Practical Machine Learning - Course Project

Elvan Ceyhan

2025-01-02

Introduction

The goal of this project is to predict the manner in which participants performed barbell exercises using accelerometer data collected from the belt, forearm, arm, and dumbbell. The target variable, `classe`, indicates the type of exercise. This report describes the steps taken to clean the data, train a model, evaluate its performance, and predict outcomes for a test dataset.

Preprocessing

Load Required Libraries

```r
library(caret)
library(randomForest)
library(dplyr)
library(ggplot2)
```

Load and Clean Data

The training and testing datasets were loaded and preprocessed to remove columns with missing values and irrelevant features, such as timestamps.

```r
train_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

train_data <- read.csv(train_url, na.strings = c("NA", "#DIV/0!", ""))
test_data <- read.csv(test_url, na.strings = c("NA", "#DIV/0!", ""))

# Remove columns with mostly missing values
train_data <- train_data[, colSums(is.na(train_data)) == 0]
test_data <- test_data[, colSums(is.na(test_data)) == 0]

# Remove irrelevant columns
irr_cols <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
              "cvtd_timestamp", "new_window", "num_window")
train_data <- train_data %>% select(-one_of(irr_cols))
test_data <- test_data %>% select(-one_of(irr_cols))
```

Data Partitioning

The training dataset was split into 70% for training and 30% for validation.

```r
set.seed(123)
in_train <- createDataPartition(train_data$classe, p = 0.7, list = FALSE)
train_set <- train_data[in_train, ]
```

```r
valid_set <- train_data[-in_train, ]

# Ensure 'classe' is a factor
train_set$classe <- as.factor(train_set$classe)
valid_set$classe <- as.factor(valid_set$classe)
```

Model Training

We used a Random Forest classifier due to its robustness and high performance for classification tasks.

```r
set.seed(123)
rf_mod <- randomForest(classe ~ ., data = train_set, importance = TRUE, ntree = 100)
```

Model Evaluation

We evaluate the model on the validation set, achieving an accuracy of 99.54%.

```r
rf_preds <- predict(rf_mod, valid_set)
conf_mat <- confusionMatrix(rf_preds, valid_set$classe)
print(conf_mat)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 1674    5    0    0    0
##          B    0 1130    2    0    0
##          C    0    4 1024    8    4
##          D    0    0    0  956    4
##          E    0    0    0    0 1074
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9933, 0.997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9942
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9921   0.9981   0.9917   0.9926
## Specificity            0.9988   0.9996   0.9967   0.9992   1.0000
## Pos Pred Value         0.9970   0.9982   0.9846   0.9958   1.0000
## Neg Pred Value         1.0000   0.9981   0.9996   0.9984   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1920   0.1740   0.1624   0.1825
## Detection Prevalence   0.2853   0.1924   0.1767   0.1631   0.1825
## Balanced Accuracy      0.9994   0.9958   0.9974   0.9954   0.9963
```

```r
# Expected out-of-sample error
oos_err <- 1 - conf_mat$overall['Accuracy']
print(paste("Expected Out-of-Sample Error:", round(oos_err, 4)))
```

```
## [1] "Expected Out-of-Sample Error: 0.0046"
```

Confusion Matrix

	A	B	C	D	E
A	1674	5	0	0	0
B	0	1130	2	0	0
C	0	4	1024	8	4
D	0	0	0	956	4
E	0	0	0	0	1074

The model's confusion matrix indicates strong performance across all classes.

Predictions on Test Data

The trained model was applied to the test dataset to predict the 20 test cases.

```r
test_preds <- predict(rf_mod, test_data)

print(test_preds)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

Discussion and Conclusion

Main Decisions:

- **Model Choice**: Random Forest was selected due to its ability to handle high-dimensional data and provide feature importance metrics.
- **Cross-Validation**: A 70%-30% train-validation split ensured robust performance evaluation.
- **Preprocessing**: Removing irrelevant and missing value-heavy columns reduced noise in the data.

Expected Out-of-Sample Error: The expected out-of-sample error is approximately 0.46%, demonstrating excellent model performance.

Future Improvements: Further enhancements could include:

- Hyperparameter tuning for the Random Forest model.
- Exploration of ensemble methods for even better accuracy.

References

1. Data Source: Weight Lifting Exercise Dataset
2. Random Forest: Breiman, L. (2001). Random Forests. Machine Learning.