Practical Machine Learning - Project

AUTHOR
Cooper Carpenter

PUBLISHED
August 7, 2024

Introduction

As the final project for the Practice Machine Learning course, I predicted how a person lifted a barbell via data collected from various accelerometers placed across 6 participants' bodies. Using the "classe" variable as my outcome I ran three different models using a k-fold cross-validation. A validation data set was used to obtain the accuracy and out-of-sample error for the models. Using these results, I applied the top performing model to 20 test cases.

Please note, the url with additional documentation did not work at the time of this project. So detailed information on variables types will not be available in this report.

Loading Data

The data and packages required for this project are loaded below. There are 19222 observations and 160 variables in the initial training data.

```
# Loading Necessary Libraries and Checking Location
library(tidyverse)
library(magrittr)
library(caret)

# Loading data
training <- read_csv("data/pml-training.csv")
testing <- read_csv("data/pml-testing.csv")

# checking data sets dimensions
dim(training)
```

```
[1] 19622   160
```

```
dim(testing)
```

```
[1]  20 160
```

Data Cleaning

In the code block below, I clean the data set by removing metadata unrelated to the participants' physical movement. Additionally, I drop variables that are predominantly missing (>90%) , and highly correlated variables (> 0.9 correlation). I tested for variables with near zero variance, but none were identified after dropping predominantly missing variables. After cleaning, the training data has 46 variables. A correlation plot of these variables can be found in the appendix.

```
set.seed(111)
# removing meta data
training <- training[,-c(1:7)]

# removing variables that are predominantly NA (>90%)
drop_missing <- training %>%
  summarise(across(everything(), ~sum(is.na(.))/n())) %>%
  pivot_longer(everything(), names_to = "var", values_to = "prop_missing") %>%
  filter(prop_missing > .9) %>% pull(var)

training <- training %>%  select(-all_of(drop_missing))

# removing variables with near zero variance - no cases
drop_nzv <- nearZeroVar(training, names = TRUE)

# removing highly correlated variables
cors <- cor(training[,-length(training)])

high_cor <- findCorrelation(cors,
                            cutoff = 0.9,
                            names = TRUE)

training <- training %>% select(-all_of(high_cor))

rm(drop_missing, drop_nzv, high_cor, cors)

dim(training)
```

```
[1] 19622    46
```

Splitting Data

The cleaned training data is split into a training and validation data set below. The validation data set will be used to determine the models' performance.

```
# making a validation data set out of training data
set.seed(123)
inTrain <- createDataPartition(y = training$classe, p = .7, list = FALSE)
train <- training[inTrain,]
valid <- training[-inTrain,]
rm(inTrain)
```

Fitting Models

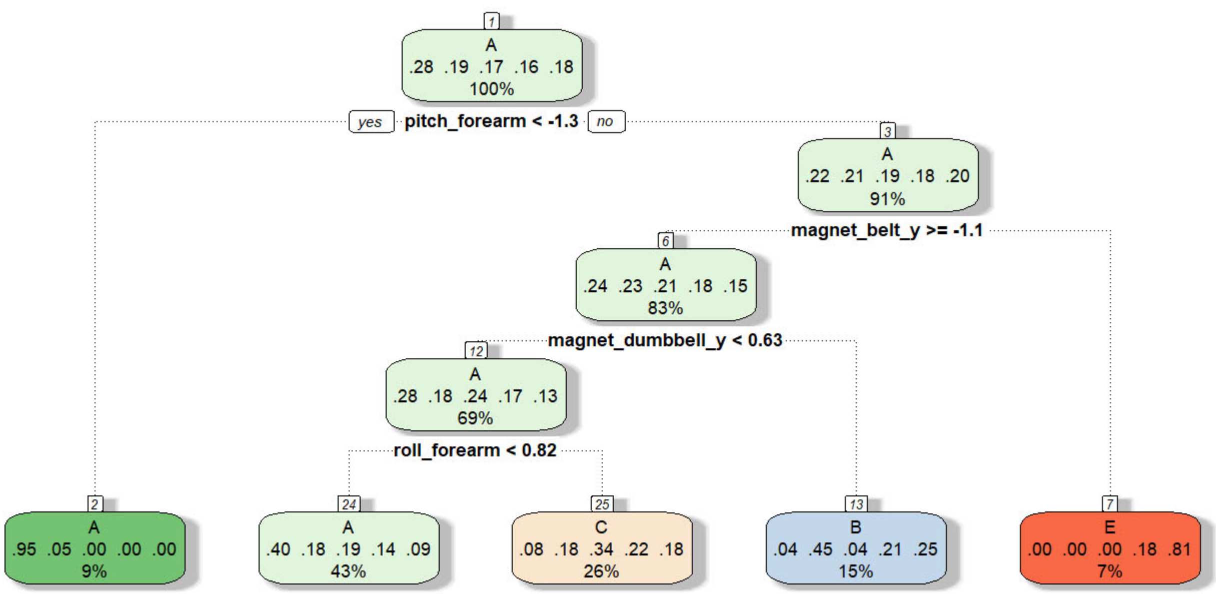
I fit three different model types: decision tree, random forest, and gradient boost trees. I predicted the observation's classe for all the models via all remaining variables in the cleaned training data. Each model also used a 3-fold cross-validation set up below.

```
# setting k-fold cross validation
cv <- trainControl(method = "cv", number = 3, verboseIter = FALSE)
```

Decision tree

The code for fitting and predicting data via a decision tree model is below. A plot of the decision is also included.

```
# fitting
set.seed(234)
mod_tree <- train(classe ~ .,
                  method = "rpart",
                  preProcess = c("center", "scale"),
                  trControl = cv,
                  data = train)
rattle::fancyRpartPlot(mod_tree$finalModel)
```



Rattle 2024-Aug-07 09:45:42 CooperCarpenter

```
# predicting
pred_tree <- predict(mod_tree, valid)
confusionMatrix(factor(valid$classe), pred_tree)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1521   36  115    0    2
         B  450  410  279    0    0
         C  457   40  529    0    0
         D  369  185  328    0   82
         E  203  220  291    0  368

Overall Statistics

               Accuracy : 0.4805
                 95% CI : (0.4677, 0.4934)
    No Information Rate : 0.5098
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3218

 Mcnemar's Test P-Value : <2e-16

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.5070  0.46016  0.34306       NA  0.81416
Specificity            0.9470  0.85402  0.88556   0.8362  0.86858
Pos Pred Value         0.9086  0.35996  0.51559       NA  0.34011
Neg Pred Value         0.6488  0.89865  0.79152       NA  0.98251
Prevalence             0.5098  0.15140  0.26202   0.0000  0.07681
Detection Rate         0.2585  0.06967  0.08989   0.0000  0.06253
Detection Prevalence   0.2845  0.19354  0.17434   0.1638  0.18386
Balanced Accuracy      0.7270  0.65709  0.61431       NA  0.84137
```

Random Forest

The code for fitting and predicting data via random forest model.

```
# fitting
set.seed(345)
mod_rf <- train(classe ~ .,
                method = "rf",
                preProcess = c("center", "scale"),
                trControl = cv,
                data = train)
#plot(mod_rf)
# prediciting
```

```
pred_rf <- predict(mod_rf, valid)
confusionMatrix(factor(valid$classe), pred_rf)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1673    1    0    0    0
         B    8 1125    6    0    0
         C    0    6 1019    1    0
         D    0    0    5  958    1
         E    0    0    4    4 1074

Overall Statistics

               Accuracy : 0.9939
                 95% CI : (0.9915, 0.9957)
    No Information Rate : 0.2856
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9923

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9952   0.9938   0.9855   0.9948   0.9991
Specificity            0.9998   0.9971   0.9986   0.9988   0.9983
Pos Pred Value         0.9994   0.9877   0.9932   0.9938   0.9926
Neg Pred Value         0.9981   0.9985   0.9969   0.9990   0.9998
Prevalence             0.2856   0.1924   0.1757   0.1636   0.1827
Detection Rate         0.2843   0.1912   0.1732   0.1628   0.1825
Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
Balanced Accuracy      0.9975   0.9954   0.9920   0.9968   0.9987
```

Gradient Boost

The code for fitting and predicting data via gradient boost trees model.

```
# fitting
set.seed(456)
mod_gbm <- train(classe ~ .,
                 method = "gbm",
                 preProcess = c("center", "scale"),
                 trControl = cv,
                 verbose = FALSE,
                 data = train)
# prediciting
```

```
pred_gbm <- predict(mod_gbm, valid)
confusionMatrix(factor(valid$classe), pred_gbm)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1647   16    5    3    3
         B   30 1065   39    1    4
         C    0   31  982   11    2
         D    2   11   32  908   11
         E    3   14   15   14 1036

Overall Statistics

               Accuracy : 0.958
                 95% CI : (0.9526, 0.963)
    No Information Rate : 0.2858
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9469

 Mcnemar's Test P-Value : 2.342e-06

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9792   0.9367   0.9152   0.9691   0.9811
Specificity            0.9936   0.9844   0.9909   0.9887   0.9905
Pos Pred Value         0.9839   0.9350   0.9571   0.9419   0.9575
Neg Pred Value         0.9917   0.9848   0.9813   0.9941   0.9958
Prevalence             0.2858   0.1932   0.1823   0.1592   0.1794
Detection Rate         0.2799   0.1810   0.1669   0.1543   0.1760
Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
Balanced Accuracy      0.9864   0.9605   0.9530   0.9789   0.9858
```

Model Results

The accuracy and out-of-sample error rates of the three different models are presented below. Random forest was the top performing model with an accuracy of 0.994 and out-of-sample error rate of 0.006 and will be used to predict the test cases.

```
mod_results <- tibble(
  Model = c("Decision Tree", "Random Forest", "Gradient Boost"),
  Accuracy = c(confusionMatrix(factor(valid$classe), pred_tree)$overall[1],
               confusionMatrix(factor(valid$classe), pred_rf)$overall[1],
               confusionMatrix(factor(valid$classe), pred_gbm)$overall[1]),
  `Out-of-Sample Error` = 1- Accuracy)
```

```
# rounding data
mod_results <- mod_results %>%
  mutate(Accuracy = round(Accuracy, 4),
         `Out-of-Sample Error` = round(`Out-of-Sample Error`, 4))

mod_results
```

```
# A tibble: 3 x 3
  Model          Accuracy `Out-of-Sample Error`
  <chr>             <dbl>                 <dbl>
1 Decision Tree     0.480                0.520
2 Random Forest     0.994                0.0061
3 Gradient Boost    0.958                0.042
```

Predicting Test Cases

Predictions of the 20 test cases using the random forest model are presented below. Classe “B” was predicted for 8 cases while classe “A” was predicted for 7. The remaining 5 cases were predicted for classes “C”, “D”, and “E”. As the testing data set does not have a classe variable, the accuracy of these predictions can not be obtained.

```
pred_rf_test <- predict(mod_rf, testing)

# list of predictions
pred_rf_test
```

```
 [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
```

```
# table of predictions
table(pred_rf_test)
```

```
pred_rf_test
A B C D E
7 8 1 1 3
```

Appendix

The correlation plot for the cleaned training data is displayed below.

```
cors <- cor(training[,-length(training)])
corrplot::corrplot(cors, method="color")
```