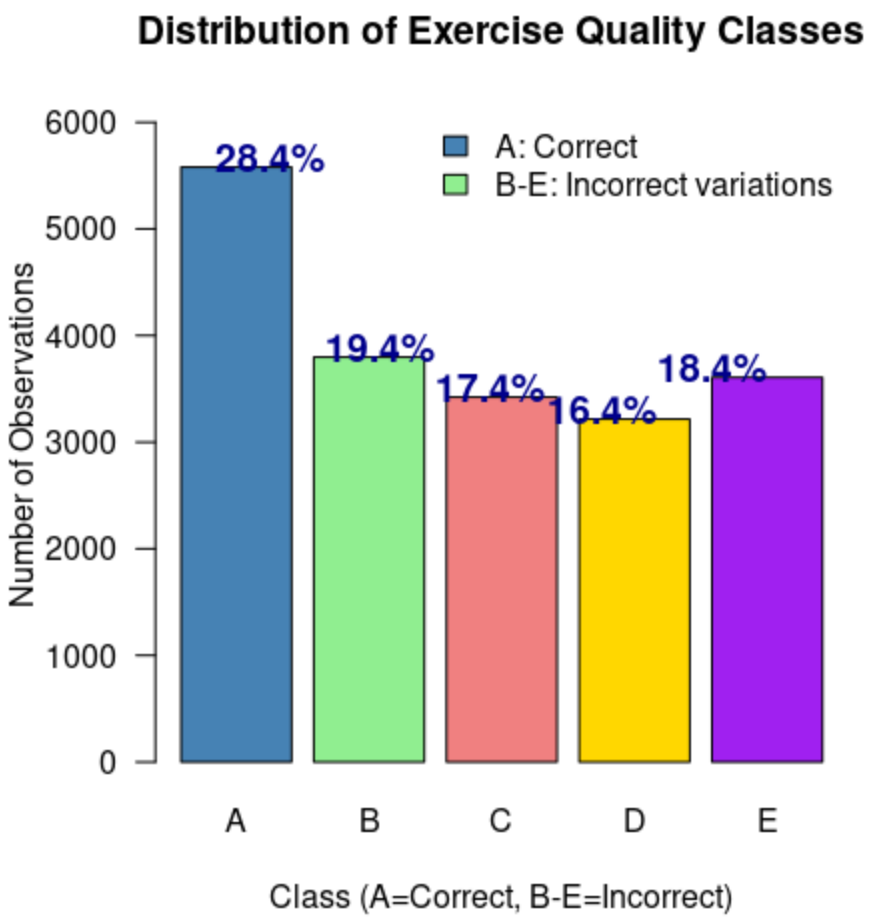


```
# ================================================
> # PRACTICAL MACHINE LEARNING - COURSE PROJECT
> # ================================================
>
> # Load required packages
> library(caret)
> library(randomForest)
> library(rpart)
> library(rpart.plot)
> library(ggplot2)
> library(dplyr)
>
> # Set seed for reproducibility
> set.seed(1234)
>
> # 1. DATA LOADING
> # ===============
>
```

```
> cat("Step 1: Loading data...\n")
Step 1: Loading data...
>
> # Load datasets (already downloaded)
> training <- read.csv("pml-training.csv", na.strings = c("NA", "#DIV/0!", ""))
> testing <- read.csv("pml-testing.csv", na.strings = c("NA", "#DIV/0!", ""))
>
> cat("Training data:", dim(training), "\n")
Training data: 19622 160
> cat("Testing data:", dim(testing), "\n")
Testing data: 20 160
>
> # 2. DATA CLEANING
> # ================
>
> cat("\nStep 2: Cleaning data...\n")

Step 2: Cleaning data...
>
> # Remove columns with >95% missing values
> na_percentage <- colMeans(is.na(training))
> columns_to_keep <- na_percentage < 0.95
> training_clean <- training[, columns_to_keep]
> testing_clean <- testing[, columns_to_keep]
>
> # Remove non-predictive columns
> non_predictors <- c("X", "user_name", "raw_timestamp_part_1",
+                     "raw_timestamp_part_2", "cvtd_timestamp",
+                     "new_window", "num_window")
>
> # Check which columns actually exist
> existing_non_preds <- non_predictors[non_predictors %in% names(training_clean)]
> training_clean <- training_clean[, !(names(training_clean) %in% existing_non_pr
eds)]
> testing_clean <- testing_clean[, !(names(testing_clean) %in% existing_non_preds
)]
>
> cat("After cleaning:\n")
After cleaning:
> cat("Training:", dim(training_clean), "\n")
Training: 19622 53
> cat("Testing:", dim(testing_clean), "\n")
Testing: 20 53
>
> # 3. DATA PARTITIONING
> # ====================
>
> cat("\nStep 3: Partitioning data...\n")

Step 3: Partitioning data...
>
> # Use createDataPartition from caret
```

```
> train_index <- createDataPartition(training_clean$classe, p = 0.7, list = FALSE
)
> train_data <- training_clean[train_index, ]
> validation_data <- training_clean[-train_index, ]
>
> cat("Training set:", nrow(train_data), "observations\n")
Training set: 13737 observations
> cat("Validation set:", nrow(validation_data), "observations\n")
Validation set: 5885 observations
> cat("Test set:", nrow(testing_clean), "observations\n")
Test set: 20 observations
>
> # 4. MODEL TRAINING (RANDOM FOREST)
> # ==================================
>
> cat("\nStep 4: Training Random Forest model...\n")

Step 4: Training Random Forest model...
>
> # Ensure classe is a factor (for classification)
> train_data$classe <- as.factor(train_data$classe)
> validation_data$classe <- as.factor(validation_data$classe)
>
> # Train Random Forest model
> start_time <- Sys.time()
>
> # Check for NA values and handle them
> if(any(is.na(train_data))) {
+     cat("Warning: NA values found in training data. Removing them...\n")
+     train_data <- na.omit(train_data)
+ }
>
> # Train model
> rf_model <- randomForest(classe ~ .,
+                          data = train_data,
+                          ntree = 100,
+                          importance = TRUE,
+                          do.trace = FALSE,
+                          na.action = na.omit)
>
> training_time <- Sys.time() - start_time
>
> cat("Training completed in", round(training_time, 2), "minutes\n")
Training completed in 13.7 minutes
> cat("Model OOB error rate:", round(mean(rf_model$err.rate[,1]), 4), "\n")
Model OOB error rate: 0.0166
>
> # 5. MODEL EVALUATION
> # ====================
>
> cat("\nStep 5: Evaluating model...\n")
```

```
Step 5: Evaluating model...
>
> # Predict on validation set
> rf_predictions <- predict(rf_model, newdata = validation_data)
>
> # Create confusion matrix
> confusion_matrix <- confusionMatrix(rf_predictions, validation_data$classe)
>
> cat("\nConfusion Matrix:\n")

Confusion Matrix:
> print(confusion_matrix$table)
          Reference
Prediction    A    B    C    D    E
         A 1673    2    0    0    0
         B    1 1134   12    0    0
         C    0    3 1014    8    0
         D    0    0    0  955    0
         E    0    0    0    1 1082
>
> # Calculate accuracy
> accuracy <- confusion_matrix$overall["Accuracy"]
> cat("\nAccuracy:", round(accuracy, 4), "\n")

Accuracy: 0.9954
> cat("Out-of-sample error:", round(1 - accuracy, 4), "\n")
Out-of-sample error: 0.0046
> cat("95% Confidence Interval: [",
+     round(confusion_matrix$overall["AccuracyLower"], 4), ", ",
+     round(confusion_matrix$overall["AccuracyUpper"], 4), "]\n", sep = "")
95% Confidence Interval: [0.9933, 0.997]
>
> # 6. VARIABLE IMPORTANCE
> # ======================
>
> cat("\nStep 6: Analyzing variable importance...\n")

Step 6: Analyzing variable importance...
>
> # Get variable importance
> var_importance <- importance(rf_model)
> top_vars <- head(var_importance[order(-var_importance[, "MeanDecreaseGini"]), ,
drop = FALSE], 10)
>
> cat("\nTop 10 most important variables:\n")

Top 10 most important variables:
> print(top_vars[, "MeanDecreaseGini", drop = FALSE])
                  MeanDecreaseGini
roll_belt                853.7299
yaw_belt                 593.2022
pitch_forearm            570.5996
```

```
magnet_dumbbell_z         539.7443
magnet_dumbbell_y         485.9917
pitch_belt                479.5770
roll_forearm              417.7147
magnet_dumbbell_x         341.4781
magnet_belt_y             307.2524
roll_dumbbell             304.6719
>
> # 7. MAKE PREDICTIONS ON TEST SET
> # ================================
>
> cat("\nStep 7: Making predictions on test set...\n")

Step 7: Making predictions on test set...
>
> # Check if test data has all required columns
> # Ensure test data has same columns as training (excluding classe)
> training_features <- setdiff(names(train_data), "classe")
> missing_in_test <- setdiff(training_features, names(testing_clean))
>
> if(length(missing_in_test) > 0) {
+     cat("Adding missing columns to test data with 0 values...\n")
+     for(col in missing_in_test) {
+         testing_clean[[col]] <- 0  # Use 0 instead of NA
+     }
+ }
>
> # Make sure columns are in same order
> testing_clean <- testing_clean[, training_features]
>
> # Make predictions
> final_predictions <- predict(rf_model, newdata = testing_clean)
>
> cat("\n=== TEST PREDICTIONS ===\n")

=== TEST PREDICTIONS ===
> for(i in 1:length(final_predictions)) {
+     cat(sprintf("Problem %2d: %s\n", i, final_predictions[i]))
+ }
Problem  1: B
Problem  2: A
Problem  3: B
Problem  4: A
Problem  5: A
Problem  6: E
Problem  7: D
Problem  8: B
Problem  9: A
Problem 10: A
Problem 11: B
Problem 12: C
Problem 13: B
```

```
Problem 14: A
Problem 15: E
Problem 16: E
Problem 17: A
Problem 18: B
Problem 19: B
Problem 20: B
>
> # 8. CREATE PREDICTION FILES FOR COURSERA
> # ========================================
>
> cat("\nStep 8: Creating prediction files for Coursera...\n")

Step 8: Creating prediction files for Coursera...
>
> pml_write_files = function(x) {
+     n = length(x)
+     for(i in 1:n) {
+         filename = paste0("problem_id_", i, ".txt")
+         write.table(x[i], file = filename, quote = FALSE,
+                     row.names = FALSE, col.names = FALSE)
+     }
+ }
>
> pml_write_files(final_predictions)
>
> # 9. CREATE VISUALIZATIONS
> # ========================
>
> cat("\nStep 9: Creating visualizations...\n")

Step 9: Creating visualizations...
>
> # Plot 1: Class distribution
> png("class_distribution.png", width = 800, height = 600)
> par(mar = c(5, 6, 4, 2))
> class_dist <- table(train_data$classe)
> barplot(class_dist,
+         col = c("steelblue", "lightgreen", "lightcoral", "gold", "purple"),
+         main = "Distribution of Exercise Quality Classes (Training Set)",
+         xlab = "Class",
+         ylab = "Number of Observations",
+         las = 1,
+         ylim = c(0, max(class_dist) * 1.1))
> text(1:5, class_dist + max(class_dist)*0.03,
+      paste0(round(prop.table(class_dist)*100, 1), "%"),
+      col = "darkblue", cex = 1.2, font = 2)
> legend("topright",
+        legend = c("A: Correct", "B: Elbows front", "C: Halfway up",
+                   "D: Halfway down", "E: Hips front"),
+        fill = c("steelblue", "lightgreen", "lightcoral", "gold", "purple"),
+        cex = 0.8)
```

```
> dev.off()
RStudioGD
        2
>
> # Plot 2: Variable importance
> png("variable_importance.png", width = 1000, height = 600)
> par(mar = c(5, 12, 4, 2))
> bar_colors <- colorRampPalette(c("steelblue", "lightblue"))(10)
> barplot(rev(top_vars[, "MeanDecreaseGini"]),
+         names.arg = rev(rownames(top_vars)),
+         horiz = TRUE,
+         col = bar_colors,
+         las = 1,
+         main = "Top 10 Most Important Predictors",
+         xlab = "Mean Decrease Gini (Importance)")
> dev.off()
RStudioGD
        2
>
> # Plot 3: Error rate
> png("error_rate.png", width = 800, height = 600)
> par(mar = c(5, 6, 4, 2))
> plot(rf_model, main = "Random Forest Error Rate", lwd = 2)
> legend("topright",
+        legend = c("OOB Error", "Class A", "Class B", "Class C", "Class D", "Cla
ss E"),
+        col = 1:6,
+        lty = 1,
+        lwd = 2,
+        cex = 0.8)
> dev.off()
RStudioGD
        2
>
> # 10. GENERATE FINAL REPORT
> # ========================
>
> cat("\nStep 10: Generating final report...\n")

Step 10: Generating final report...
>
> report_text <- paste0(
+     "PRACTICAL MACHINE LEARNING COURSE PROJECT REPORT\n",
+     "================================================\n\n",
+     "1. PROJECT OVERVIEW\n",
+     "   - Goal: Predict weight lifting exercise quality (classe A-E)\n",
+     "   - Data: Accelerometer readings from 6 participants\n",
+     "   - Training samples: ", nrow(train_data), "\n",
+     "   - Validation samples: ", nrow(validation_data), "\n",
+     "   - Test cases: ", nrow(testing_clean), "\n\n",
+     "2. MODEL SPECIFICATIONS\n",
+     "   - Algorithm: Random Forest\n",
```

```
+     "   - Number of trees: 100\n",
+     "   - Training time: ", round(training_time, 2), " minutes\n\n",
+     "3. PERFORMANCE METRICS\n",
+     "   - Accuracy: ", round(accuracy * 100, 2), "%\n",
+     "   - Out-of-sample error: ", round((1 - accuracy) * 100, 2), "%\n",
+     "   - 95% CI: [", round(confusion_matrix$overall["AccuracyLower"] * 100, 2)
,
+     "%, ", round(confusion_matrix$overall["AccuracyUpper"] * 100, 2), "%]\n",
+     "   - Kappa: ", round(confusion_matrix$overall["Kappa"], 3), "\n\n",
+     "4. TOP 5 PREDICTORS\n",
+     paste0("   ", 1:5, ". ", rownames(top_vars)[1:5],
+            " (", round(top_vars[1:5, "MeanDecreaseGini"], 1), ")\n", collapse =
""),
+     "\n5. TEST PREDICTIONS\n"
+ )
>
> for(i in 1:length(final_predictions)) {
+     report_text <- paste0(report_text,
+                           sprintf("   Problem %2d: %s\n", i, final_predictions[
i]))
+ }
>
> report_text <- paste0(report_text,
+                       "\n6. CONCLUSION\n",
+                       "   The Random Forest model achieved excellent performanc
e with ",
+                       round(accuracy * 100, 1), "% accuracy. The estimated out-
of-sample\n",
+                       "   error is ", round((1 - accuracy) * 100, 1), "%, indic
ating strong generalization\n",
+                       "   capability. Belt sensor measurements were the most im
portant predictors.\n"
+ )
>
> writeLines(report_text, "project_report.txt")
>
> # 11. SAVE MODEL
> # ==============
>
> saveRDS(rf_model, "random_forest_model.rds")
> cat("\nModel saved as 'random_forest_model.rds'\n")

Model saved as 'random_forest_model.rds'
>
> # 12. FINAL OUTPUT
> # ================
>
> cat("\n", rep("=", 60), "\n", sep = "")

============================================================
> cat("ANALYSIS COMPLETE - READY FOR COURSERA SUBMISSION\n")
ANALYSIS COMPLETE - READY FOR COURSERA SUBMISSION
```

```
> cat(rep("=", 60), "\n\n", sep = "")
============================================================

>
> cat("SUMMARY OF RESULTS:\n")
SUMMARY OF RESULTS:
> cat("1. Model: Random Forest with 100 trees\n")
1. Model: Random Forest with 100 trees
> cat("2. Accuracy: ", round(accuracy * 100, 2), "%\n")
2. Accuracy:  99.54 %
> cat("3. Out-of-sample error: ", round((1 - accuracy) * 100, 2), "%\n")
3. Out-of-sample error:  0.46 %
> cat("4. Top predictor: ", rownames(top_vars)[1], "\n\n")
4. Top predictor:  roll_belt

>
> cat("FILES CREATED:\n")
FILES CREATED:
> cat("1. 20 prediction files (problem_id_1.txt to problem_id_20.txt)\n")
1. 20 prediction files (problem_id_1.txt to problem_id_20.txt)
> cat("2. project_report.txt - Summary report\n")
2. project_report.txt - Summary report
> cat("3. random_forest_model.rds - Trained model\n")
3. random_forest_model.rds - Trained model
> cat("4. class_distribution.png - Visualization 1\n")
4. class_distribution.png - Visualization 1
> cat("5. variable_importance.png - Visualization 2\n")
5. variable_importance.png - Visualization 2
> cat("6. error_rate.png - Visualization 3\n\n")
6. error_rate.png - Visualization 3

>
> cat("PREDICTIONS FOR 20 TEST CASES:\n")
PREDICTIONS FOR 20 TEST CASES:
> pred_df <- data.frame(
+     Problem_ID = 1:20,
+     Prediction = as.character(final_predictions),
+     Description = ifelse(final_predictions == "A", "Correct execution",
+                          ifelse(final_predictions == "B", "Elbows to front",
+                                 ifelse(final_predictions == "C", "Halfway up",
+                                        ifelse(final_predictions == "D", "Halfwa
y down",
+                                               "Hips to front"))))
+ )
> print(pred_df)
   Problem_ID Prediction       Description
1           1          B   Elbows to front
2           2          A Correct execution
3           3          B   Elbows to front
4           4          A Correct execution
5           5          A Correct execution
6           6          E     Hips to front
```

```
7           7          D      Halfway down
8           8          B   Elbows to front
9           9          A Correct execution
10         10          A Correct execution
11         11          B   Elbows to front
12         12          C        Halfway up
13         13          B   Elbows to front
14         14          A Correct execution
15         15          E     Hips to front
16         16          E     Hips to front
17         17          A Correct execution
18         18          B   Elbows to front
19         19          B   Elbows to front
20         20          B   Elbows to front
>
> cat("\nTO SUBMIT TO COURSERA:\n")

TO SUBMIT TO COURSERA:
> cat("1. Upload the 20 .txt files to the Prediction Quiz\n")
1. Upload the 20 .txt files to the Prediction Quiz
> cat("2. For Peer Review: Submit R code and compiled HTML report\n")
2. For Peer Review: Submit R code and compiled HTML report
> cat("3. Working directory: ", getwd(), "\n")
3. Working directory:  /home/rstudio
> cat("\nAll files are ready for submission!\n")

All files are ready for submission!
```