Practical Machine Learning Course Project

Hariharan

10/14/2020

Overview

This project aims to fit a machine learning model to predict the manner in which excersice is done We would be importing the dataset and cleaning it up and then use popular classification algortihms to predict the manner in which the excercise is done After applying these algorithms we would select the best algorithm available and apply is to the validate set given

Importing Data Sets

Brief on the data set

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, our goal is 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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

```
train_data_cleaned_raw <- read.csv('pml-training.csv' , header=T)
validate_data_raw <- read.csv('pml-testing.csv' , header=T)
```

Observing dimensions of Data sets

```
dim(train_data_cleaned_raw)
```

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

```
dim(validate_data_raw)
```

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

We can observe we have 19622 observations and 160 columns/variables present in the training set

Loading required packages

```
library(caret)
```

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

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

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

```
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.6.3
```

```
library(rattle)
```

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

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

Cleaning the Training Data

Removing columns with Missing values

```
train_data_cleaned <- train_data_cleaned_raw[, colSums(is.na(train_data_cleaned_raw))==0]
dim(train_data_cleaned)
```

```
## [1] 19622    93
```

The First 7 columns in the data Set is for identifying the person performing the exercise and is irrelevant to what we are trying to predict Therefore we remove them

```
train_data_cleaned <- train_data_cleaned[ , -c(1:7)]
dim(train_data_cleaned)
```

```
## [1] 19622    86
```

Now we will remove the variables that are near zero variance

```
NZV_cleaning_data <- nearZeroVar(train_data_cleaned)
train_data_cleaned <- train_data_cleaned[, -NZV_cleaning_data]
dim(train_data_cleaned)
```

```
## [1] 19622    53
```

We Split the Training data to training and testing data for prediction using various models

The training data will be used for trianing the models and the test set to verify, The test data provided for this project is used as a validate data set on the the final model which will be selected based on perfomance

```
set.seed(9999)
splitter <- createDataPartition(train_data_cleaned$classe, p = 0.8, list = FALSE)
train_data_cleaned <- train_data_cleaned[splitter, ]
test_data_cleaned <- train_data_cleaned[-splitter, ]
dim(train_data_cleaned)
```

```
## [1] 15699    53
```

Model Building

In this section we will go thru three classification algorithms - Rndom Forest - Decision Trees - Generalized Boosted Regresion Models

For Each algorithm we will fit the training dat set and use the test set to predict and porduce confusion matrices and look into prediction accuracy

Modellin Using Rndom Forest

```
set.seed(9999)
trainer <- trainControl(method='cv' , number = 3 , verboseIter = FALSE)
rf_model <- train(classe ~ . , data = train_data_cleaned , method = 'rf' , trControl = trainer )
rf_model$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.56%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4464    0    0    0    0 0.000000000
## B   16 3016    6    0    0 0.007241606
## C    0   16 2720    2    0 0.006574142
## D    0    0   39 2533    1 0.015546055
## E    0    0    3    5 2878 0.002772003
```

```
random_forest_predict <- predict(rf_model, newdata=test_data_cleaned)
confuse_rf <- confusionMatrix(random_forest_predict, test_data_cleaned$classe)
confuse_rf
```

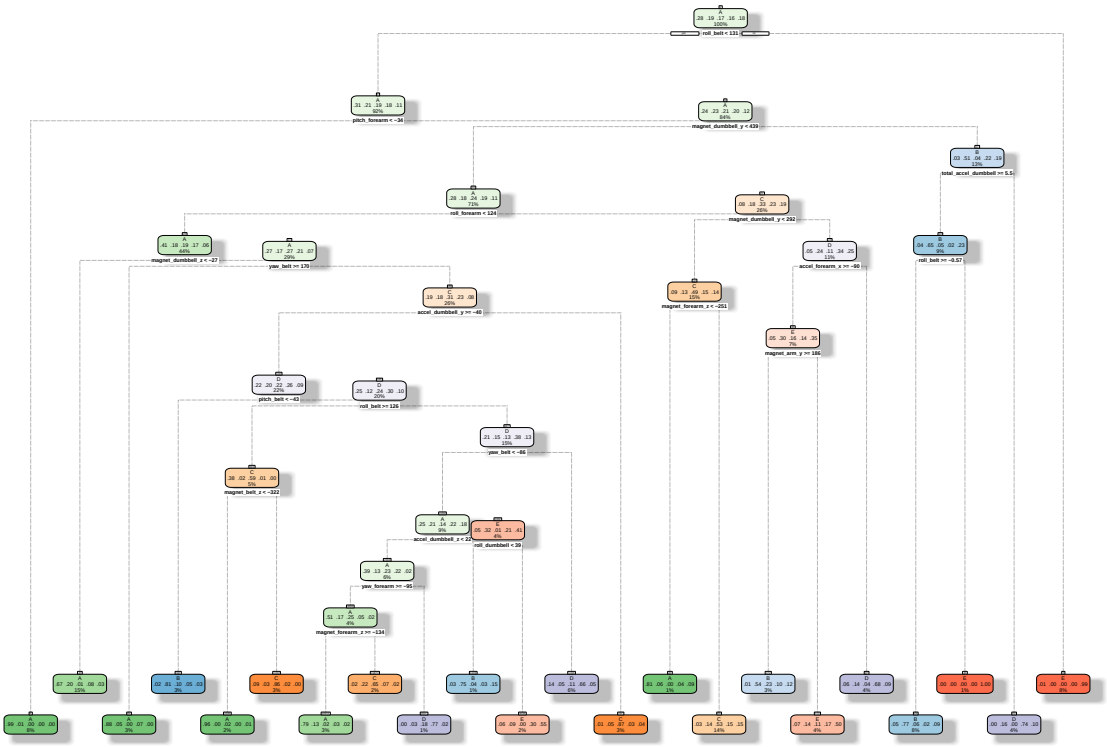
Predicting using Rndom Forest

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 899   0   0   0   0
##          B   0 591   0   0   0
##          C   0   0 554   0   0
##          D   0   0   0 518   0
##          E   0   0   0   0 567
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9988, 1)
##     No Information Rate : 0.2873
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2873   0.1889   0.1771   0.1655   0.1812
## Detection Rate         0.2873   0.1889   0.1771   0.1655   0.1812
## Detection Prevalence   0.2873   0.1889   0.1771   0.1655   0.1812
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Modellin Using Decision Trees

```
set.seed(999)
decision_tree_model <- rpart(classe ~ ., data=train_data_cleaned, method="class")
fancyRpartPlot(decision_tree_model)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```




```
decision_tree_predict <- predict(decision_tree_model, test_data_cleaned, type = "class")
confuse_decision <- confusionMatrix(decision_tree_predict, test_data_cleaned$classe)
confuse_decision
```

Predicting using Decision Trees

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 814 110  10  45  15
##          B  12 326  45  24  40
##          C  18  87 453  79  66
##          D  25  46  30 325  30
##          E  30  22  16  45 416
##
## Overall Statistics
##
##                Accuracy : 0.7459
##                  95% CI : (0.7303, 0.7611)
##     No Information Rate : 0.2873
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.6772
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9055   0.5516   0.8177   0.6274   0.7337
## Specificity            0.9193   0.9523   0.9029   0.9498   0.9559
## Pos Pred Value         0.8189   0.7293   0.6444   0.7127   0.7864
## Neg Pred Value         0.9602   0.9012   0.9584   0.9278   0.9419
## Prevalence             0.2873   0.1889   0.1771   0.1655   0.1812
## Detection Rate         0.2601   0.1042   0.1448   0.1039   0.1329
## Detection Prevalence   0.3177   0.1429   0.2247   0.1457   0.1691
## Balanced Accuracy      0.9124   0.7520   0.8603   0.7886   0.8448
```

Modellin Using Generalized Boosted Regression Models

```
set.seed(9999)
trainer_gbm <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
gbm_model  <- train(classe ~ ., data=train_data_cleaned, method = "gbm", trControl = trainer_gbm, verbos
gbm_model$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

```
print(gbm_model)
```

```
## Stochastic Gradient Boosting
##
## 15699 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 12559, 12558, 12559, 12560, 12560
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7557176  0.6902222
##   1                  100      0.8222810  0.7750264
##   1                  150      0.8540674  0.8152761
##   2                   50      0.8563605  0.8180011
##   2                  100      0.9078288  0.8833640
##   2                  150      0.9320347  0.9139915
##   3                   50      0.8975096  0.8702209
##   3                  100      0.9398691  0.9239122
##   3                  150      0.9615900  0.9514007
##
```

```
## 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 = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
gbm_predict <- predict(gbm_model, newdata=test_data_cleaned)
confuse_gbm <- confusionMatrix(gbm_predict, test_data_cleaned$classe)
confuse_gbm
```

Predictig using Generalized Boosted Regression Models

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 893  11   0   1   0
##          B   4 565  11   2   4
##          C   1  14 539  14   3
##          D   1   1   4 496   7
##          E   0   0   0   5 553
##
## Overall Statistics
##
##                Accuracy : 0.9735
##                  95% CI : (0.9672, 0.9788)
##     No Information Rate : 0.2873
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9664
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9933   0.9560   0.9729   0.9575   0.9753
## Specificity            0.9946   0.9917   0.9876   0.9950   0.9980
## Pos Pred Value         0.9867   0.9642   0.9440   0.9745   0.9910
## Neg Pred Value         0.9973   0.9898   0.9941   0.9916   0.9946
## Prevalence             0.2873   0.1889   0.1771   0.1655   0.1812
## Detection Rate         0.2854   0.1806   0.1723   0.1585   0.1767
## Detection Prevalence   0.2892   0.1873   0.1825   0.1627   0.1783
## Balanced Accuracy      0.9940   0.9739   0.9802   0.9763   0.9867
```

Predicting Output for the provided Test Set(Used as Validate set in this)

Applying same prepossessing to the validate set to get clean dat

```
validate_data <- validate_data_raw[, colSums(is.na(validate_data_raw))==0]
validate_data <- validate_data[ , -c(1:7)]
dim(validate_data)
```

```
## [1] 20 53
```

Applying best Model available to make preditions

The accurasy metrics of the implemented algorithms are as follows - Radom forest Classifier -> 100 % (Overfit Model) - Decesion Tree Classifier -> 74.59 % - Genearalized Boosted Regression Models -> 97.35 % Since Random forest model has the best accuracy among the three models used in this project, we will apply it to the validation set

```
validate <- predict(rf_model, newdata=validate_data)
validate
```

```
##  [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
```