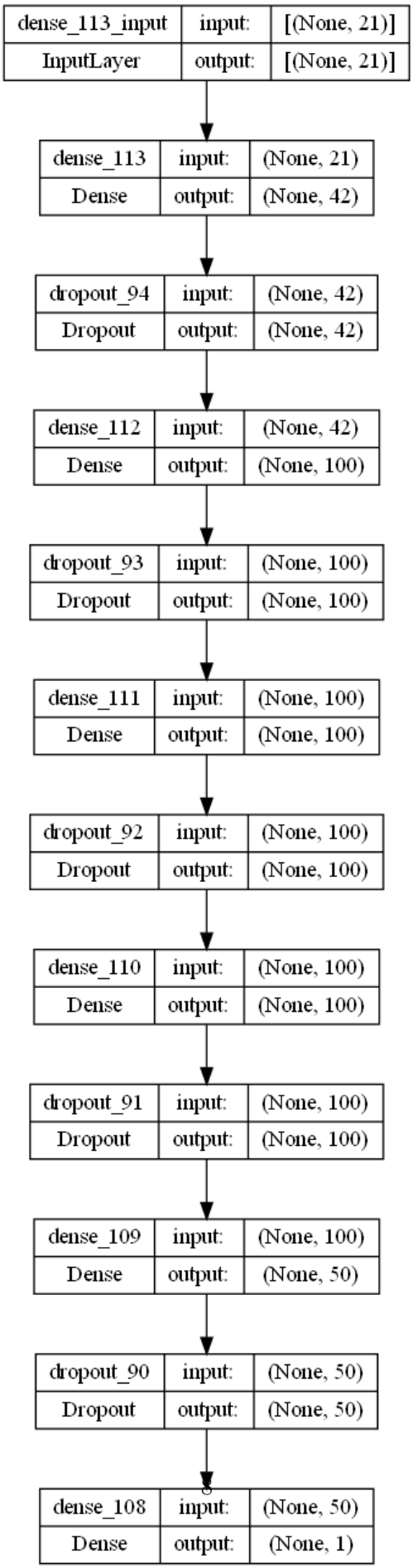
Model Creation

[178]:
```
embedding_size = min(50, length(cols)/2)
model <- keras_model_sequential()
model %>%
        layer_dense(units = length(cols)*2, activation = 'relu', input_shape =␣
 ↪c(length(cols))) %>%
        layer_dropout(rate=0.4)  %>%
        layer_dense(units = 100, activation = 'relu') %>%
        layer_dropout(rate=0.4)  %>%
        layer_dense(units = 100, activation = 'relu') %>%
        layer_dropout(rate=0.4)  %>%
        layer_dense(units = 100, activation = 'relu') %>%
        layer_dropout(rate=0.4)  %>%
        layer_dense(units = 50, activation = 'relu')  %>%
        layer_dropout(rate=0.2)  %>%
```

```
        layer_dense(units = 1)
```

Visual Representation of Model

[179]:
```
plot(model,
     show_shapes = T,
     dpi = 96,
     to_file = here("out.png"))
```

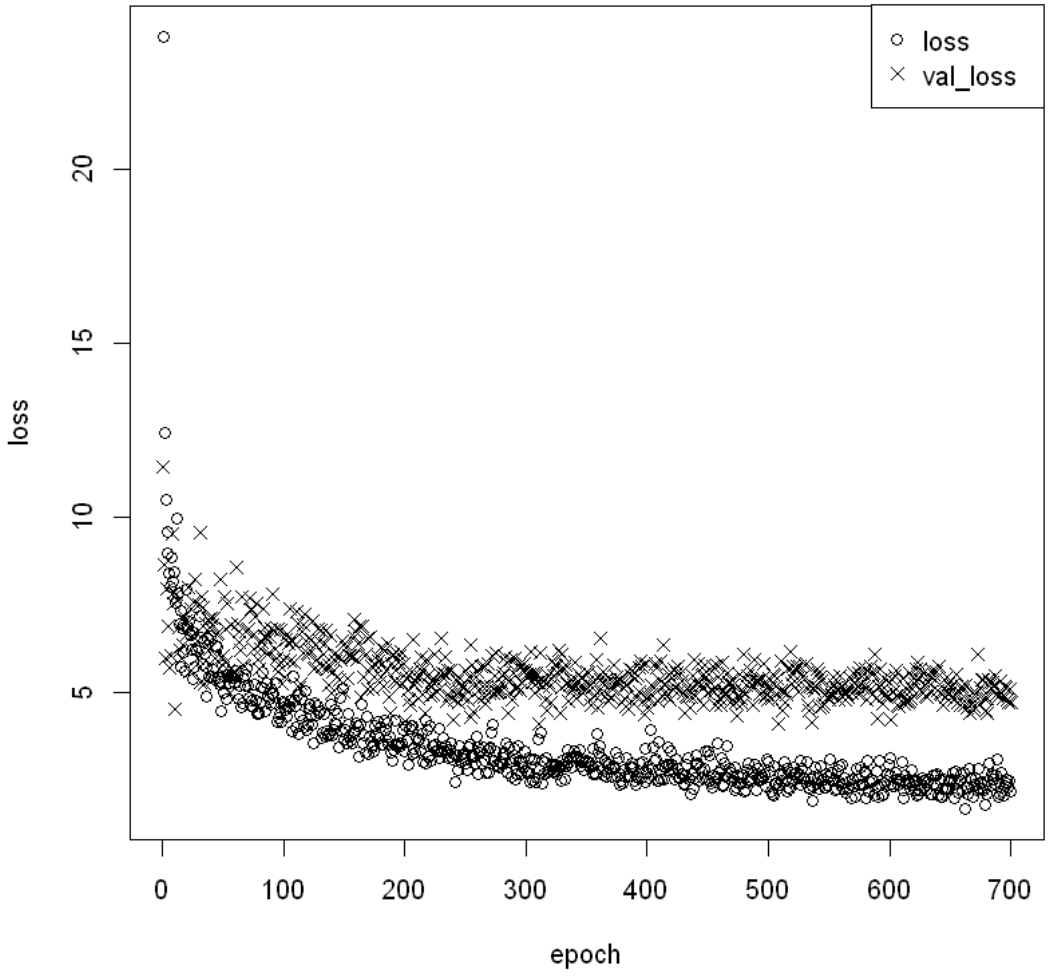

dense_113_input | input: | [(None, 21)]
InputLayer | output: | [(None, 21)]
dense_113 | input: | (None, 21)
Dense | output: | (None, 42)
dropout_94 | input: | (None, 42)
Dropout | output: | (None, 42)
dense_112 | input: | (None, 42)
Dense | output: | (None, 100)
dropout_93 | input: | (None, 100)
Dropout | output: | (None, 100)
dense_111 | input: | (None, 100)
Dense | output: | (None, 100)
dropout_92 | input: | (None, 100)
Dropout | output: | (None, 100)
dense_110 | input: | (None, 100)
Dense | output: | (None, 100)
dropout_91 | input: | (None, 100)
Dropout | output: | (None, 100)
dense_109 | input: | (None, 100)
Dense | output: | (None, 50)
dropout_90 | input: | (None, 50)
Dropout | output: | (None, 50)
dense_108 | input: | (None, 50)
Dense | output: | (None, 1)

Model Compilation

[180]:
```
model %>% compile(loss = 'mse',
optimizer = 'rmsprop',
metrics = 'mae')
```

Model Fitting

[181]:
```
mymodel <- model %>%
fit(training,trainingtarget,
              epochs = 700,
              verbose = 2,
              batch_size = 32,
              validation_split = 0.2) %>% plot(metrics = c("loss"))
```

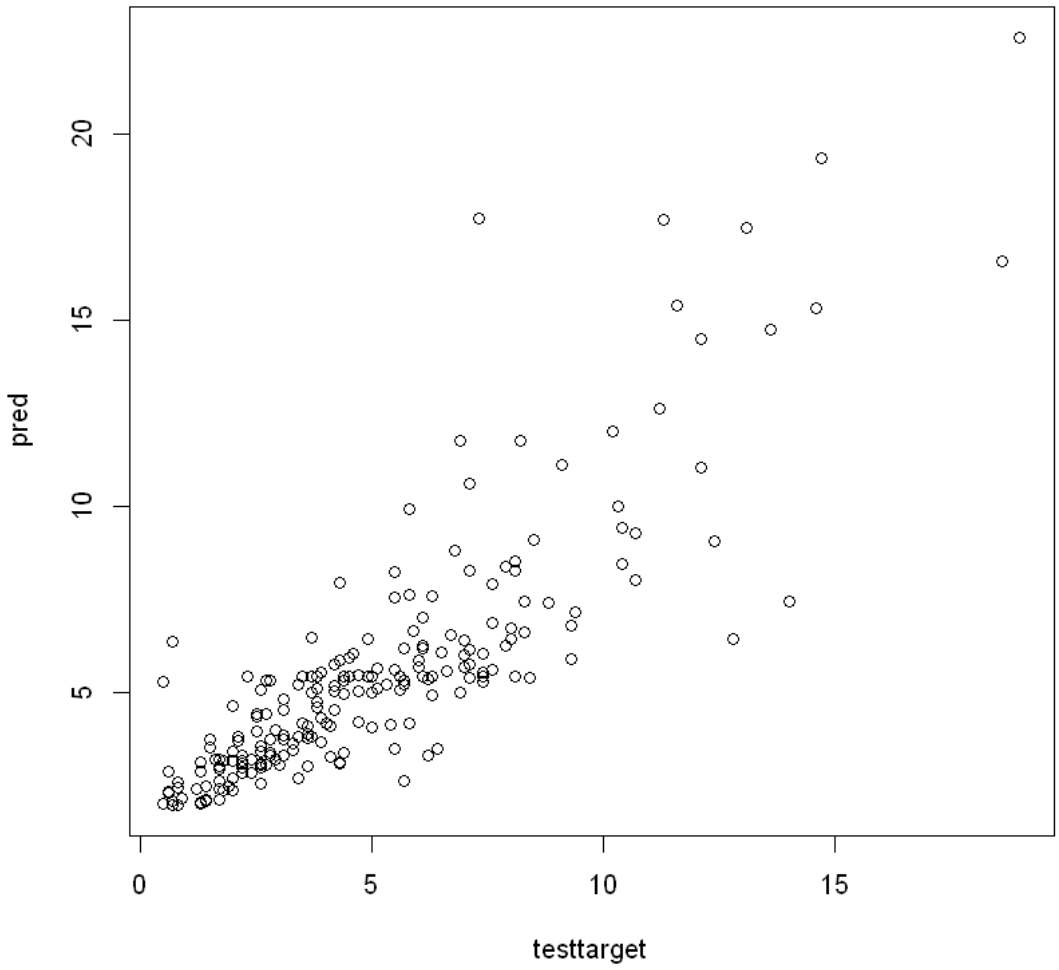


Model Validation (!!Not Final Submission!!)

[182]:
```
model %>% evaluate(test, testtarget)
pred <- model %>% predict(test)
```

loss 3.85272359848022 **mae** 1.43750381469727

[183]:
```
plot(testtarget, pred)
```



[184]:
```
# run this code to get the R^2 value on the test set from your model
test_predictions = predict(model, test)
print(paste("The test R^2 value was: ", cor(testtarget, test_predictions) ^ 2))
```

```
[1] "The test R^2 value was:  0.710979330176"
```

[185]:
```
rsq = cor(testtarget, test_predictions) ^ 2
if (rsq > prev){
    prev = rsq
    model %>% save_model_tf(filepath = here("model_out"))
    print(rsq)
}
```

Final Submission

[186]:
```
final_model <- load_model_tf(filepath = here("model_out"))
test_predictions = predict(final_model, test)
print(paste("The test R^2 value was: ", cor(testtarget, test_predictions) ^ 2))
```

```
[1] "The test R^2 value was:  0.71472931744183"
```