Assignment

Machine learning

2024-11-24

```
# To load the libraries
library(caret) # data pratitioning and model
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

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

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

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

```
## Warning: package 'lattice' was built under R version 4.3.3
```

```
# training
#libray(ggplot2) # for visualization
library(e1071)   # for Naive Bayes and SVM
```

```
## Warning: package 'e1071' was built under R version 4.3.3
```

```
library(randomForest)  # for Random Forest
```

```
## Warning: package 'randomForest' was built under R version 4.3.3
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(MASS)          # Bostan dataset
```

Load the Bostan Housing Dataset

```
Bost=read.csv("D:\\sandip sir 3rd sem lab\\Boston.csv")
head(Bost)
```

```
##   X    crim zn indus chas   nox    rm  age    dis rad tax ptratio
black lstat
## 1 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 3
96.90  4.98
## 2 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 3
96.90  9.14
## 3 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 3
92.83  4.03
## 4 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 3
94.63  2.94
## 5 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 3
96.90  5.33
## 6 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 3
94.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

Remove unusal column

```
Bost$X<-NULL
head(Bost)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  bl
ack lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 39
6.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 39
6.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 39
2.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 39
4.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 39
6.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 39
4.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

Cheacking the null value

```
sum(is.na(Bost))
```

```
## [1] 0
```

```
colSums(is.na(Bost))
```

```
##    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
```

Spliting the data for train and test

```
# Create a binary target variable for logistic regression (e.g., MEDV
> 25)
Bost$medv_binary <- ifelse(Bost$medv > 25, 1, 0)

# Set seed for reproducibility
set.seed(123)

# Split the dataset into training (80%) and testing (20%) sets
trainIndex <- createDataPartition(Bost$medv, p = .8,
                                  list = FALSE,
                                  times = 1)
Bost_train <- Bost[trainIndex, ]
Bost_test <- Bost[-trainIndex, ]
```

Linear Regression Model

```
# Fit a linear regression model
linear_model <- lm(medv ~ ., data = Bost_train)

# Predictions on the test set
linear_predictions <- predict(linear_model, newdata = Bost_test)

# Calculate MSE and R² for linear regression
linear_mse <- mean((Bost_test$medv - linear_predictions)^2)
linear_r2 <- summary(linear_model)$r.squared

cat("Linear Regression MSE:", linear_mse, "\n")
```

```
## Linear Regression MSE: 16.3339
```

```
cat("Linear Regression R²:", linear_r2, "\n")
```

```
## Linear Regression R²: 0.8224706
```

Polynomial Regression Model

```
# Fit a polynomial regression model (degree 2 for example)
poly_model <- lm(medv ~ poly(rm, 2), data = Bost_train)

# Predictions on the test set
poly_predictions <- predict(poly_model, newdata = Bost_test)

# Calculate MSE and R² for polynomial regression
poly_mse <- mean((Bost_test$medv - poly_predictions)^2)
poly_r2 <- summary(poly_model)$r.squared

cat("Polynomial Regression MSE:", poly_mse, "\n")
```

```
## Polynomial Regression MSE: 36.66255
```

```
cat("Polynomial Regression R²:", poly_r2, "\n")
```

```
## Polynomial Regression R²: 0.5394765
```

Logistic Regression Model

```
# Fit a logistic regression model (using binary target)
logistic_model <- glm(medv_binary ~ ., family = binomial, data = Bost_
train)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# Predictions on the test set (probabilities)
logistic_probabilities <- predict(logistic_model, newdata = Bost_test,
type = "response")

# Convert probabilities to binary predictions using a threshold of 0.5
logistic_predictions <- ifelse(logistic_probabilities > 0.5, 1, 0)

# Calculate accuracy for logistic regression
logistic_accuracy <- mean(logistic_predictions == Bost_test$medv_binar
y)
cat("Logistic Regression Accuracy:", logistic_accuracy * 100, "%\n")
```

```
## Logistic Regression Accuracy: 98.9899 %
```

Naive Bayes Model

```
library(e1071)
# Fit a Naive Bayes model using the binary target variable
naive_bayes_model <- naiveBayes(medv_binary ~ ., data = Bost_train)

# Predictions on the test set
naive_bayes_predictions <- predict(naive_bayes_model, newdata = Bost_t
est)

# Calculate accuracy for Naive Bayes
naive_bayes_accuracy <- mean(naive_bayes_predictions == Bost_test$medv
_binary)
cat("Naive Bayes Accuracy:", naive_bayes_accuracy * 100, "%\n")
```

```
## Naive Bayes Accuracy: 78.78788 %
```

K-Nearest Neighbors (KNN) Model

```
# Fit KNN model (using caret's train function)
knn_model <- train(medv_binary ~ ., data = Bost_train,
                   method = "knn",
                   tuneLength = 10)
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to
do
## regression and your outcome only has two possible values Are you tr
ying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```
# Predictions on the test set
knn_predictions <- predict(knn_model, newdata = Bost_test)

# Calculate accuracy for KNN
knn_accuracy <- mean(knn_predictions == Bost_test$medv_binary)
cat("KNN Accuracy:", knn_accuracy * 100, "%\n")
```

```
## KNN Accuracy: 54.54545 %
```

Random Forest Model

```
library(randomForest)
# Fit Random Forest model
rf_model <- randomForest(medv ~ ., data = Bost_train)

# Predictions on the test set
rf_predictions <- predict(rf_model, newdata = Bost_test)

# Calculate MSE and R² for Random Forest
rf_mse <- mean((Bost_test$medv - rf_predictions)^2)
rf_r2 <- cor(Bost_test$medv, rf_predictions)^2

cat("Random Forest MSE:", rf_mse, "\n")
```

```
## Random Forest MSE: 5.412079
```

```
cat("Random Forest R²:", rf_r2, "\n")
```

```
## Random Forest R²: 0.9476195
```

Support Vector Machine (SVM) Model

```
library(e1071)
# Fit SVM model (for regression)
svm_model <- svm(medv ~ ., data = Bost_train)

# Predictions on the test set
svm_predictions <- predict(svm_model, newdata = Bost_test)

# Calculate MSE and R² for SVM
svm_mse <- mean((Bost_test$medv - svm_predictions)^2)
svm_r2 <- cor(Bost_test$medv, svm_predictions)^2

cat("SVM MSE:", svm_mse, "\n")
```

```
## SVM MSE: 9.606465
```

```
cat("SVM R²:", svm_r2, "\n")
```

```
## SVM R²: 0.9101539
```

Decision Tree Model

```
library(rpart)       # for decision Tree model
```

```
## Warning: package 'rpart' was built under R version 4.3.3
```

```
library(rpart.plot)   # For visualize
```

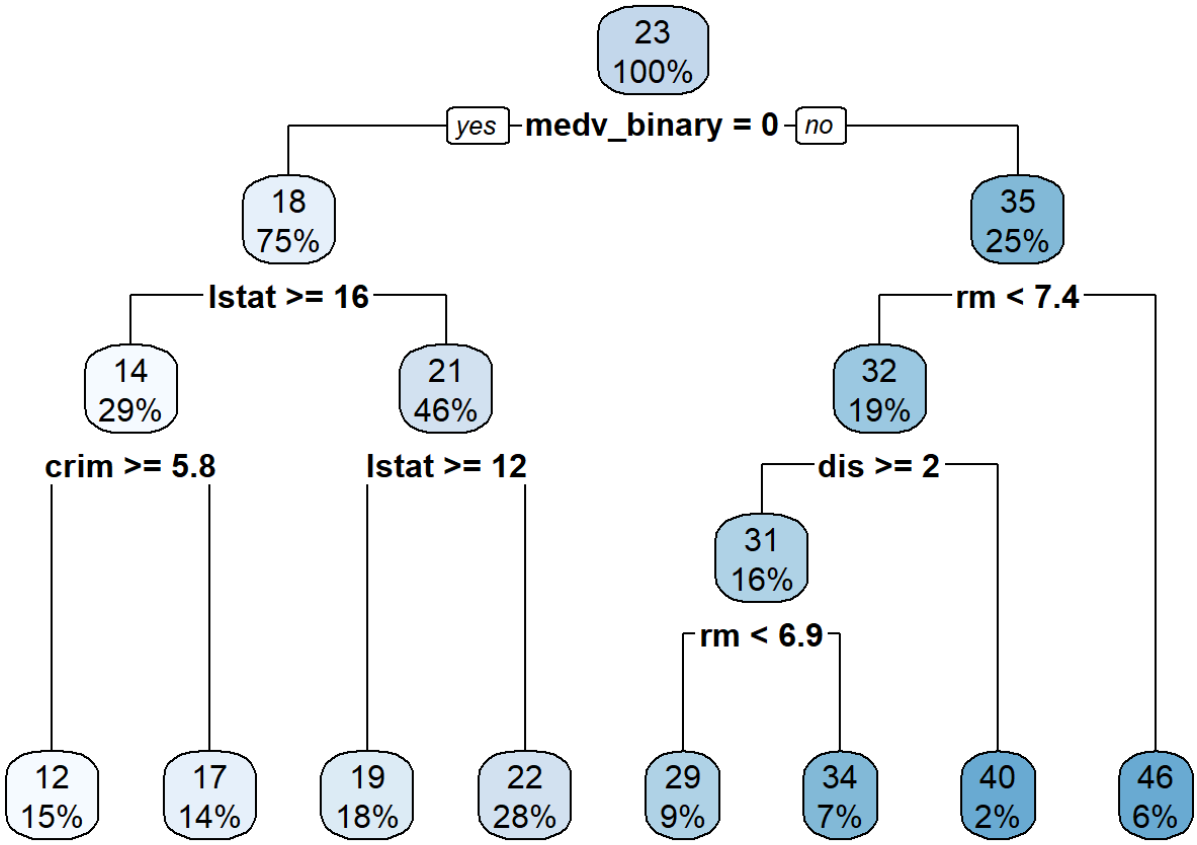
```
## Warning: package 'rpart.plot' was built under R version 4.3.3
```

```
# Fit a decision tree model
tree_model <- rpart(medv ~ ., data = Bost_train)

# Print the model summary
print(tree_model)
```

```
## n= 407
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 407 34125.5800 22.51057
##    2) medv_binary< 0.5 307  6707.0460 18.37296
##      4) lstat>=16.085 119  1815.9550 14.19328
##        8) crim>=5.76921 62   625.7987 11.92581 *
##        9) crim< 5.76921 57   524.6572 16.65965 *
##      5) lstat< 16.085 188  1496.2850 21.01862
##       10) lstat>=11.67 74   560.0541 19.26216 *
##       11) lstat< 11.67 114   559.7362 22.15877 *
##    3) medv_binary>=0.5 100  6027.5930 35.21300
##      6) rm< 7.443 77  2262.2880 32.07403
##       12) dis>=1.9704 67   707.7272 30.94776
##         24) rm< 6.945 38   196.6382 28.97105 *
##         25) rm>=6.945 29   168.0483 33.53793 *
##       13) dis< 1.9704 10   900.1560 39.62000 *
##      7) rm>=7.443 23   466.6391 45.72174 *
```

```
# Visualize the decision tree
rpart.plot(tree_model)
```




```
# Make predictions on the test set
tree_predictions <- predict(tree_model, newdata = Bost_test)
# Calculate MSE for the decision tree model
tree_mse <- mean((Bost_test$medv - tree_predictions)^2)

# Calculate R² for the decision tree model
tree_r2 <- cor(Bost_test$medv, tree_predictions)^2

cat("Decision Tree MSE:", tree_mse, "\n")
```

```
## Decision Tree MSE: 12.63413
```

```
cat("Decision Tree R²:", tree_r2, "\n")
```

```
## Decision Tree R²: 0.8559424
```

Comparision between above Model

Linear Regression MSE: 16.3339 Linear Regression R^2: 0.8224706 Polynomial Regression MSE: 36.66255 Polynomial Regression R^2: 0.5394765 Logistic Regression Accuracy: 98.9899 % Naive Bayes Accuracy: 78.78788 % KNN Accuracy: 54.54545 % Random Forest MSE: 5.412079 Random Forest R^2: 0.9476195 SVM MSE: 9.606465 SVM R^2: 0.9101539 Decision Tree MSE: 12.63413 Decision Tree R^2: 0.8559424

Higher R2 indicate high explain the variability of data,here Random Forest is better for all then SVM and also logistics regression accuracy is 98.9899 % it is good in performence.