Assignment error minimum reproducible example

Loading libraries and raw data

```
library(rpart)
library(caret)
```

```
pml_training_smaller <- read.csv("pml_training_smaller.csv")
pml_testing_smaller  <- read.csv("pml_testing_smaller.csv")
```

```
dim(pml_training_smaller); dim(pml_testing_smaller)
```

```
## [1] 19622    53
```

```
## [1] 20 53
```

```
names_training <- names(pml_training_smaller)
names_testing <- names(pml_testing_smaller)
print(names_training)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

```
names_training[!(names_training %in% names_testing)]; names_testing[!(names_testing %in% names_training
```

```
## character(0)
```

```
## character(0)
```

So these two input files contain data with the same number of columns and there is nothing that appears in the list of testing column names that does not appear in the list of training column names, and vice-versa.

The outcome to be predicted is the variable 'classe' (i.e. the final column). This is a factor with 5 levels, A, B, C, D, and E. All other columns used to predict this outcome are numeric or integer class.

Predicting using partitioned subsets of raw training data as my testing and training data

For this part, original testing data loaded from the file “pml_testing_smaller.csv” is irrelevant, because a new testing set is created by partitioning the original training data set:

```
set.seed(1)
partition <- createDataPartition(pml_training_smaller$classe, p=0.7, list=FALSE)
training_subset <- pml_training_smaller[partition, ]
testing_subset <- pml_training_smaller[-partition, ]
dim(training_subset); dim(testing_subset)
```

```
## [1] 13737    53
```

```
## [1] 5885   53
```

Now fit and predict:

```
modfit_pml_testing_rpart_1  <- rpart(classe ~ ., data = training_subset, method = "class")
predict_pml_testing_rpart_1 <- predict(modfit_pml_testing_rpart_1, testing_subset, type = "class")
confmat_pml_testing_rpart_1 <- confusionMatrix(predict_pml_testing_rpart_1, testing_subset$classe)
confmat_pml_testing_rpart_1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1473  160   24   44   15
##          B   55  699   56   73  101
##          C   57  121  818  161  132
##          D   56   82   60  609   53
##          E   33   77   68   77  781
##
## Overall Statistics
##
##                Accuracy : 0.7443
##                  95% CI : (0.7329, 0.7554)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6764
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8799   0.6137   0.7973   0.6317   0.7218
## Specificity            0.9423   0.9399   0.9031   0.9490   0.9469
## Pos Pred Value         0.8584   0.7104   0.6346   0.7081   0.7539
## Neg Pred Value         0.9518   0.9102   0.9547   0.9294   0.9379
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2503   0.1188   0.1390   0.1035   0.1327
## Detection Prevalence   0.2916   0.1672   0.2190   0.1461   0.1760
## Balanced Accuracy      0.9111   0.7768   0.8502   0.7904   0.8344
```

So what I have here is a sensible result based on a training and testing set that have identical format because they are subsets of the same wider set (which is the original training data, ‘pml_training_smaller’). So far so good.

Predicting using the whole original training data as my training set, and the original testing data as my testing data set

Originally the testing data found in the file “pml_testing_smaller.csv” did not have the ‘classe’ column that the training data found in the file “pml_training_smaller.csv” has. Instead its name was ‘problem_id’, and in each cell, it simply contained an integer (i.e. the number of the row in question).

I modified this column in “pml_testing_smaller.csv” myself before creating this document, so that it has the same column name as in “pml_training_smaller.csv” (i.e. ‘classe’), and that it contains fake/made-up factor values, A, B, C, D, or E. Thus, it has the same name, type and levels as the corresponding column in “pml_training_smaller.csv”, and I therefore have the same situation as what I had in the previous section with the partitioned data.

So, unsurprisingly, I can generate a confusion matrix in the same manner as I did in the previous section:

```r
modfit_pml_testing_rpart_2  <- rpart(classe ~ ., data = pml_training_smaller, method = "class")
predict_pml_testing_rpart_2 <- predict(modfit_pml_testing_rpart_2, pml_testing_smaller, type = "class")
confmat_pml_testing_rpart_2 <- confusionMatrix(predict_pml_testing_rpart_2, pml_testing_smaller$classe)
confmat_pml_testing_rpart_2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction A B C D E
##          A 0 2 1 2 2
##          B 1 1 2 1 1
##          C 2 0 1 0 0
##          D 0 1 0 1 0
##          E 1 0 0 0 1
##
## Overall Statistics
##
##                Accuracy : 0.2
##                  95% CI : (0.0573, 0.4366)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : 0.5886
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.0000   0.2500   0.2500   0.2500   0.2500
## Specificity            0.5625   0.6875   0.8750   0.9375   0.9375
## Pos Pred Value         0.0000   0.1667   0.3333   0.5000   0.5000
## Neg Pred Value         0.6923   0.7857   0.8235   0.8333   0.8333
## Prevalence             0.2000   0.2000   0.2000   0.2000   0.2000
## Detection Rate         0.0000   0.0500   0.0500   0.0500   0.0500
## Detection Prevalence   0.3500   0.3000   0.1500   0.1000   0.1000
## Balanced Accuracy      0.2812   0.4688   0.5625   0.5938   0.5938
```

It works as expected (despite the accuracy being low; this is because I faked the values in the ‘classe’ column of the testing set used in this instance, whereas in the previous section, the corresponding column had real values).

The actual problem I’m having

However, my problem is that rather than faking the values, I am supposed to be predicting what these ‘classe’ values should really be, given all the other columns in “pml_testing_smaller.csv” (the ‘classe’ values it currently has are fake ones I added to see what happens).

I will now put that column back to the way it is supposed to be:

```
pml_testing_smaller$classe <- c(1:20)
colnames(pml_testing_smaller)[ncol(pml_testing_smaller)] <- "problem_id"
#str(pml_testing_smaller)
```

Now I try generating a confusion matrix in the same manner as I did in the previous section:

```
modfit_pml_testing_rpart_3  <- rpart(classe ~ ., data = pml_training_smaller, method = "class")
predict_pml_testing_rpart_3 <- predict(modfit_pml_testing_rpart_3, pml_testing_smaller, type = "class")
confmat_pml_testing_rpart_3 <- confusionMatrix(predict_pml_testing_rpart_3, pml_testing_smaller$classe)
```

```
## Error in confusionMatrix.default(predict_pml_testing_rpart_3, pml_testing_smaller$classe): the data
```

```
confmat_pml_testing_rpart_3
```

```
## Error in eval(expr, envir, enclos): object 'confmat_pml_testing_rpart_3' not found
```

Thus, I’m told that “the data cannot have more levels than the reference”. I get the same error message if I remove the column entirely.

How am I supposed to predict the outcome ‘classe’ for each row of my testing data stored in ‘pml_testing_smaller’?
