Practical Machine Learning Course Project

Sarah Gartenmann

2023-01-31

Introduction

This is the final report for Coursera's Practical Machine Learning course, which is a part of the Data Science Specialization track offered by the John Hopkins University. Below is a copy of the course project description:

> Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

In this project, I predicted whether or not six individuals performed their barbell lifts correctly or not based on various accelerometer data. The ensuring report describes how I tidied the data, tested and evaluated four models based on their accuracy, how I used cross validation, and how I used the final model to predict 20 different test cases.

Loading Data and Required Packages

I first loaded the required R packages and set the seed for data reproducibility.

```r
# load required packages
library(tidyverse); library(caret); library(rpart); library(rpart.plot); library(corrplot); library(rat
set.seed(0807)
```

Next, I downloaded the training (`Training`) and test (`Test`) datasets from the provided URLs and read them into R. All empty cells in both data sets were read in as "NA". The `Training` and `Test` datasets are both comprised of 160 variables with 20 and 19622 observations, respectively.

```r
# set the URL for the download
URLtraining <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
URLtesting <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# download and read in the datasets
Training <- read.csv(url(URLtraining), na.strings = c("", "NA"));
Testing  <- read.csv(url(URLtesting), na.strings = c("", "NA"))
```

```r
# check dimensions
dim(Training); dim(Testing)
```

```
## [1] 19622   160
```

```
## [1]  20 160
```

Tidying and Splitting the Data

I first tidied the data in both the `Training` and `Testing` datasets as many of the 160 variables actually did not contain any observations. All columns containing "NA" or irrelevant identification variables were removed. This left me with a total of 54 variables in both data sets, now called `TidyTraining` and `TidyTesting`, respectively.

```r
# removing NA columns
TidyTraining <- Training[ , colSums(is.na(Training))==0]
TidyTesting <- Testing[ , colSums(is.na(Testing))==0]

# removing columns with identification variables
TidyTraining <- TidyTraining[, -(1:6)]
TidyTesting  <- TidyTesting[, -(1:6)]

# check dimensions
dim(TidyTraining); dim(TidyTesting)
```

```
## [1] 19622    54
```

```
## [1] 20 54
```

I split the `TidyTraining` dataset into a `TrainSet` (70%) and a `ValidationSet` (30%) set using the `createDataPartition()` command. The `TrainSet` set was used henceforth for the modeling process, and the `ValidationSet` set for the validation. The `TidyTesting` dataset remained untouched until the end when I actually ran our final model. Additionally, I set up a control for the `TrainSet` using the `trainControl` command and setting the arguments `method = "cv"` and `number = 3` to indicate three-fold cross-validation. Cross-validation allows us to measure the performance of a predictive model on new test data sets by fitting a statistical method multiple times using different data subsets. In this case, this was done three times.

```r
# create a partition with the training dataset and checking dimensions
inTrain  <- caret::createDataPartition(TidyTraining$classe, p=0.7, list=FALSE)
TrainSet <- TidyTraining[inTrain, ]
ValidationSet  <- TidyTraining[-inTrain, ]
dim(TrainSet); dim(ValidationSet)
```

```
## [1] 13737    54
```

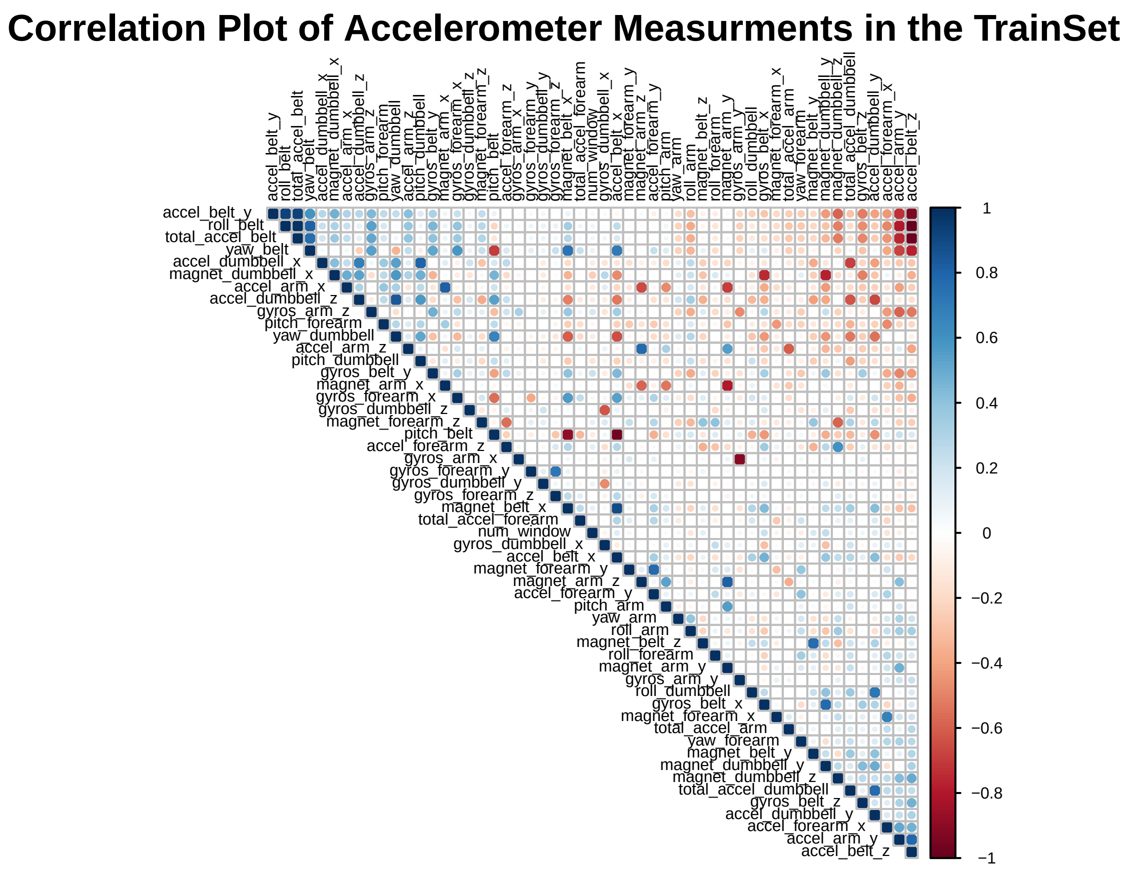
```
## [1] 5885   54
```

```r
# control for three-fold cross validation
control <- caret::trainControl(method="cv", number=3, verboseIter=F)
```

Exploring the Data

To explore the data of the `TrainSet`, I first generated a correlation plot. Using the command `cor()`, a correlation matrix was generated using all columns of the `TrainSet`, except the 54th column (the character variable "classe", the variable that we want to predict). The command `corrplot()` generates the correlation plot and we can see that only a few variables are highly correlated with each other (large dark red and dark blue circles). The term `order = "FPC"` denotes the use of the first principal component.

```r
corMatrix <- cor(TrainSet[, -54])
corrplot::corrplot(corMatrix, type = "upper", order = "FPC", cl.cex = 0.5, tl.cex = 0.5, tl.col = rgb(0
```



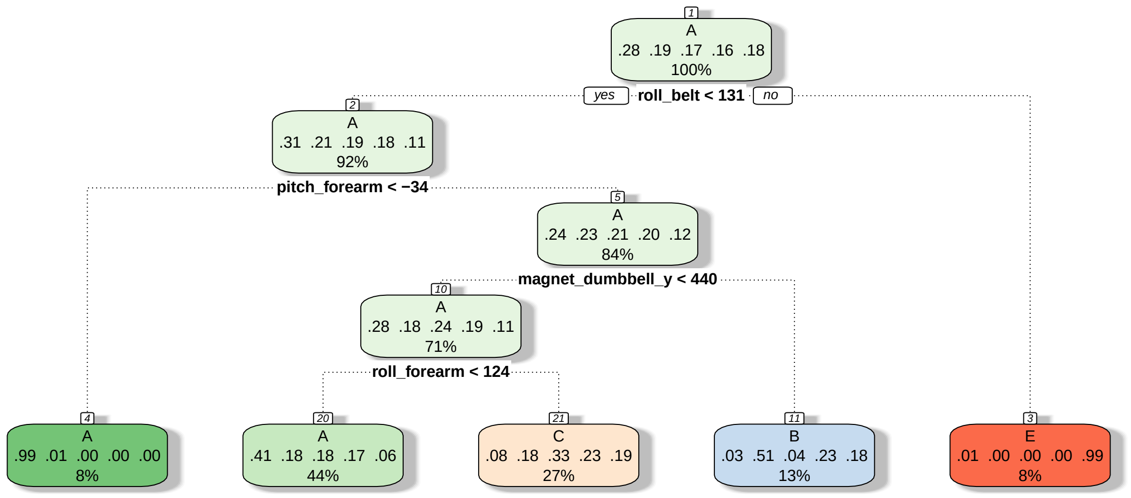
## Model Creation and Testing I applied four different methods to the `TrainSet` and the one with the highest accuracy on the `ValidationSet` was then used for the final prediction of the `TidyTesting` data set. The first model I applied to the data was a Decision Tree Model by using the `method = "rpart"` on the `TrainSet` and subsequently generating the tree using the `fancyRpartPlot` command of the `rattle` package. To evaluate the model, I generated a confusion matrix which categorized the predictions against their actual values. In the interest of keeping this report concise, I decided to only print the accuracy value instead of the entire matrix. For this model, the accuracy of was 49.11%.

```r
# Decision Tree
mod_tree <- caret::train(classe~., data=TrainSet, method="rpart", trControl=control)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
rattle::fancyRpartPlot(mod_tree$finalModel)
```



Rattle 2023−Jan−31 15:22:15 gartens1

```
pred_tree <- predict(mod_tree, ValidationSet)
conMatrix_tree <- confusionMatrix(pred_tree, factor(ValidationSet$classe))
print(conMatrix_tree$overall[1])
```

```
##  Accuracy
## 0.4937978
```

The second model that I tested was a Random Forest Model. This model is generally very good for dealing with large numbers of features. The model draws `ntree` bootstrap samples (the default number of trees is `ntree = 500`, but I decreased it to `ntree = 100` so that the model runs a bit quicker) from which it then grows an un-pruned tree by choosing the best split based on a random sample of `mtry` predictors at each node. Based on majority vote, predictions on new data are then done based on the best performing tree. I applied the `method = "rf"` to the `TestSet` and generated a confusion matrix to evaluate the model. This model has an accuracy of 99.8%.

```
# Random Forest
mod_forest <- caret::train(classe~., data=TrainSet, method="rf", trControl=control, ntree=100)
pred_forest <- predict(mod_forest, ValidationSet)
conMatrix_forest <- confusionMatrix(pred_forest, factor(ValidationSet$classe))
print(conMatrix_forest$overall[1])
```

```
##  Accuracy
## 0.9979609
```

The third model that I tested was a boosted model. Boosting combines a subset of classifiers by taking a lot of (possibly) weak predictors, weighting and summing them up, to create a stronger predictor and thereby producing a more accurate model. R has multiple boosting libraries and I applied a Gradient Boosted Model which is based on trees. For this I set the `method = "gbm"`. The confusion matrix gave an accuracy of 98.95% for this model.

```
# Gradient Boosted Model
mod_gbm <- caret::train(classe~., data=TrainSet, method="gbm", verbose=FALSE, trControl=control)
pred_gbm <- predict(mod_gbm, ValidationSet)
conMatrix_gbm <- confusionMatrix(pred_gbm, factor(ValidationSet$classe))
print(conMatrix_gbm$overall[1])
```

```
##  Accuracy
## 0.9892948
```

The fourth and last model tested was a Support Vector Machine (SVM) which is a supervised learning model with associated learning algorithms that analyses the data used for classification and regression. The SVM model places a decision boundary between different datapoints to differentiate between them. Here, I set the `method = "svmLinear"` and the confusion matrix gives us an accuracy of 79.25%.

```
mod_svm <- caret::train(classe~., data=TrainSet, method="svmLinear", verbose=FALSE)
pred_svm <- predict(mod_svm, ValidationSet)
conMatrix_svm <- confusionMatrix(pred_svm, factor(ValidationSet$classe))
print(conMatrix_svm$overall[1])
```

```
## Accuracy
## 0.782328
```

To evaluate which of the four models is the best to apply to the `TidyTesting` dataset, I compared their accuracy and their out of sample error in a neat table. The accuracy tells us the percentage of correct predictions, and the out of sample error rate tells us the error rate we get when running the model on a new data set. From the table below we can see that the Random Forest Model is the best performing model.

```
# generate a table with accuracy of each model
accuracy <- round(c(conMatrix_tree$overall[1], conMatrix_forest$overall[1], conMatrix_gbm$overall[1], c
oos <- 1 - accuracy

Accuracy_Results <- data.frame(
  Accuracy = accuracy,
  'Out of Sample Error' = oos,
  row.names = c('Decision Tree Model', 'Random Forest Model', 'Generalised Boosted Model', 'Support Vec

print(Accuracy_Results)
```

```
##                               Accuracy Out.of.Sample.Error
## Decision Tree Model              0.494               0.506
## Random Forest Model              0.998               0.002
## Generalised Boosted Model        0.989               0.011
## Support Vector Machine Model     0.782               0.218
```

Prediction Model on Test Set

We then applied the final model `mod_forest`on the `TidyTesting` dataset to see if it can predict what `classe` each of the 20 observations belong to.

```r
PredictionFinal <- predict(mod_forest, TidyTesting)
PredictionFinal_Results <- data.frame(
  problem_id=TidyTesting$problem_id,
  predicted=PredictionFinal
)
print(PredictionFinal_Results)
```

```
##    problem_id predicted
## 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
```

Conclusion

I was able to fit the data in a Random Forest Model with cross validation with a high degree of accuracy. There are of course many additional models available that I could have tested, however I feel that with this Random Forest Model showing over 99% accuracy, I can already now confidently say that this model is sufficient for predictive analytics.