Practical ML Project

Anthony Cazares

2023-04-29

```
library(AppliedPredictiveModeling)
library(caret)
library(lubridate)
library(pgmm)
library(rpart)
library(gbm)
library(forecast)
library(xgboost)
library(Amelia)
library(randomForest)
library(tidyverse)
```

Load Data

```
testing <- read.csv("~\\GitHub\\JohnsHopkinsCourse\\datasciencecoursera\\PracticalML\\Project\\pml-testing.csv")

training <- read.csv("~\\GitHub\\JohnsHopkinsCourse\\datasciencecoursera\\PracticalML\\Project\\pml-training.cs
v")
```

Create Data Partitions

This will allow me to estimate the out of sample accuracy/error rate. Will create a partition of the testing data as a test set.

```
inTrain <- createDataPartition(training$classe,times = 1, p = 0.8, list = FALSE)
training <- training[inTrain,]
testtrain <- training[-inTrain,]
```

Fix Data

The “classe” value is a factor variable.

```
training$classe <- as.factor(training$classe)
testtrain$classe <- as.factor(testtrain$classe)
```

Find variables without missing values to not have NA problems in fitting models.

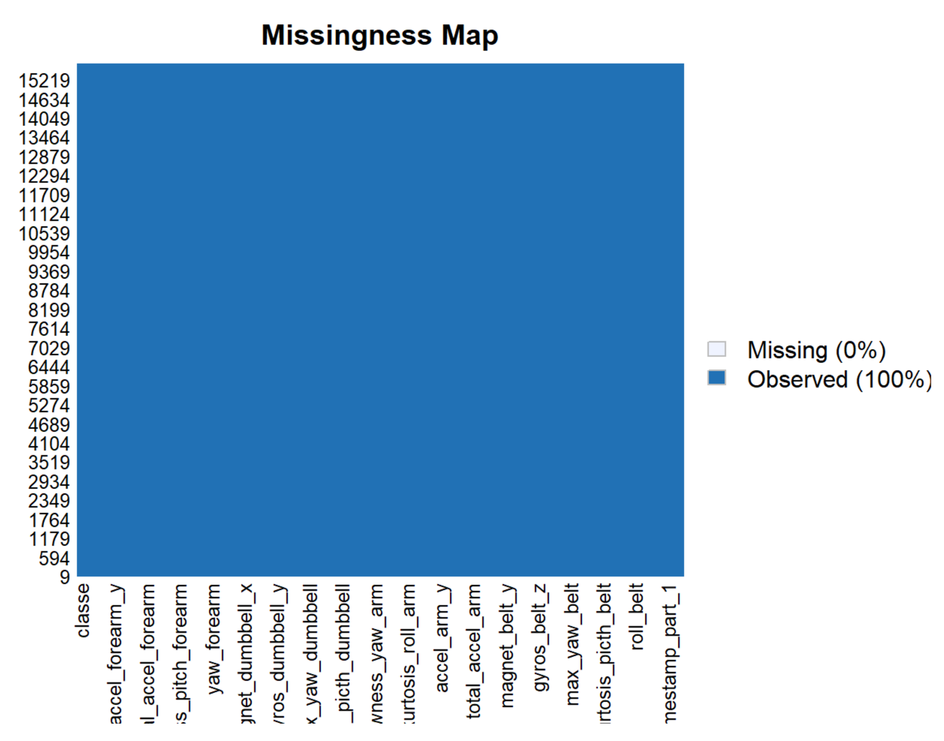
```
fullvars <- c()
for (col in colnames(training)) {
  if(sum(!is.na(training[,col]))/15699>0.9){
    fullvars <- c(fullvars, col)
  }
}
```

Create Data with the columns that dont have missing values

```
fulltr <- training[,fullvars]
```

Confirming there is nothing missing.

```
missmap(fulltr)
```



There are too many variables. Will try to weed out the nonimportant variables by fitting a default random forest then using Variable importance function.

Default Random Forest

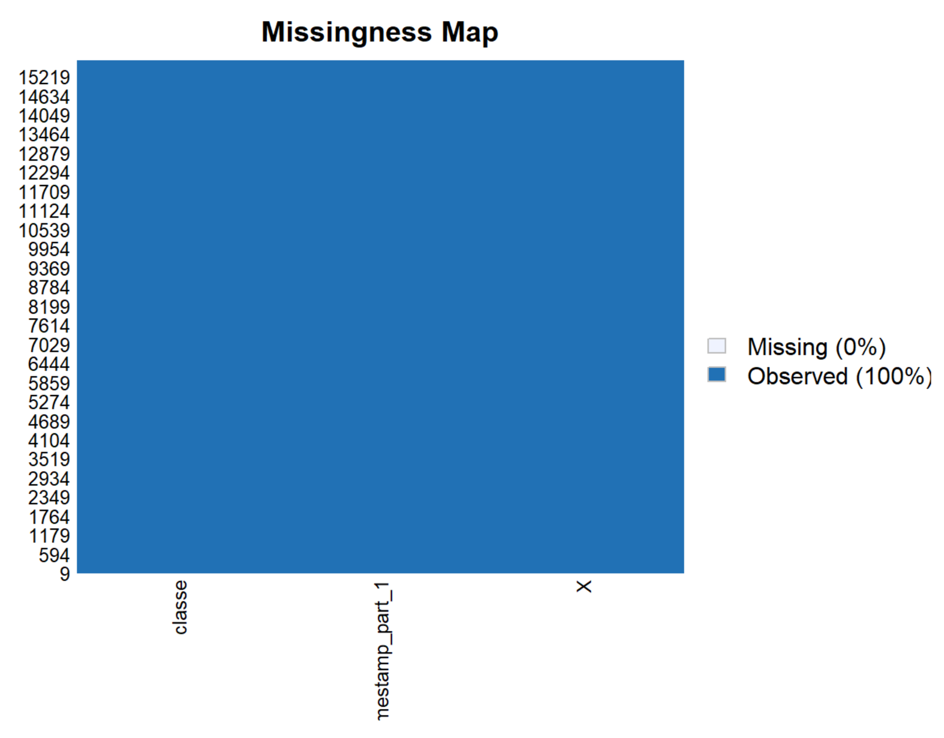
```
rf1 <- randomForest(classe~.,
                    data = fulltr,
                    treesize=1000)
```

Saving the Important Variables

```
impVars <- varImp(rf1) %>% arrange(-Overall) %>%
  filter(Overall >800) %>% row.names()
```

Checking the missingness of the important variables

```
imptraining <- training %>% select(impVars, classe)
missmap(imptraining)
```



Cross-Validation

10-Fold Cross Validation only once to save time.

```
cvcontrol <- trainControl(## 10-fold CV
                          method = "repeatedcv",
                          number = 10,
                          ## repeated ten times
                          repeats = 1)
```

Train Models

Stochastic Gradient Boosting, Random Forest, and Linear Discriminant Analysis

```
set.seed(321)

gbmFit1 <- train(classe ~ .,
                 data = imptraining,
                 method = "gbm",
                 trControl = cvcontrol,
                 verbose = FALSE)

rfFit1 <- train(classe~.,
                data = imptraining,
                method = "rf",
                trControl = cvcontrol)
```

```
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
```

```
ldaFit1 <- train(classe~.,
                data = imptraining,
                method = "lda",
                trControl = cvcontrol)
```

Model Accuracy

Stochastic Gradient Boosting has an accuracy of 99.98% as seen below.

```
gbmFit1
```

```
## Stochastic Gradient Boosting
##
## 15699 samples
##     2 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 14129, 14128, 14130, 14129, 14130, 14128, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.9998089  0.9997583
##   1                  100      0.9997452  0.9996777
##   1                  150      0.9997452  0.9996777
##   2                   50      0.9998089  0.9997583
##   2                  100      0.9998089  0.9997583
##   2                  150      0.9998089  0.9997583
##   3                   50      0.9998089  0.9997583
##   3                  100      0.9998089  0.9997583
##   3                  150      0.9998089  0.9997583
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 50, interaction.depth =
##  1, shrinkage = 0.1 and n.minobsinnode = 10.
```

Random Forest has an accuracy of 99.99% as well as seen below.

```
rfFit1
```

```
## Random Forest
##
## 15699 samples
##     2 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 14129, 14130, 14130, 14127, 14130, 14130, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9999363  0.9999194
##
## Tuning parameter 'mtry' was held constant at a value of 2
```

Linear Discriminant Analysis has an accuracy of 95.94% as seen below.

```
ldaFit1
```

```
## Linear Discriminant Analysis
##
## 15699 samples
##     2 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 14128, 14130, 14128, 14130, 14129, 14130, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.959488  0.9488191
```

Out of Sample Error

Now we can estimate the out of sample error with the testing sample we partitioned before. I will predict with each model and check accuracy.

Out of sample accuracy for Stochastic Gradient Boosting

```
gbmpred1 <- predict(gbmFit1, testtrain)

sum(gbmpred1==testtrain$classe)/length(gbmpred1)
```

```
## [1] 1
```

Out of sample accuracy for Random Forest

```
rfpred1 <- predict(rfFit1, testtrain)

sum(rfpred1==testtrain$classe)/length(rfpred1)
```

```
## [1] 1
```

Out of sample accuracy for Linear Discriminant Analysis

```
ldapred1 <- predict(ldaFit1, testtrain)

sum(ldapred1==testtrain$classe)/length(ldapred1)
```

```
## [1] 0.956798
```

Stochastic Gradient Boost and Random Forest get an estimated 100% accuracy on the out of sample testing set and Linear Discriminant Analysis get 95.67%.

Final Predictions

I will use Stochastic Gradient Boosting for my predictions for the final Test set of 20 samples.

```
finalpred <- predict(gbmFit1, testing)
```

The model predicts they will all be classe A, crossing my fingers.