Лабораторная работа 1. Регрессионный анализ данных.

Сокуров Р.Е.

13.09.2024

Задание

Целью данной работы является формирование у магистрантов знаний в области статистики и навыков программирования на языке программирования R при решении регрессионных задач на основе набора данных.

1. Изучите Статью https://www.r-bloggers.com/2015/09/fitting-a-neural-network-in-r-neuralnet-package/.
2. Выполните расчеты по Статье, сравните два метода и проведите перекрестную проверку (cross validation) на языке программирования R.
3. Выполните аналогичные расчеты для примера выбранного самостоятельно (https://www.kaggle.com, datasetsearch) на языке программирования R.
4. Отчет сформируйте в RStudio в pdf-формате и прикрепите в качестве ответа.

Ход работы

1. Выполнение расчётов по статье.

1.1. Линейная регрессия

Линейная регрессия — это метод анализа данных, который предсказывает неизвестные данные с помощью известных значения данных. Он математически моделирует неизвестную или зависимую переменную и известную или независимую переменную в виде линейного уравнения.

Согласно статье, для работы был выбран датасет `Boston` из библиотеки `MASS` и выполнена проверка целостности данных:

```
set.seed(500) # установка зерна случайной генерации
library(MASS) # подключение библиотеки с датасетами
data <- Boston # в data кладём данные с датасета Boston

apply(data,2,function(x) sum(is.na(x))) # проверка целостности данных
```

```
##    crim      zn   indus    chas     nox      rm     age     dis     rad     tax 
##       0       0       0       0       0       0       0       0       0       0 
## ptratio   black   lstat    medv 
##       0       0       0       0
```

Затем полученный набор данных был разделён на обучающий и тестовый наборы в соотношении 75/25%:

```
index <- sample(1:nrow(data),round(0.75*nrow(data))) # выбираем случайные индексы 75% датасета
train <- data[index,] # 75% датасета для обучения
test <- data[-index,] # остальное (25%) для теста
```

После чего была создана модель линейной регрессии для предсказания средней стоимости дома в тысячах долларах (medv в дальнейшем), обучена и протестирована.

```
lm.fit <- glm(medv~., data=train) # создание модели линейной регрессии для предсказания medv
pr.lm <- predict(lm.fit, test) # кладём в pr.lm предсказывания на тестовом наборе
MSE.lm <- sum((pr.lm - test$medv)^2)/nrow(test) # считаем MSE между реальностью и предсказ. знач.
summary(lm.fit) # отчёт о полученной модели
```

```
## 
## Call:
## glm(formula = medv ~ ., data = train)
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  31.111702   5.459811   5.698 2.49e-08 ***
## crim         -0.111372   0.033256  -3.349 0.000895 ***
## zn            0.042633   0.014307   2.980 0.003077 ** 
## indus         0.001483   0.067455   0.022 0.982473    
## chas          1.756844   0.981087   1.791 0.074166 .  
## nox         -18.184847   4.471572  -4.067 5.84e-05 ***
## rm            4.760341   0.480472   9.908  < 2e-16 ***
## age          -0.013439   0.014101  -0.953 0.341190    
## dis          -1.553748   0.218929  -7.097 6.65e-12 ***
## rad           0.288181   0.072017   4.002 7.62e-05 ***
## tax          -0.013739   0.004060  -3.384 0.000791 ***
## ptratio      -0.947549   0.140120  -6.762 5.38e-11 ***
## black         0.009502   0.002901   3.276 0.001154 ** 
## lstat        -0.388902   0.059733  -6.511 2.47e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for gaussian family taken to be 20.23806)
## 
##     Null deviance: 32463.5  on 379  degrees of freedom
## Residual deviance:  7407.1  on 366  degrees of freedom
## AIC: 2237
## 
## Number of Fisher Scoring iterations: 2
```

1.2. Нейронная сеть

Нейронная сеть — математическая модель, а также ее программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма.

Перед созданием и обучением нейронной сети, была выполнена нормализация данных в датасете в диапазон [0, 1]:

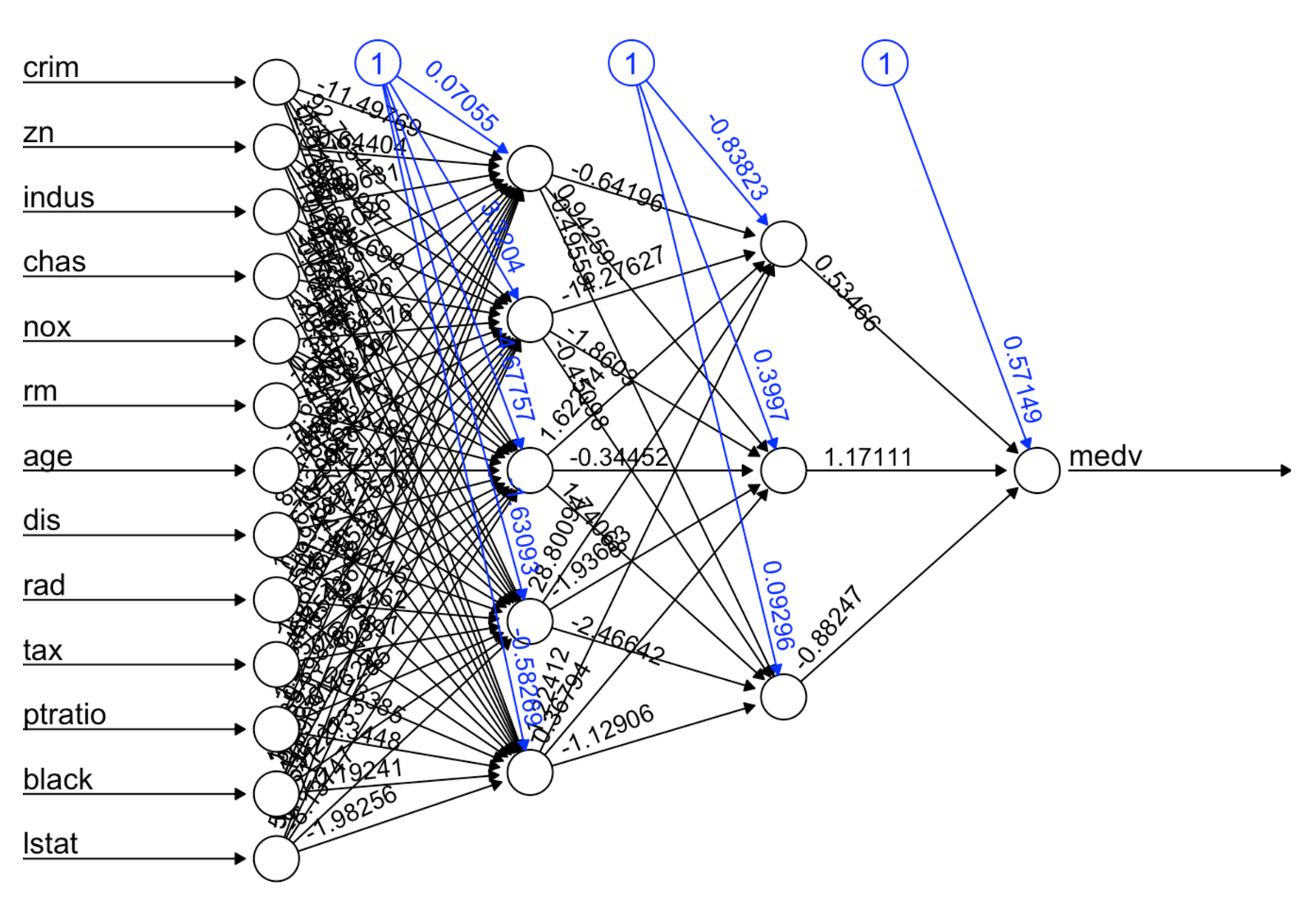
```
maxs <- apply(data, 2, max) # забираем из каждого столбца его max значение
mins <- apply(data, 2, min) # забираем из каждого столбца его min значение

scaled <- as.data.frame(scale(data, center = mins, scale = maxs - mins)) # нормализуем данные в интервал [0, 1]

train_ <- scaled[index,] # часть для обучения: нормализованные 75% сета
test_ <- scaled[-index,] # тест: нормализованные 25% сета
```

Затем была создана и обучена модель многослойного перцептрона (MLP)

```
library(neuralnet) # подключаем библиотеку с нейронной сетью
n <- names(train_) # копируем имена столбцов
f <- as.formula(paste("medv ~", paste(n[!n %in% "medv"], collapse = " + "))) # формируем формулу для предсказыван
ия medv на основе всех остальных данных
nn <- neuralnet(f,data=train_,hidden=c(5,3),linear.output=T) # создаём и обучаем MLP
plot(nn) # и выводим
```



Полученная модель была использована для предсказания средней стоимости дома в тысячах долларах (medv в дальнейшем). Также была посчитана MSE.

```
pr.nn <- compute(nn,test_[,1:13]) # прогноз нейросети на тестовой выборке на первых 13 столбцах (без medv)
pr.nn_ <- pr.nn$net.result*(max(data$medv)-min(data$medv))+min(data$medv) # возвращаем нормализованные [0, 1] пре
дсказания обратно [min, max]
test.r <- (test_$medv)*(max(data$medv)-min(data$medv))+min(data$medv) # то же самое, что и в прошлой строчке, тол
ько для medv
MSE.nn <- sum((test.r - pr.nn_)^2)/nrow(test_) # считаем MSE
```

1.3 Сравнение линейной регрессии и нейронной сети

Значения MSE двух моделей:

```
print(paste(MSE.lm,MSE.nn)) # выводим MSE LM и MSE NN
```

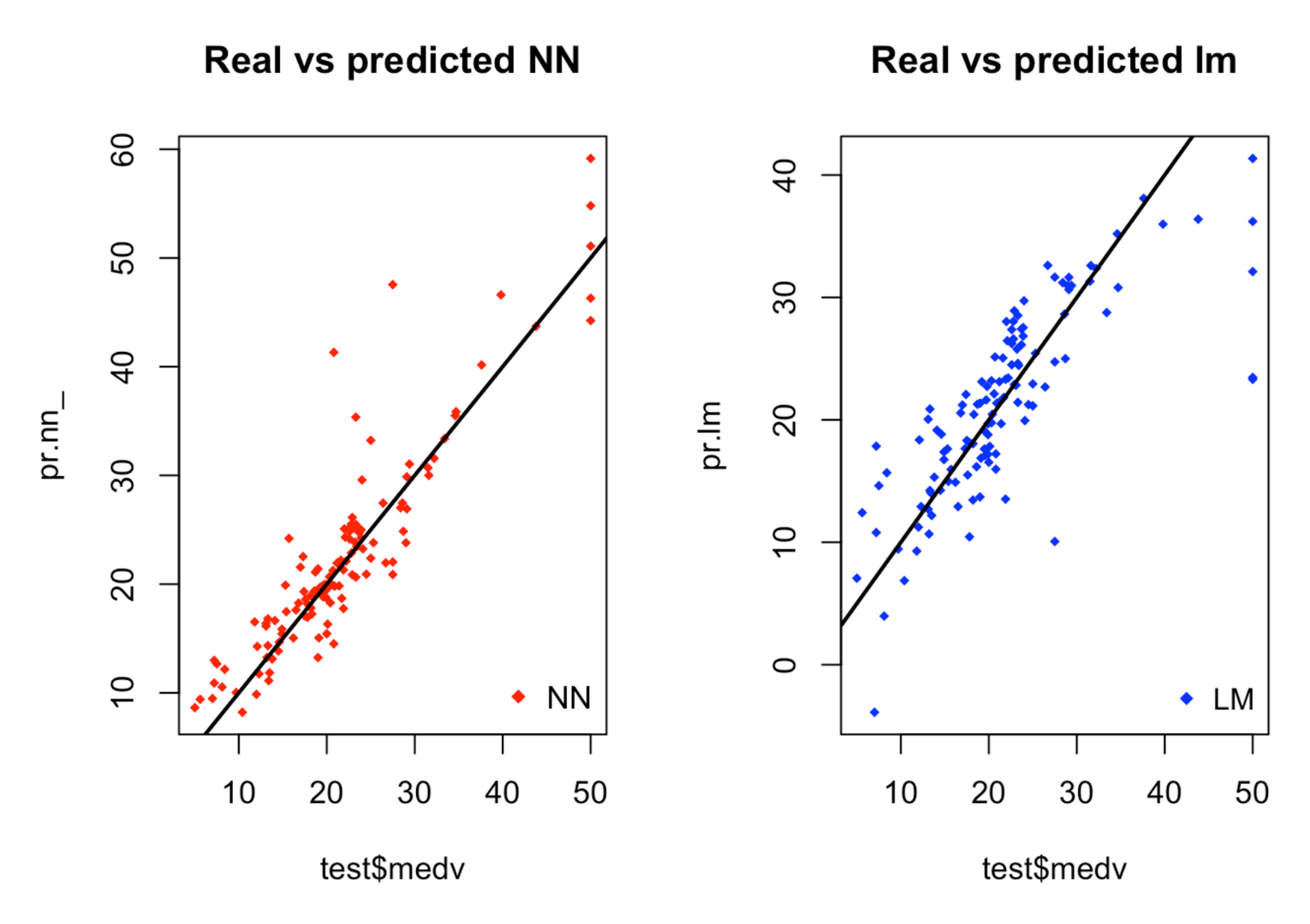
```
## [1] "31.2630222372615 16.4595537665717"
```

Затем были построены два графика (для каждой модели), где по оси X были отложены реальные значения medv, а по оси Y предсказанные значения. Таким образом, идеальную модель описывает чёрная линия на графиках.

```
par(mfrow=c(1,2)) # создаём в одном окне 2 графика

# Построение графиков, которые позволяют визуализировать,
# насколько близки прогнозы моделей к фактическим значениям.
plot(test$medv,pr.nn_,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN',pch=18,col='red', bty='n')

plot(test$medv,pr.lm,col='blue',main='Real vs predicted lm',pch=18, cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='LM',pch=18,col='blue', bty='n', cex=.95)
```



По итогам сравнения можно сделать вывод о том, что нейронная сеть лучше справляется с предсказанием значений medv, поскольку значение MSE у неё меньше, что также подтверждается визуальной интерпретацией полученных графиков.

1.4 Кросс-валидация линейной регрессионной модели

Кросс-валидация — это метод оценки производительности модели машинного обучения, используемый для проверки её способности обобщать данные.

```
library(boot) # загружаем библиотеку для кросс-валидации
set.seed(200) # установка зерна случайной генерации
lm.fit <- glm(medv~.,data=data) # вновь создаём модель линейной регрессии
cv.glm(data,lm.fit,K=10)$delta[1] # выполняем кросс-валидацию на 10 фолдах
```

```
## [1] 23.17094
```

1.5 Кросс-валидация нейронной сети

Примечание: для корректного отображения результаты выполнения следующих двух ячеек были скрыты из отчёта.

```
set.seed(450) # меняем зерно случайных чисел
cv.error <- NULL # создаём переменную для хранения ошибки кросс-валидации
k <- 10 # количество фолдов
library(plyr) # библиотека для отображения прогресс-баров
pbar <- create_progress_bar('text') # выбираем текстовый прогресс бар
pbar$init(k) # инициализируем его с максимальным количеством шагов равным k
```

```
for(i in 1:k){ # цикл с 1 до k
  index <- sample(1:nrow(data),round(0.9*nrow(data))) # делим выборку 90/10%
  train.cv <- scaled[index,] # 90%
  test.cv <- scaled[-index,] # 10%

  nn <- neuralnet(f,data=train.cv,hidden=c(5,2),linear.output=T) # обучаем нейр. сеть

  pr.nn <- compute(nn,test.cv[,1:13]) # вычисляем прогнозы на тестовых данных
  pr.nn <- pr.nn$net.result*(max(data$medv)-min(data$medv))+min(data$medv) # обратная нормализация

  test.cv.r <- (test.cv$medv)*(max(data$medv)-min(data$medv))+min(data$medv) # обратная нормализация

  cv.error[i] <- sum((test.cv.r - pr.nn)^2)/nrow(test.cv) # сохраняем MSE текущего фолда

  pbar$step() # шаг для прогресс-бара
}
```

После чего были выведены средняя ошибка кросс-валидации и весь набор ошибок на каждом шаге:

```
mean(cv.error) # считаем среднюю ошибку кросс-валидаци
```

```
## [1] 7.641292
```

```
cv.error # вывод полного вектора ошибок (ошибки каждого шага)
```

```
##  [1] 13.331937  7.099840  6.580337  5.697609  6.841745  5.771481 10.751406
##  [8]  5.384253  9.452109  5.502201
```

```
boxplot(cv.error,xlab='MSE CV',col='cyan',
        border='blue',names='CV error (MSE)',
        main='CV error (MSE) for NN',horizontal=TRUE)
```

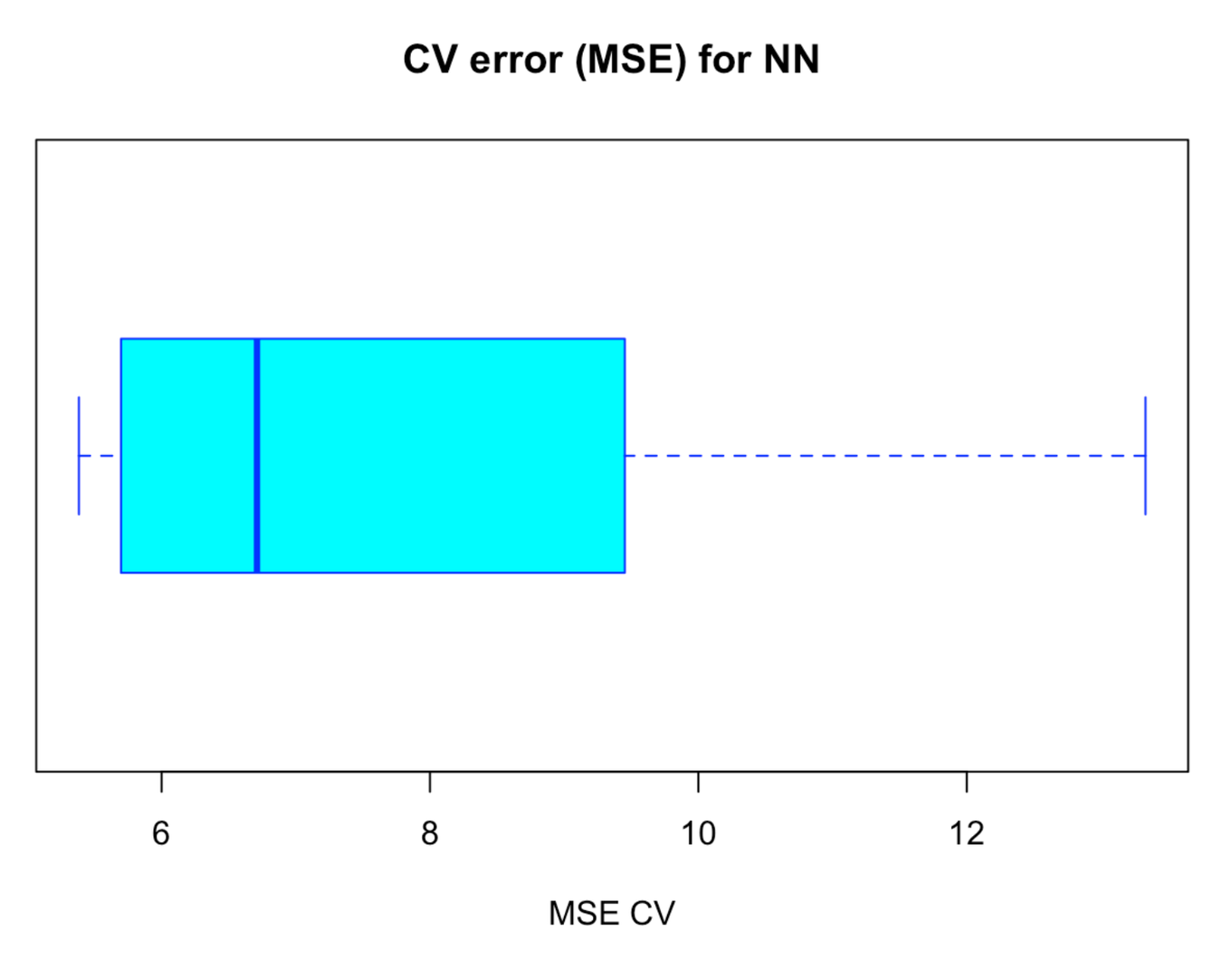


График показывает разброс ошибок MSE для нейронной сети, где медиана находится ближе к 7, а диапазон значений распределен от около 5 до 13. Левая граница прямоугольника находится в районе 6, что соответствует первому квартилю распределения. Правая граница прямоугольника находится в районе 10, что соотвествутет третьему квартилю распределения.

2. Выполнение расчётов по выбранной теме

2.1 Линейная регрессия

Для самостоятельной части работы был выбран датасет `Car information dataset` https://www.kaggle.com/datasets/tawfikelmetwally/automobile-dataset. Предполагается предсказание величины Acceleration - способности разгоняться, измеренной в секундах.

Перед созданием модели необходимо обработать данные: удалить ненужные столбцы (например столбец name характеризующий название автомобиля), а также удалить все строки, в которых отсутствуют данные.

```
data <- read.csv("Automobile.csv")

data <- data[, -c(1, ncol(data))] # удаляем ненужные стобцы типа названия автомобиля
data <- drop_na(data) # удаляем NA из датасета
```

Затем полученный набор данных был разделён на обучающий и тестовый наборы в соотношении 75/25%:

```
index <- sample(1:nrow(data),round(0.75*nrow(data))) # забираем 75% случайных индексов записей
train <- data[index,] # 75% всех индексов идут на обучение
test <- data[-index,] # оставшаяся часть (25%) на тест
```

После чего была создана модель линейной регрессии для предсказания Acceleration, обучена и протестирована.

```
lm.fit <- glm(acceleration~., data=train) # собираем модель линейной регрессии и обучаем её на train сете
summary(lm.fit) # отчёт о модели
```

```
## 
## Call:
## glm(formula = acceleration ~ ., data = train)
## 
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  21.6435807  2.5385658   8.526  8.8e-16 ***
## mpg           0.0272629  0.0291634   0.935   0.3507    
## cylinders    -0.1675343  0.2101645  -0.797   0.4260    
## displacement -0.0094996  0.0042996  -2.209   0.0279 *  
## horsepower   -0.0839583  0.0061365 -13.682  < 2e-16 ***
## weight        0.0034583  0.0003906   8.853  < 2e-16 ***
## model_year   -0.0718884  0.0375683  -1.914   0.0567 .  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for gaussian family taken to be 3.091726)
## 
##     Null deviance: 2241.39  on 293  degrees of freedom
## Residual deviance:  887.33  on 287  degrees of freedom
## AIC: 1175.1
## 
## Number of Fisher Scoring iterations: 2
```

```
pr.lm <- predict(lm.fit, test) # кладём в переменную предсказываемые lm значения acceleration
MSE.lm <- sum((pr.lm - test$acceleration)^2)/nrow(test) # считаем MSE
```

2.2. Нейронная сеть

Перед созданием и обучением нейронной сети, была выполнена нормализация данных в датасете в диапазон [0, 1]:

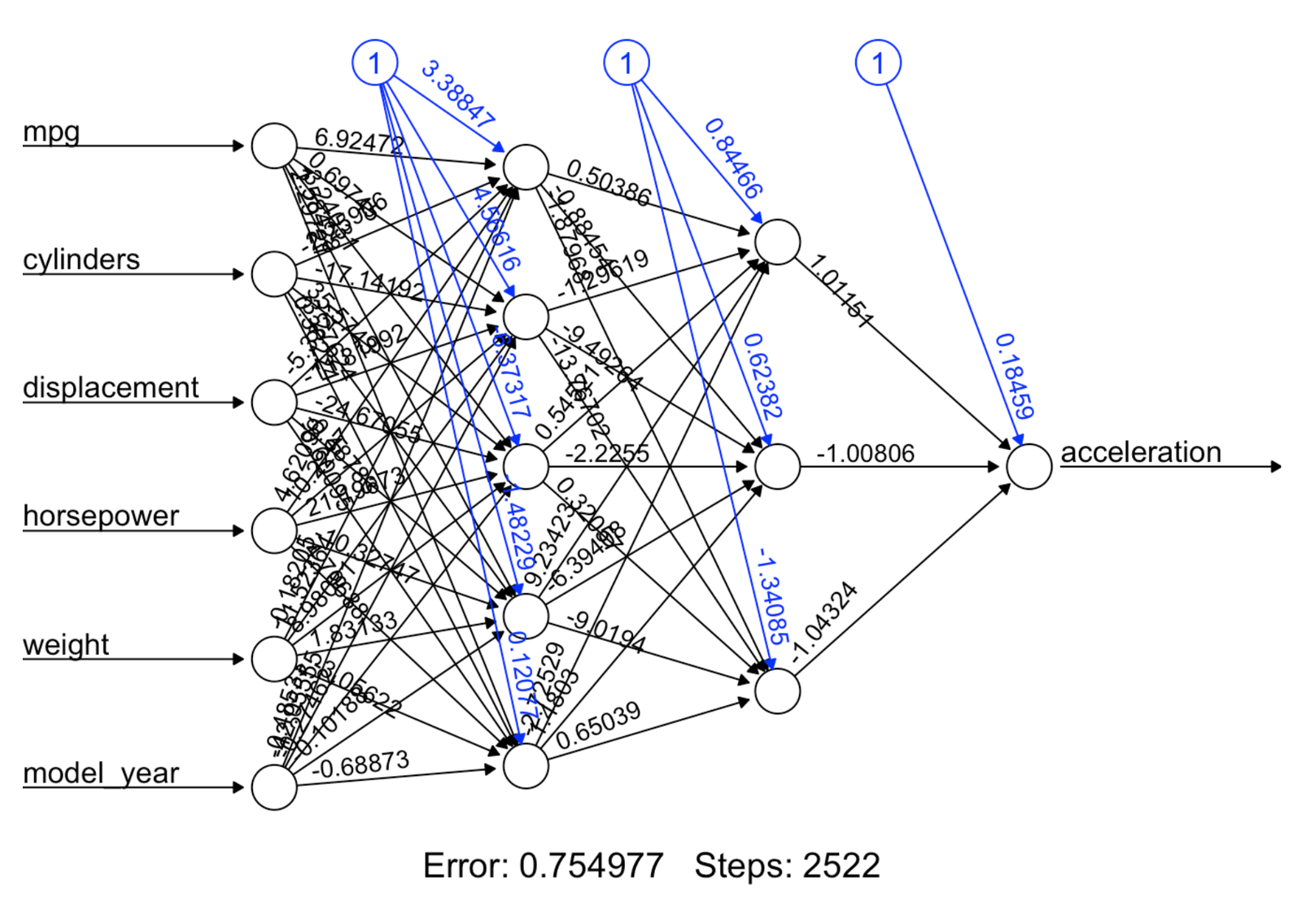
```
maxs <- apply(data, 2, max) # забираем из каждого столбца его max значение
mins <- apply(data, 2, min) # забираем из каждого столбца его min значение

scaled <- as.data.frame(scale(data, center = mins, scale = maxs - mins)) # нормализуем данные в интервал [0, 1]

train_ <- scaled[index,] # часть для обучения: нормализованные 75% сета
test_ <- scaled[-index,] # тест: нормализованные 25% сета
```

Затем была создана и обучена модель многослойного перцептрона (MLP)

```
library(neuralnet) # подключение библиотеки с нейросетями
n <- names(train_) # n - имена всех столбцов датасета
f <- as.formula(paste("acceleration ~", paste(n[!n %in% "acceleration"], collapse = " + "))) # генерация формулы
из string
nn <- neuralnet(f,data=train_,hidden=c(5,3),linear.output=T) # создаём MLP и обучаем его на train
```



Полученная модель была использована для предсказания acceleration. Также была посчитана MSE.

```
test_data <- test_[, -which(names(test_) == "acceleration")]# выбираем все столбцы кроме acceleration

pr.nn <- neuralnet::compute(nn, test_data)
pr.nn_ <- pr.nn$net.result*(max(data$acceleration)-min(data$acceleration))+min(data$acceleration) # возвращаем но
рмализованные [0, 1] предсказания обратно [min, max]
test.r <- (test_$acceleration)*(max(data$acceleration)-min(data$acceleration))+min(data$acceleration) # то же сам
ое, что и в прошлой строчке, только для quality
MSE.nn <- sum((test.r - pr.nn_)^2)/nrow(test_) # считаем MSE
```

2.3 Сравнение линейной регрессии и нейронной сети

Значения MSE двух моделей:

```
print(paste(MSE.lm,MSE.nn)) # выводим MSE LM и MSE NN
```

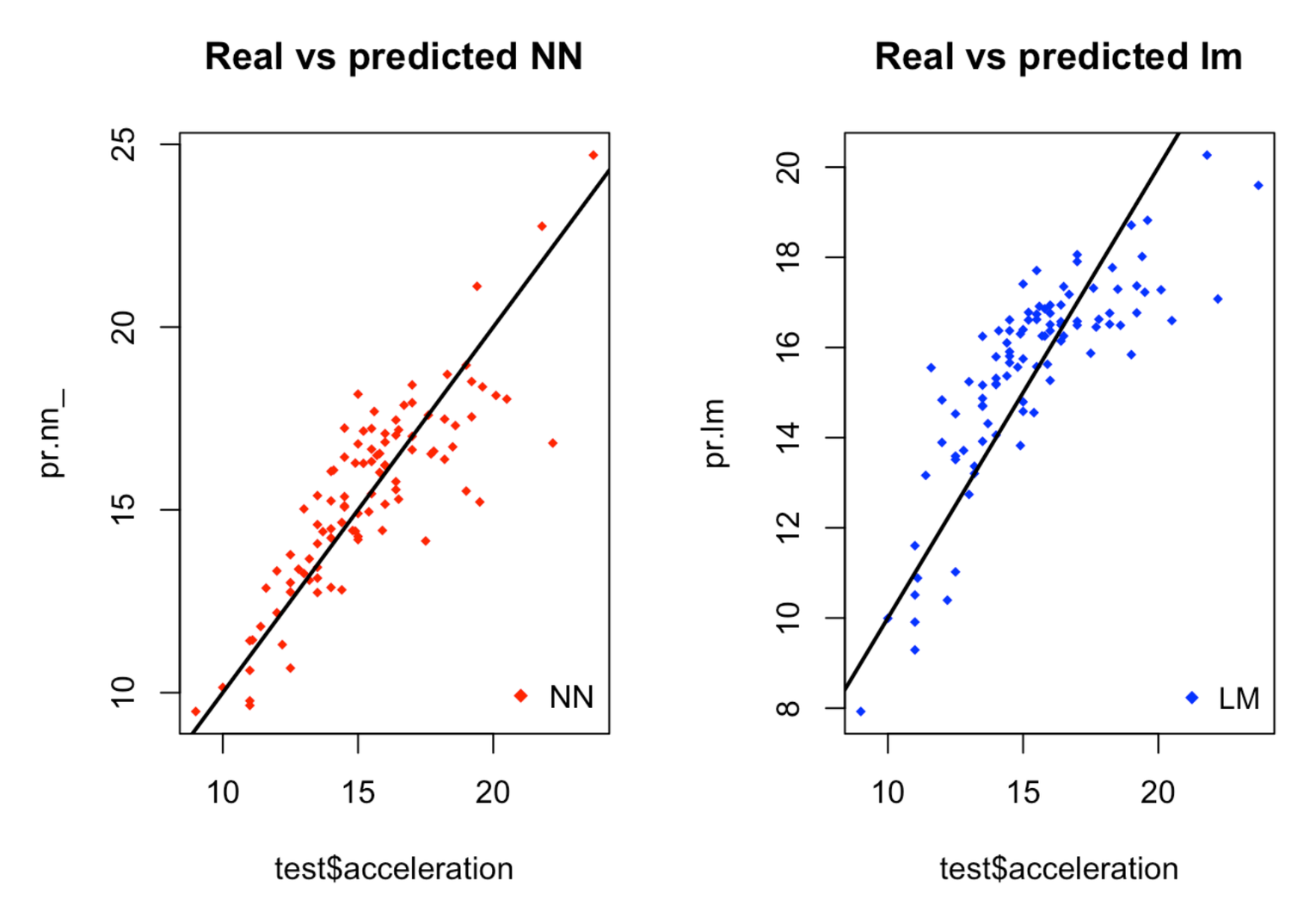
```
## [1] "2.51568941727259 2.03437916753043"
```

Затем были построены два графика (для каждой модели), где по оси X были отложены реальные значения acceleration, а по оси Y предсказанные значения. Таким образом, идеальную модель описывает чёрная линия на графиках.

```
par(mfrow=c(1,2)) # создаём в одном окне 2 графика

# Построение графиков, которые позволяют визуализировать,
# насколько близки прогнозы моделей к фактическим значениям.
plot(test$acceleration,pr.nn_,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN',pch=18,col='red', bty='n')

plot(test$acceleration,pr.lm,col='blue',main='Real vs predicted lm',pch=18, cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='LM',pch=18,col='blue', bty='n', cex=.95)
```



По итогам сравнения можно сделать вывод о том, что нейронная сеть лучше справляется с предсказанием значений Acceleration, поскольку значение MSE у неё меньше, что также подтверждается визуальной интерпретацией полученных графиков. ### 2.4 Кросс-валидация линейной регрессионной модели

```
library(boot) # загружаем библиотеку для кросс-валидации
set.seed(200) # устанавливаем зерно случайных чисел
lm.fit <- glm(acceleration~.,data=data) # вновь создаём модель линейной регрессии
cv.glm(data,lm.fit,K=10)$delta[1] # выполняем кросс-валидацию на 10 фолдах
```

```
## [1] 3.078596
```

2.5 Кросс-валидация нейронной сети

Примечание: для корректного отображения результаты выполнения следующих двух ячеек были скрыты из отчёта.

```
set.seed(450) # меняем зерно случайных чисел
cv.error <- NULL # создаём переменную для хранения ошибки кросс-валидации
k <- 10 # количество фолдов
library(plyr) # библиотека для отображения прогресс-баров
pbar <- create_progress_bar('text') # выбираем текстовый прогресс бар
pbar$init(k) # инициализируем его с максимальным количеством шагов равным k
```

```
for(i in 1:k){ # цикл с 1 до k
  index <- sample(1:nrow(data),round(0.9*nrow(data))) # делим выборку 90/10%
  train.cv <- scaled[index,] # 90%
  test.cv <- scaled[-index,] # 10%

  test_data <- test.cv[, -which(names(test_) == "acceleration")]
  nn <- neuralnet(f,data=train.cv,hidden=c(5,2),linear.output=T) # обучаем нейр. сеть
  pr.nn <- neuralnet::compute(nn,test_data) # вычисляем прогнозы на тестовых данных
  pr.nn <- pr.nn$net.result*(max(data$acceleration)-min(data$acceleration))+min(data$acceleration) # обратная нор
мализация

  test.cv.r <- (test.cv$acceleration)*(max(data$acceleration)-min(data$acceleration))+min(data$acceleration) # об
ратная нормализация

  cv.error[i] <- sum((test.cv.r - pr.nn)^2)/nrow(test.cv) # сохраняем MSE текущего фолда

  pbar$step() # шаг для прогресс-бара
}
```

После чего были выведены средняя ошибка кросс-валидации и весь набор ошибок на каждом шаге:

```
mean(cv.error) # считаем среднюю ошибку кросс-валидаци
```

```
## [1] 2.187643
```

```
cv.error # вывод полного вектора ошибок (ошибки каждого фолда)
```

```
##  [1] 3.081813 2.921352 2.606293 1.514568 1.880253 2.082523 2.210578 1.683268
##  [9] 1.795076 2.100705
```

```
boxplot(cv.error,xlab='MSE CV',col='cyan',
        border='blue',names='CV error (MSE)',
        main='CV error (MSE) for NN',horizontal=TRUE)
```

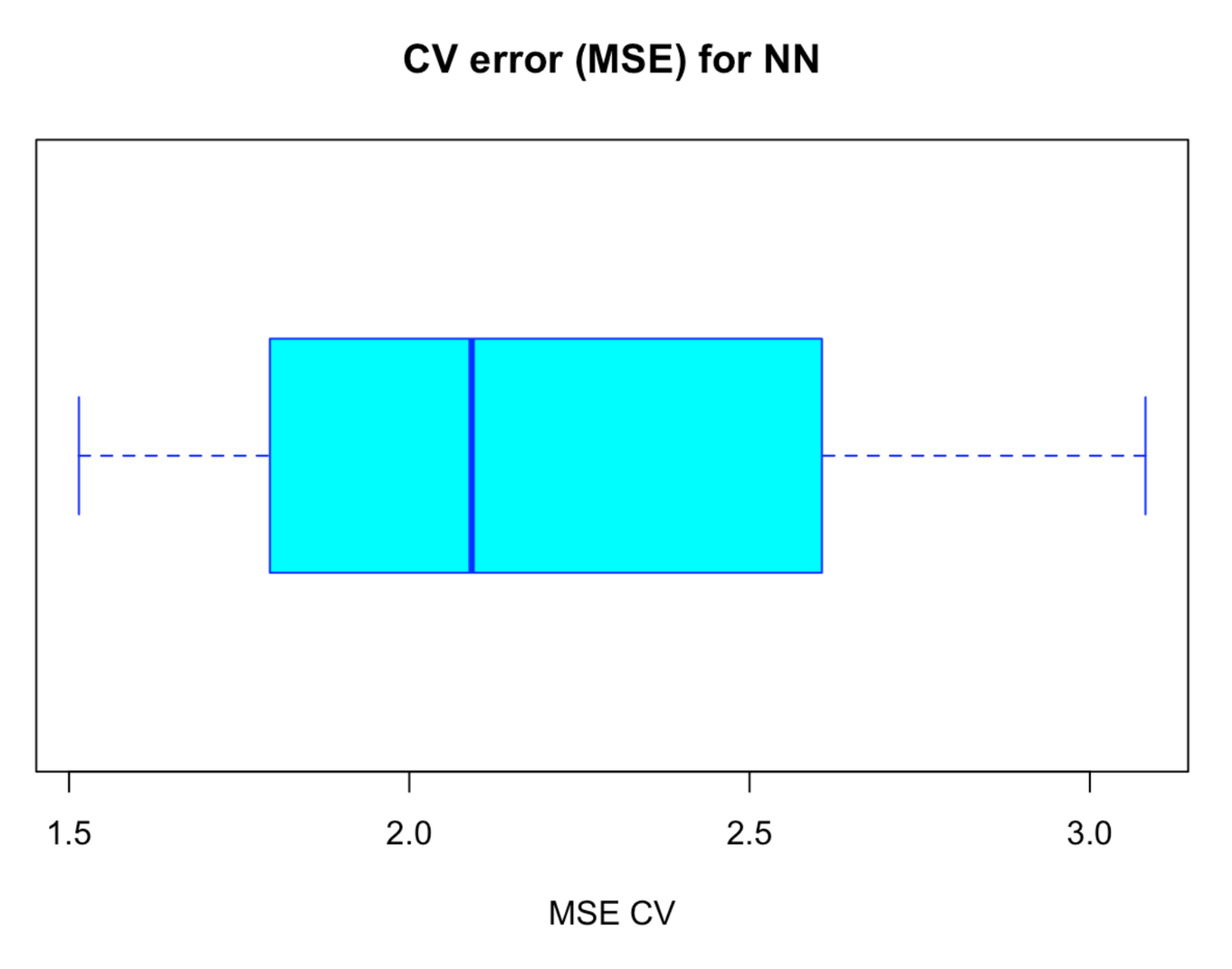


График показывает разброс ошибок MSE для нейронной сети, где медиана находится ближе к 2.1, а диапазон значений распределен от приблизительно 1.6 до 2.8. Левая граница прямоугольника находится в районе 1.8, что соответствует первому квартилю распределения. Правая граница прямоугольника находится в районе 2.4, что соотвествутет третьему квартилю распределения.

Вывод

В ходе данной лабораторной работы был произведён регрессионный анализ данных двумя разными методами, а именно с помощью линейной регрессионной модели и модели искусственной нейронной сети. Первая часть работы была выполнена в соответствии с обучающей статьёй, а во второй части был использован датасет `Car information dataset` с сайта https://www.kaggle.com.

В обоих частях работы нейронная сеть демонстрировала более высокие показатели точности предсказывания целевой переменной, что подтверждается значениями MSE и визуальной интерпретацией графиков в разделах 1.3 и 2.3. Устойчивость моделей была также проверена кросс-валидацией.