Peer-graded_Assignment

2025-10-07

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

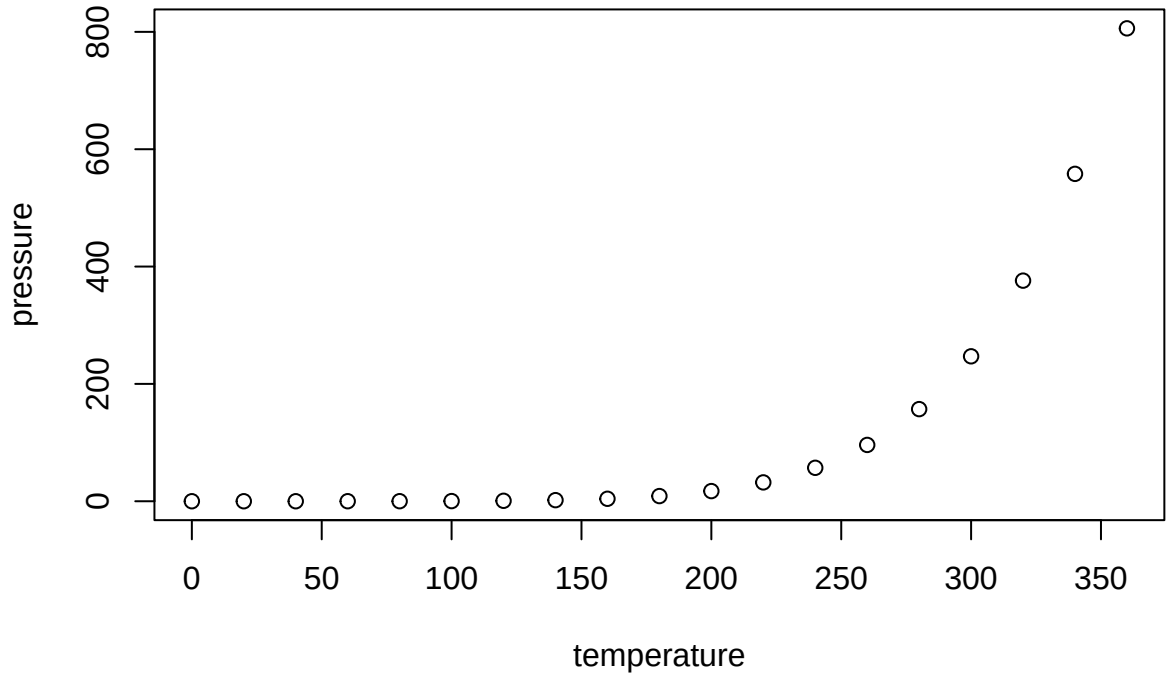
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```
# --- 1. Setup: Load Libraries and Set Seed ---

# Load necessary libraries
library(caret)
```

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

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

```
library(randomForest)
```

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

# Set seed for reproducibility
set.seed(12345)
```

```r
# --- 2. Data Loading and Cleaning ---

# Load datasets
training_raw <- read.csv("pml-training.csv", na.strings = c("NA", ""))
testing_raw <- read.csv("pml-testing.csv", na.strings = c("NA", ""))

# Remove columns with a high percentage of missing values
good_cols <- colSums(is.na(training_raw)) < (nrow(training_raw) * 0.80)
training_cleaned <- training_raw[, good_cols]
testing_cleaned <- testing_raw[, good_cols]

# Remove metadata columns
training_final <- training_cleaned[, -c(1:7)]
testing_final <- testing_cleaned[, -c(1:7)]

# **NEW CORRECTION:** Convert the outcome variable 'classe' to a factor.
# This is the key step to ensure levels are consistent everywhere.
training_final$classe <- as.factor(training_final$classe)


# --- 3. Data Partitioning for Model Validation ---

# Split the cleaned training data into a training subset (75%) and a validation subset (25%)
inTrain <- createDataPartition(y = training_final$classe, p = 0.75, list = FALSE)
training_subset <- training_final[inTrain, ]
validation_subset <- training_final[-inTrain, ]


# --- 4. Model Training and Cross-Validation ---

# Model 1: Decision Tree (rpart)
trControl_dt <- trainControl(method = "cv", number = 5)
model_dt <- train(classe ~ .,
                  data = training_subset,
                  method = "rpart",
                  trControl = trControl_dt)

# Model 2: Random Forest (rf)
# This step can take a few minutes to run.
trControl_rf <- trainControl(method = "cv", number = 5)
model_rf <- train(classe ~ .,
                  data = training_subset,
                  method = "rf",
                  trControl = trControl_rf)


# --- 5. Model Evaluation and Selection ---

# Predictions with the Decision Tree model
pred_dt <- predict(model_dt, newdata = validation_subset)
cm_dt <- confusionMatrix(data = pred_dt, reference = validation_subset$classe)
print("Decision Tree Performance on Validation Set:")
```

```
## [1] "Decision Tree Performance on Validation Set:"
```

```
print(cm_dt)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1252  396  434  343  114
##          B   30  317   24  151  132
##          C   90  236  397  310  229
##          D    0    0    0    0    0
##          E   23    0    0    0  426
##
## Overall Statistics
##
##                Accuracy : 0.4878
##                  95% CI : (0.4737, 0.5019)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3306
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8975  0.33404  0.46433   0.0000  0.47281
## Specificity            0.6332  0.91479  0.78637   1.0000  0.99425
## Pos Pred Value         0.4931  0.48471  0.31458      NaN  0.94878
## Neg Pred Value         0.9395  0.85129  0.87424   0.8361  0.89338
## Prevalence             0.2845  0.19352  0.17435   0.1639  0.18373
## Detection Rate         0.2553  0.06464  0.08095   0.0000  0.08687
## Detection Prevalence   0.5177  0.13336  0.25734   0.0000  0.09156
## Balanced Accuracy      0.7654  0.62441  0.62535   0.5000  0.73353
```

```
# Predictions with the Random Forest model
pred_rf <- predict(model_rf, newdata = validation_subset)
cm_rf <- confusionMatrix(data = pred_rf, reference = validation_subset$classe)
print("Random Forest Performance on Validation Set:")
```

```
## [1] "Random Forest Performance on Validation Set:"
```

```
print(cm_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  947    6    0    0
##          C    0    1  848   15    0
##          D    0    0    1  784    1
##          E    0    0    0    5  900
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9913, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9979   0.9918   0.9751   0.9989
## Specificity            0.9997   0.9985   0.9960   0.9995   0.9988
## Pos Pred Value         0.9993   0.9937   0.9815   0.9975   0.9945
## Neg Pred Value         1.0000   0.9995   0.9983   0.9951   0.9997
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1931   0.1729   0.1599   0.1835
## Detection Prevalence   0.2847   0.1943   0.1762   0.1603   0.1845
## Balanced Accuracy      0.9999   0.9982   0.9939   0.9873   0.9988
```

```
# Model Selection: The Random Forest model is chosen due to higher accuracy.


# --- 6. Final Prediction on the Official Test Set ---

# Use the chosen model (Random Forest) to predict the 20 outcomes
final_predictions <- predict(model_rf, newdata = testing_final)

# Display the final 20 predictions
print("Final Predictions for the 20 Test Cases:")
```

```
## [1] "Final Predictions for the 20 Test Cases:"
```

```
print(final_predictions)
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```