practical machine learning project

Kevin B

1/17/2022

First load the data and remove columns with NA values. As well as remove the first 7 columns with variables not pertinent to the exercises

```
pml_validation <- read_csv("C:/Users/bucze/Downloads/pml-testing.csv",
                           na = c("NA", "#DIV/0!", ""))
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_logical(),
##   X1 = col_double(),
##   user_name = col_character(),
##   raw_timestamp_part_1 = col_double(),
##   raw_timestamp_part_2 = col_double(),
##   cvtd_timestamp = col_character(),
##   new_window = col_character(),
##   num_window = col_double(),
##   roll_belt = col_double(),
##   pitch_belt = col_double(),
##   yaw_belt = col_double(),
##   total_accel_belt = col_double(),
##   gyros_belt_x = col_double(),
##   gyros_belt_y = col_double(),
##   gyros_belt_z = col_double(),
##   accel_belt_x = col_double(),
##   accel_belt_y = col_double(),
##   accel_belt_z = col_double(),
##   magnet_belt_x = col_double(),
##   magnet_belt_y = col_double(),
##   magnet_belt_z = col_double()
##   # ... with 40 more columns
## )
## i Use `spec()` for the full column specifications.
```

```
pml_training <-read.csv("C:/Users/bucze/Downloads/pml-training.csv",
                           na = c("NA", "#DIV/0!", ""))

cleanvalidation <- pml_validation[, colSums(is.na(pml_validation)) == 0]
```

```
cleantrain <- pml_training[, colSums(is.na(pml_training)) == 0]

cleantrain <- cleantrain[, -c(1:7)]
cleanvalidation <- cleanvalidation[, -c(1:7)]
validation <- cleanvalidation
```

Then, partition the training data set so that I can validate the model before performing the final test with the test set.

```
set.seed(430)
inTrain <- createDataPartition(y=cleantrain$classe,
                               p=0.75, list = FALSE)
training <- cleantrain[inTrain,]
testing <- cleantrain[-inTrain,]
```

When the goal of a model is prediction, there's a lot more freedom for model choice because you don't need to be able to derive any knowledge from the model, as would be the case if the goal is for inference. We don't have to understand the model as long as it predicts what we want it to predict.

The goal here is classification, not predicting a quantitative outcome. For this type of problem I thought that random forest or k means clustering would be the best method, and in this case I chose random forest as my method.

```
#I was getting errors so I had to convert classe to a factor variable instead of a character.
training$classe = factor(training$classe); testing$classe = factor(testing$classe)

modRF <- train(classe ~ ., data = training, method="rf", ntree = 100)
modRF
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9891448  0.9862676
##   27    0.9884031  0.9853296
##   52    0.9788997  0.9733058
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Then checking the model with our testing set.

```
pred <- predict(modRF, testing)
confusionMatrix(pred, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    9    0    0    0
##          B    0  940    9    0    0
##          C    0    0  846    5    0
##          D    0    0    0  796    0
##          E    0    0    0    3  901
##
## Overall Statistics
##
##                Accuracy : 0.9947
##                  95% CI : (0.9922, 0.9965)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9933
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9905   0.9895   0.9900   1.0000
## Specificity            0.9974   0.9977   0.9988   1.0000   0.9993
## Pos Pred Value         0.9936   0.9905   0.9941   1.0000   0.9967
## Neg Pred Value         1.0000   0.9977   0.9978   0.9981   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1917   0.1725   0.1623   0.1837
## Detection Prevalence   0.2863   0.1935   0.1735   0.1623   0.1843
## Balanced Accuracy      0.9987   0.9941   0.9941   0.9950   0.9996
```

We get an accuracy of 99% using our random forest model on the test set.

And now to run our validation set through the model to see what classifiers we get.

```
validation_predict <- predict(modRF, validation)
validation_predict
```

```
##  [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
```