Assignment Description

Esra Guc

07/02/2021

First I loaded libraries that I will need for this assignment

```
library(caret)
```

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

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

```
library(lattice)
library(ggplot2)
library(rpart)
library(rpart.plot)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(randomForest)
```

```
## 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(rattle)
```

```
## 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.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

Then I set the seed and installed the data from my computer

```
setwd("/Users/eguc/Desktop")
set.seed(123)
```

```
train_dat<- read.csv("pml-training.csv")[,-1]
test_dat<- read.csv("pml-testing.csv")[,-1]
dim(train_dat)
```

```
## [1] 19622   159
```

```
dim(test_dat)
```

```
## [1]  20 159
```

There are 159 variable in this data set however some variables have a lot of NA, some has almost 0 variation also first 5 variables wont be used in training so these are removed

```
Zerovar<- nearZeroVar(train_dat)
training<- train_dat[,-Zerovar]
testing<- test_dat[,-Zerovar]
NAVal <- sapply(training, function(x) mean(is.na(x))) > 0.95
training <- training[, NAVal == "FALSE"]
testing <- testing[, NAVal== "FALSE"]
training <- training[,-c(1:5)]
testing <- testing[,-c(1:5)]
```

I checked the number of variables again

```
dim(training)
```

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

```
dim(testing)
```

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

variable numbers are decreased to 53

There are few variables that are highly correlated (negative correlations is dark red(-1) and positive correlation is dark blue(1)) therefore PCA analysis is not necessary.

Then I start preparing the training and testing data to test models. I will use Decision Tree and Random Forest models. I created two partitions (70% and 30%) from training data.

```
inTraining<- createDataPartition(y=training$classe, p=0.7, list=FALSE)
train_final<- training[inTraining, ]
test_final<- training[-inTraining, ]
```

I first tested Decision Tree Modeling. I classified the data argument in as.factor so data and reference factors will be in the same number of levels.

```
mod_DT <- train(classe ~ .,data=train_final, method = "rpart")
predict_train_DT <- predict(mod_DT, train_final)
ConfM_train_DT <- confusionMatrix(predict_train_DT,as.factor(train_final$classe))
predict_test_DT <- predict(mod_DT, test_final)
ConfM_test_DT <- confusionMatrix(predict_test_DT,as.factor(test_final$classe))
print(ConfM_test_DT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530  464  469  440  144
##          B   28  397   30  169  145
```

```
##          C  114  278  527  355  306
##          D    0    0    0    0    0
##          E    2    0    0    0  487
##
## Overall Statistics
##
##                Accuracy : 0.4997
##                  95% CI : (0.4869, 0.5126)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3464
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140  0.34855  0.51365   0.0000  0.45009
## Specificity            0.6398  0.92162  0.78329   1.0000  0.99958
## Pos Pred Value         0.5021  0.51625  0.33354      NaN  0.99591
## Neg Pred Value         0.9493  0.85496  0.88409   0.8362  0.88973
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2600  0.06746  0.08955   0.0000  0.08275
## Detection Prevalence   0.5178  0.13067  0.26848   0.0000  0.08309
## Balanced Accuracy      0.7769  0.63508  0.64847   0.5000  0.72484
```

Decision Tree Modeling gives an 0.4997 accuracy. Then I tried Random Forest Modeling. However Random Forest data processing takes very long, I got this tip from Coursera Forum, data needs to be tuned using tuneGrid

```
mod_RF <- train(classe ~ .,data=train_final, method = "rf", trControl=trainControl(method="none"), tune
predict_train_RF <- predict(mod_RF, train_final)
ConfM_train_RF <- confusionMatrix(predict_train_RF,as.factor(train_final$classe))
predict_test_RF <- predict(mod_RF, test_final)
ConfM_test_RF <- confusionMatrix(predict_test_RF,as.factor(test_final$classe))
print(ConfM_test_RF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    3    0    0    0
##          B    0 1131    5    0    0
##          C    0    5 1021   12    4
##          D    0    0    0  952    4
##          E    0    0    0    0 1074
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9921, 0.9961)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.9929
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9930   0.9951   0.9876   0.9926
## Specificity            0.9993   0.9989   0.9957   0.9992   1.0000
## Pos Pred Value         0.9982   0.9956   0.9798   0.9958   1.0000
## Neg Pred Value         1.0000   0.9983   0.9990   0.9976   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1922   0.1735   0.1618   0.1825
## Detection Prevalence   0.2850   0.1930   0.1771   0.1624   0.1825
## Balanced Accuracy      0.9996   0.9960   0.9954   0.9934   0.9963
```

Random Forest model predicts with higher accuracy overall 0.9944 compared to Decision Tree Modeling Accuracy which was 0.4997.

Because Random Forest gives the best prediction,there is no need for cross-validation and define out of sample error using a seperate test set.

Then I used random forest model to predict 20 different test cases.

```
print(predict(mod_RF, newdata=testing))
```

```
##  [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
```