PracticalMLAssignment

Renison Macwan

15/06/2020

Goal

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Reproducibility

We have set seed to make this project reproducible.

Model

We have selected two different machine learning classification models here: (RandomForest and Recursive Partitioning and Regression Trees) and predictions have been made on the basis of them.

Cross-validating

We have a large training set which is divided into subparts of training and testing. We cross validate our model on the testing subpart of the main training set.

Out-of-sample error

Here the variable to be predicted is and unordered factor. Hence OOSE = (1-Accuracy of model)

Download the datasets and read them

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

train <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
test <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

Load the libraries

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(rpart)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.3
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(MASS)
library(TH.data)
```

```
## Warning: package 'TH.data' was built under R version 3.6.3
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
##
## Attaching package: 'TH.data'
```

```
## The following object is masked from 'package:MASS':
##
##     geyser
```

```
library(ipred)
```

```
## Warning: package 'ipred' was built under R version 3.6.3
```

```
set.seed(13)
```

Remove unuseful parameters

The parameters like ID, name, timestamp etc are not necessary for making a prediction. Hence all columns with headings containing name/timestamp/window are removed.

The new dataframe such formed is called train1. A copy of train1 is made and stored as train2.

```
NoNeed <- grep("name|timestamp|window|X", colnames(train), value=F)
train1 <- train[,-NoNeed]
train2 <- train1
```

Remove predictors with large number of na entries

Here we run a for loop on train1 and find the indices of the columns that have more than 45 % NAs. These columns are removed from the copy dataset train2.

A training and testing partition is made for train2 with 75:25 ratio. The training set is called subTrain and test set is called subTest.

```
for(i in 1:length(train1)) {
        if( sum( is.na( train1[, i] ) ) /nrow(train1) >= 0.45){
             for(j in 1:length(train2)) {
            if( length( grep(names(train1[i]), names(train2)[j]) ) ==1)  {
                train2 <- train2[ , -j]
            }
        }
    }
}

train2 <- train2[,-c(14)]

subsamples <- createDataPartition(y=train2$classe, p=0.75, list=FALSE)
subTrain <- train2[subsamples, ]
subTest <- train2[-subsamples, ]
```

Checking for any predictors with near zero variance

We still want to remove any predictors that might be unnecessary. So we run the nearZeroVar function.

```
train3 <- nearZeroVar(train2, saveMetrics=TRUE)
NZVindice <- which(train3$nzv == "TRUE")
```

No columns in train2 have near zero variance and so we move on.

Decision Tree model

```
DTmodel <- rpart(classe ~ ., data=subTrain, method="class")
predictDT <- predict(DTmodel, subTest, type = "class")
confusionMatrix(predictDT, subTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1239  140   20   38   35
##          B   54  637   80   87   88
##          C   10   90  637  134   79
##          D   48   51   56  492   36
##          E   44   31   62   53  663
##
## Overall Statistics
##
##                Accuracy : 0.748
##                  95% CI : (0.7356, 0.7601)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6804
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8882   0.6712   0.7450   0.6119   0.7358
## Specificity            0.9336   0.9219   0.9227   0.9534   0.9525
## Pos Pred Value         0.8417   0.6734   0.6705   0.7204   0.7773
## Neg Pred Value         0.9545   0.9212   0.9449   0.9261   0.9412
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2527   0.1299   0.1299   0.1003   0.1352
## Detection Prevalence   0.3002   0.1929   0.1937   0.1393   0.1739
## Balanced Accuracy      0.9109   0.7966   0.8339   0.7827   0.8442
```

The accuracy we find here is only 72 percent.

Random Forest model

```
RFmodel <- randomForest(classe ~. , data=subTrain,  type = "class")
predictRF <- predict(RFmodel, subTest, type = "class")
confusionMatrix(predictRF, subTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    2    0    0    0
##          B    0  943    7    0    0
##          C    0    4  846    5    0
##          D    0    0    2  798    1
##          E    0    0    0    1  900
##
## Overall Statistics
##
##                Accuracy : 0.9955
##                  95% CI : (0.9932, 0.9972)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9943
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9937   0.9895   0.9925   0.9989
## Specificity            0.9994   0.9982   0.9978   0.9993   0.9998
## Pos Pred Value         0.9986   0.9926   0.9895   0.9963   0.9989
## Neg Pred Value         1.0000   0.9985   0.9978   0.9985   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1923   0.1725   0.1627   0.1835
## Detection Prevalence   0.2849   0.1937   0.1743   0.1633   0.1837
## Balanced Accuracy      0.9997   0.9960   0.9936   0.9959   0.9993
```

A high accuracy of 99.45 % is found using the random forest model.

Bagging model

```
Bmodel <- bagging(classe ~. , data=subTrain,  coob=TRUE)
predictB <- predict(Bmodel, subTest, type = "class")
confusionMatrix(predictB, subTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1390    5    0    0    0
##          B    4  934    9    0    3
##          C    1    9  839   10    6
##          D    0    1    7  793    3
##          E    0    0    0    1  889
##
## Overall Statistics
##
##                Accuracy : 0.988
##                  95% CI : (0.9845, 0.9908)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9848
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9842   0.9813   0.9863   0.9867
## Specificity            0.9986   0.9960   0.9936   0.9973   0.9998
## Pos Pred Value         0.9964   0.9832   0.9699   0.9863   0.9989
## Neg Pred Value         0.9986   0.9962   0.9960   0.9973   0.9970
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2834   0.1905   0.1711   0.1617   0.1813
## Detection Prevalence   0.2845   0.1937   0.1764   0.1639   0.1815
## Balanced Accuracy      0.9975   0.9901   0.9874   0.9918   0.9932
```

The bagging model gives an accuracy of 98.8%.

Making predictions on the test dataset

We select the model with the highest accuracy, i.e. the RandomForest model for making predictions on the test set.

```
predictions <- predict(RFmodel, test, type = "class")
predictions
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```