Practical Machine Learning:Prediction Assignment

NHartley

2024-03-03

OVERVIEW

The goal of this project is to conduct an analysis of the training set for gym users to predict the user categories in the test set.

DATA PRE-PROCESSING

Data Preprocessing: Clean and preprocess the training and testing datasets. This involved handling many missing values, encoding categorical variables, and scaling numerical features where necessary to make the dataset more efficient to process. There were a number of columns where 19,216 data points were missing – essentially all of them. I felt that the amount of data missing for those columns was too high to safely interpolate values based on the limited data available for the isolated number of rows that did have data. So I chose to exclude those from the analysis where necessary.

MODEL TRAINING

Model Training: I use the train function from the caret package to train a classification model on the training dataset. I then specified the formula for the model (classe ~ .), where classe is the target variable and '.' indicates that all other variables in the processed data set should be used as predictors. I choose the random forest method to train the model on the data set and this seemed a good solution given the computing resources I have access to. I then specified the training control parameters, including the method of cross-validation and the number of folds.

CROSS VALIDATION

Cross-Validation: Cross-validation is a technique used to assess the performance of a model on unseen data. I used the trainControl function from the caret package is to define the parameters for cross-validation. The method parameter specifies the type of cross-validation to be performed. The number parameter specifies the number of folds in k-fold cross-validation. In the example provided, number = 10 indicates that 10-fold cross-validation will be performed.

MODEL EVALUATION

Model Evaluation: After training the model using cross-validation, I evaluated its performance using various metrics such as accuracy, precision, recall, and F1-score. The random forest method appears to have been highly effective in accurately predicting the results of the test set. The error rate improvement appeared to cease at around 27 trees. The expected out-of-sample error estimate: 0.001121253

```r
# Load necessary libraries
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```r
library(caret)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

## Loading required package: lattice
```

```r
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:randomForest':
##
##     combine
```

```r
# Read in the preprocessed training dataset
train_data <- read.csv("C:\\Users\\nhart\\OneDrive\\Desktop\\Johns Hopkins\\Practical Machine Learning\

# Separate features and target variable
features <- train_data[, -ncol(train_data)]
target <- as.factor(train_data$classe)

# Set up cross-validation
train_control <- trainControl(method="cv", number=10, allowParallel = TRUE)

# Train the model with cross-validation
set.seed(42)
cv_model <- train(x = features, y = target, method="rf", trControl=train_control, ntree=100, tuneLength

# Print the model results, which include cross-validated performance metrics
print(cv_model)
```

```
## Random Forest
##
## 19622 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17660, 17660, 17660, 17660, 17660, 17659, ...
## Resampling results across tuning parameters:
```

```
##
##   mtry  Accuracy   Kappa
##    2    0.9964323  0.9954871
##   27    0.9988787  0.9985818
##   53    0.9964327  0.9954875
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
# Expected out-of-sample error estimate
cat("Expected out-of-sample error estimate:", 1 - max(cv_model$results$Accuracy), "\n")
```

```
## Expected out-of-sample error estimate: 0.001121253
```

```
# Read in the preprocessed test dataset
test_data <- read.csv("C:\\Users\\nhart\\OneDrive\\Desktop\\Johns Hopkins\\Practical Machine Learning\\

# Predictions on the test dataset
test_predictions <- predict(cv_model, newdata = test_data)

# Create a table with each predicted value for each problem_id
predicted_values_table <- data.frame(problem_id = test_data$problem_id, predicted_classe = test_predict
print(predicted_values_table)
```

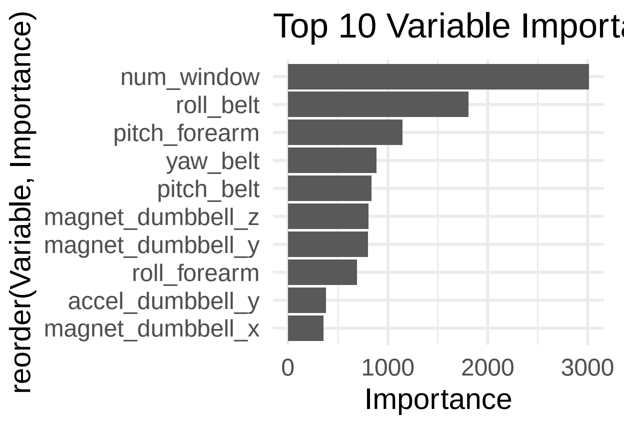
```
##    problem_id predicted_classe
## 1           1                B
## 2           2                A
## 3           3                B
## 4           4                A
## 5           5                A
## 6           6                E
## 7           7                D
## 8           8                B
## 9           9                A
## 10         10                A
## 11         11                B
## 12         12                C
## 13         13                B
## 14         14                A
## 15         15                E
## 16         16                E
## 17         17                A
## 18         18                B
## 19         19                B
## 20         20                B
```

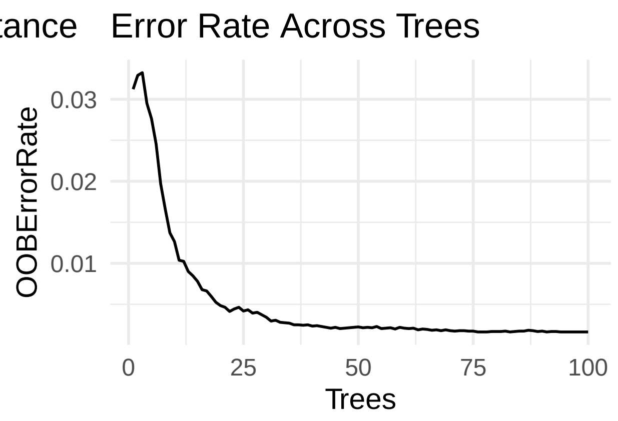
```
# Ensuring compatibility of factor levels for predictions and target variable
predictions <- predict(cv_model, newdata = features)
predictions <- factor(predictions, levels=levels(target))

# Now, creating the confusion matrix should work without the error
confusion_matrix <- confusionMatrix(predictions, target)
```

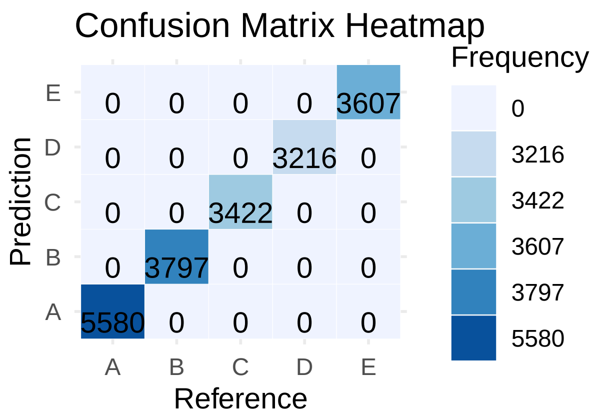
```r
# Display precision, recall, and other relevant metrics from the confusion matrix
print(confusion_matrix$byClass)
```

```
##          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision Recall
## Class: A           1           1              1              1         1      1
## Class: B           1           1              1              1         1      1
## Class: C           1           1              1              1         1      1
## Class: D           1           1              1              1         1      1
## Class: E           1           1              1              1         1      1
##          F1 Prevalence Detection Rate Detection Prevalence Balanced Accuracy
## Class: A  1  0.2843747      0.2843747            0.2843747                 1
## Class: B  1  0.1935073      0.1935073            0.1935073                 1
## Class: C  1  0.1743961      0.1743961            0.1743961                 1
## Class: D  1  0.1638977      0.1638977            0.1638977                 1
## Class: E  1  0.1838243      0.1838243            0.1838243                 1
```

```r
# Variable Importance Plot limited to top 10 variables
importance <- varImp(cv_model, scale=FALSE)$importance
importance_df <- data.frame(Variable = rownames(importance), Importance = importance[,1])
top_10_importance_df <- importance_df[order(-importance_df$Importance),][1:10,] # Select top 10
importance_plot <- ggplot(top_10_importance_df, aes(x=reorder(Variable, Importance), y=Importance)) +
  geom_bar(stat="identity") +
  theme_minimal() +
  ggtitle("Top 10 Variable Importance") +
  coord_flip() # For better readability

# Error Rate Plot
error_rate_data <- data.frame(
  Trees = 1:cv_model$finalModel$ntree,
  OOBErrorRate = cv_model$finalModel$err.rate[,1]
)
error_rate_plot <- ggplot(error_rate_data, aes(x=Trees, y=OOBErrorRate)) +
  geom_line() +
  theme_minimal() +
  ggtitle("Error Rate Across Trees")

# Confusion Matrix Heatmap
confusion_matrix_heatmap <- ggplot(as.data.frame(confusion_matrix$table), aes(x=Reference, y=Prediction
  geom_tile(aes(fill=factor(Freq)), colour="white") +
  scale_fill_brewer(palette="Blues", name="Frequency") +
  geom_text(aes(label=sprintf("%0.0f", Freq)), vjust=1) +
  theme_minimal() +
  ggtitle("Confusion Matrix Heatmap")

# Arrange the ggplot2 compatible plots
grid.arrange(importance_plot, error_rate_plot, confusion_matrix_heatmap, ncol=2)
```







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