Assignment 4.2

Suraj Bhattarai

2023-07-05

##1. Generate a 1000 random data with 10 variables [five continuous: age (18 to 90 years), height (150 - 180 cm), weight (50 - 90 kg), income (10000 - 200000), diastolic blood pressure (70 - 170 mm Hg) and five categorical: sex (male/female), education (no education, primary, secondary, tertiary), place of residence (rural/urban), socio-economic status (low/medium/high) and exercise (yes/no)] using set.seed(your roll number and save it as SR object

```
set.seed(38)

# Generate random continuous variables
age <- sample(18:90, 1000, replace = TRUE)
height <- sample(150:180, 1000, replace = TRUE)
weight <- sample(50:90, 1000, replace = TRUE)
income <- sample(10000:200000, 1000, replace = TRUE)
diastolic_bp <- sample(70:170, 1000, replace = TRUE)

# Generate random categorical variables
sex <- sample(c("male", "female"), 1000, replace = TRUE)
education <- sample(c("no education", "primary", "secondary", "tertiary"), 1000, replace = TRUE)
residence <- sample(c("rural", "urban"), 1000, replace = TRUE)
soc_status <- sample(c("low", "medium", "high"), 1000, replace = TRUE)
exercise <- sample(c("yes", "no"), 1000, replace = TRUE)

# Create the data frame
SR <- data.frame(age, height, weight, income, diastolic_bp, sex, education, residence, soc_status, exer
```

##2. Randomly split the SR object data as SR.train (70%) and SR.test (30%) with replacement sampling and fit multiple linear regression with diastolic blood pressure as dependent variable and rest of variables as independent variable and get fit indices (R-Square, MSE, RMSE and MAE) for the SR.test data

```
set.seed(38)
# do random sampling to divide the cases into two independent sample
ind <- sample(2,nrow(SR), replace = T, prob = c(0.7,0.3))
# data partition
SR.train <- SR[ind==1,]
SR.test <- SR[ind==2,]
# Fit multiple linear regression
lm_model <- lm(diastolic_bp ~ ., data = SR.train)
# Predict diastolic blood pressure for the test data
predicted <- predict(lm_model, newdata = SR.test)

# Calculate fit indices
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 4.2.3
```

```r
R2 <- R2(SR.test$diastolic_bp, predicted)
MSE <- mean((SR.test$diastolic_bp - predicted)^2)
RMSE <- caret::RMSE(SR.test$diastolic_bp, predicted)
MAE <- caret::MAE(SR.test$diastolic_bp, predicted)
```

##3. Fit the multiple linear regression model with Leave One Out Cross-Validation, k-fold cross validation, repeated k-fold cross validation methods and get fit indices for SR.test data and, compare the fit indices of supervised regression models fitted in step 2 and 3 above with careful interpretation

```r
## Fit multiple linear regression with Leave One Out Cross-Validation
library(caret)
train.control <- trainControl(method = "LOOCV")
loocv_model <- train(diastolic_bp  ~ ., data = SR.train, method = "lm",trControl = train.control)

# prediction on test data
l_predictions <- predict(loocv_model, newdata = SR.test)
LR2 <- caret::R2(l_predictions, SR.test$diastolic_bp)

LRMSE<-caret::RMSE(l_predictions , SR.test$diastolic_bp)
LMSE<- mean((l_predictions - SR.test$diastolic_bp)^2)
LMAE<- mean(abs(l_predictions - SR.test$diastolic_bp))

## Kfold cross validation

# Fit multiple linear regression with k-fold Cross-Validation
library(caret)
k <- 10  # Number of folds
train.control <- trainControl(method = "cv", number = k)

k_fold_model <- train(diastolic_bp ~ ., data = SR.train, method = "lm", trControl = train.control)

# Prediction on test data
k_predictions <- predict(k_fold_model, newdata = SR.test)

# Calculate fit indices
library(Metrics)
```

```
## Warning: package 'Metrics' was built under R version 4.2.3
```

```
##
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
KR2 <- R2(k_predictions, SR.test$diastolic_bp)
KRMSE <- rmse(k_predictions, SR.test$diastolic_bp)
KMSE <- mean((k_predictions - SR.test$diastolic_bp)^2)
KMAE <- mean(abs(k_predictions - SR.test$diastolic_bp))


## K fold repeated
# Fit multiple linear regression with repeated k-fold Cross-Validation
library(caret)
k <- 10  # Number of folds
repeats <- 3  # Number of repeats
train.control <- trainControl(method = "repeatedcv", number = k, repeats = repeats)

repeated_kfold_model <- train(diastolic_bp ~ ., data = SR.train, method = "lm", trControl = train.contro

# Prediction on test data
repeated_kfold_predictions <- predict(repeated_kfold_model, newdata = SR.test)

# Calculate fit indices
library(Metrics)
RKR2 <- R2(repeated_kfold_predictions, SR.test$diastolic_bp)
RKRMSE <- rmse(repeated_kfold_predictions, SR.test$diastolic_bp)
RKMSE <- mean((repeated_kfold_predictions - SR.test$diastolic_bp)^2)
RKMAE <- mean(abs(repeated_kfold_predictions - SR.test$diastolic_bp))


#comparing supervised regression fitted in step 2 and step 3
# Create a summary table
summary_table <- data.frame(Method = c("modelTesting", "Loocv testing", "k-fold testing", "repeated K-f

                            R_squared_test = c(R2, LR2, KR2, RKR2),
                            MSE_test = c(MSE, LMSE, KMSE, RKMSE ),
                            MAE_test = c(MAE, LMAE, KMAE, RKMAE),
                            RMSE_test = c(RMSE, LRMSE, KRMSE, RKRMSE)


                            )

summary_table
```

```
##                   Method R_squared_test MSE_test MAE_test RMSE_test
## 1           modelTesting   5.411103e-05 944.4858 27.38275  30.73249
## 2          Loocv testing   5.411103e-05 944.4858 27.38275  30.73249
## 3         k-fold testing   5.411103e-05 944.4858 27.38275  30.73249
## 4 repeated K-fold tesing   5.411103e-05 944.4858 27.38275  30.73249
```

```
#It appears that the fit indices (R-squared, MSE, MAE, RMSE) are the same for all the methods. This mig
```

##4. Fit KNN regression, Decision Tree regression, SVM regression and Neural Network regression using the same dependent and independent variables, get and compare fit indices of these models for SR.test data

```
# Here, we can check which regression model is best
names(SR.train)
```

```
##  [1] "age"          "height"       "weight"       "income"       "diastolic_bp"
##  [6] "sex"          "education"    "residence"    "soc_status"   "exercise"
```

```
# KNN model
library(caret)
knn_model <- train(diastolic_bp ~ ., data = SR.train, method = "knn",  trControl = train.control )
knn_predictions <- predict(knn_model, newdata = SR.test)

# Fit Decision Tree regression
# Check for missing values in the dataset



# Check for missing values in the dataset
# Fit the Decision Tree regression model

library(rpart)


# Fit the Decision Tree regression model
tree_model <- rpart(diastolic_bp ~ ., data = SR.train)
tree_predictions <- predict(tree_model, newdata = SR.test)

# Fit SVM regression
svm_model <- train(diastolic_bp ~ ., data = SR.train, method = "svmRadial", trControl = train.control)
svm_predictions <- predict(svm_model, newdata = SR.test)



# Fit the Neural Network regression model
library(nnet)
nn_model <- nnet(diastolic_bp ~ ., data = SR.train, size = 10)
```

```
## # weights:  141
## initial  value 11048415.555767
## final  value 10918611.000000
## converged
```

```
# Print the summary of the Neural Network model
summary(nn_model)
```

```
## a 12-10-1 network with 141 weights
## options were -
```

```
##   b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1
##   -0.17    0.22   -0.07   -0.33    0.40   -0.40   -0.03   -0.07    0.15    0.67
## i10->h1 i11->h1 i12->h1
##    0.10   -0.65    0.38
##   b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2  i8->h2  i9->h2
##   -0.61   -0.43    0.12    0.64   -0.41   -0.37    0.44    0.13    0.47    0.33
## i10->h2 i11->h2 i12->h2
##   -0.54    0.65    0.20
##   b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3  i7->h3  i8->h3  i9->h3
##   -0.49    0.63   -0.34   -0.16   -0.11    0.44    0.54   -0.25    0.43   -0.09
## i10->h3 i11->h3 i12->h3
##    0.47    0.37   -0.22
##   b->h4  i1->h4  i2->h4  i3->h4  i4->h4  i5->h4  i6->h4  i7->h4  i8->h4  i9->h4
##   -0.16   -0.52   -0.58    0.09    0.07    0.70   -0.10   -0.58    0.10    0.29
## i10->h4 i11->h4 i12->h4
##   -0.20   -0.05    0.11
##   b->h5  i1->h5  i2->h5  i3->h5  i4->h5  i5->h5  i6->h5  i7->h5  i8->h5  i9->h5
##   -0.09   -0.21    0.36   -0.42    0.06    0.43   -0.32    0.17    0.22   -0.22
## i10->h5 i11->h5 i12->h5
##    0.01   -0.41   -0.03
##   b->h6  i1->h6  i2->h6  i3->h6  i4->h6  i5->h6  i6->h6  i7->h6  i8->h6  i9->h6
##    0.55   -0.56    0.39    0.35    0.42    0.01   -0.60   -0.42    0.60    0.26
## i10->h6 i11->h6 i12->h6
##   -0.45    0.50    0.55
##   b->h7  i1->h7  i2->h7  i3->h7  i4->h7  i5->h7  i6->h7  i7->h7  i8->h7  i9->h7
##   -0.02    0.23    0.28   -0.01    0.18    0.09   -0.15    0.55    0.58   -0.29
## i10->h7 i11->h7 i12->h7
##    0.12   -0.12   -0.62
##   b->h8  i1->h8  i2->h8  i3->h8  i4->h8  i5->h8  i6->h8  i7->h8  i8->h8  i9->h8
##   -0.55   -0.43   -0.01    0.58   -0.54    0.22   -0.63    0.70    0.10   -0.64
## i10->h8 i11->h8 i12->h8
##   -0.58   -0.54   -0.52
##   b->h9  i1->h9  i2->h9  i3->h9  i4->h9  i5->h9  i6->h9  i7->h9  i8->h9  i9->h9
##    0.24    0.54    0.57    0.46    0.43    0.34    0.29   -0.58   -0.02   -0.29
## i10->h9 i11->h9 i12->h9
##   -0.38   -0.50    0.34
##   b->h10  i1->h10  i2->h10  i3->h10  i4->h10  i5->h10  i6->h10  i7->h10
##    -0.57     0.29    -0.42     0.62    -0.66     0.55    -0.28     0.57
##  i8->h10  i9->h10 i10->h10 i11->h10 i12->h10
##    -0.34     0.39     0.26     0.41    -0.70
##    b->o   h1->o   h2->o   h3->o   h4->o   h5->o   h6->o   h7->o   h8->o   h9->o
## 1281.32 1280.91   -0.51    0.48 1280.79 1280.19 1280.16 1280.70   -0.60 1280.82
##  h10->o
##    0.15
```

```
nn_predictions <- predict(nn_model, newdata = SR.test)



# Calculate fit indices
library(Metrics)
KNN_R2 <- R2(knn_predictions, SR.test$diastolic_bp)
KNN_RMSE <- rmse(knn_predictions, SR.test$diastolic_bp)
KNN_MSE <- mean((knn_predictions - SR.test$diastolic_bp)^2)
```

```r
KNN_MAE <- mean(abs(knn_predictions - SR.test$diastolic_bp))

TREE_R2 <- caret::R2(tree_predictions , SR.test$diastolic_bp)
TREE_RMSE <- rmse(tree_predictions, SR.test$diastolic_bp)
TREE_MSE <- mean((tree_predictions - SR.test$diastolic_bp)^2)
TREE_MAE <- mean(abs(tree_predictions - SR.test$diastolic_bp))

SVM_R2 <- R2(svm_predictions, SR.test$diastolic_bp)
SVM_RMSE <- rmse(svm_predictions, SR.test$diastolic_bp)
SVM_MSE <- mean((svm_predictions - SR.test$diastolic_bp)^2)
SVM_MAE <- mean(abs(svm_predictions - SR.test$diastolic_bp))

NN_R2 <- R2(nn_predictions, SR.test$diastolic_bp )
NN_RMSE <- rmse(nn_predictions, SR.test$diastolic_bp)
NN_MSE <- mean((nn_predictions - SR.test$diastolic_bp)^2)
NN_MAE <- mean(abs(nn_predictions - SR.test$diastolic_bp))
models_summary<-data.frame(
  heading =c("mlr","KNN" , "Decion Tree", "SVM" , "NN") ,
  test_R2 = c(R2,KNN_R2 ,TREE_R2 , SVM_R2 , NN_R2  ),
  test_mse = c(MSE,KNN_MSE , TREE_MSE, SVM_MSE , NN_MSE),
  test_rmse = c( RMSE , KNN_RMSE , TREE_RMSE ,SVM_RMSE ,NN_RMSE ),
  test_mae = c( MAE,KNN_MAE , TREE_MAE , SVM_MAE , NN_MAE)
)


models_summary
```

```
##       heading      test_R2   test_mse test_rmse  test_mae
## 1         mlr 5.411103e-05   944.4858  30.73249  27.38275
## 2         KNN 4.291854e-04  1012.8538  31.82536  27.65423
## 3 Decion Tree 1.090467e-02   909.0599  30.15062  26.86159
## 4         SVM 1.283066e-03   978.5662  31.28204  28.01999
## 5          NN           NA 15794.3881 125.67573 121.97552
```

5. Which supervised regression model is the best model for doing prediction in the SR.test data? Why?

ans: Based on these fit indices, Multiple linear regression model performs the best among the models listed. It has the highest R-Squared value , indicating better goodness of fit compared to the other models. Additionally, it has the lowest MSE, RMSE, and MAE values, indicating better accuracy in predicting the diastolic blood pressure.

##6. Predict diastolic blood pressure of a person with 50 years, 175mm height, 80 kg weight, 90000 income, male, tertiary level education, living in urban area, medium socio-economic status and no exercise and interpret the result carefully

```r
## Create a data frame with the predictor variables for the new individual
new_data <- data.frame(age = 50, height = 175, weight = 80, income = 90000, sex = "male", education = "
# Check the variable names in the new data
names(new_data)
```

```
## [1] "age"        "height"     "weight"     "income"     "sex"
## [6] "education"  "residence"  "soc_status" "exercise"
```

```r
# Predict the diastolic blood pressure for the new individual
prediction <- predict(lm_model, newdata = new_data)

# Display the prediction
prediction
```

```
##        1
## 119.1509
```

##7. Reflection on the assignment # In this assignment, we performed various supervised regression techniques on a synthetic dataset. We started by generating random data with 10 variables, including both continuous and categorical predictors. # We split the data into training and test sets and fitted a multiple linear regression model on the training set. We then evaluated the model's performance on the test set using fit indices such as R-squared, MSE, RMSE, and MAE. # Next, we explored different cross-validation techniques, including Leave One Out Cross-Validation, k-fold Cross-Validation, and repeated k-fold Cross-Validation. We compared the fit indices of the supervised regression models fitted using these techniques with the fit indices of the initial multiple linear regression model. # Additionally, we applied other regression algorithms such as KNN regression, Decision Tree regression, SVM regression, and Neural Network regression to the dataset and compared their fit indices on the test set. # Finally, based on the comparison of fit indices, we determined the best supervised regression