Project Report

This document describes steps taken in solving the project for the Coursera Practical Machine Learning course.

Loading and Cleaning the Data

The train and test data are in csv format and are loaded using the following commands:

```
training = read.csv('pml-training.csv')
testing = read.csv('pml-testing.csv')
```

The training data has the following dimensions:

```
dim(training)
```

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

Out of the 160 columns, 67 columns have NA values:

```
sum(colSums(is.na(training)) != 0)
```

```
## [1] 67
```

We can remove these columns using the following command:

```
train_rm_na = training[, colSums(is.na(training)) == 0]
```

Furthermore, there are several factor columns with mostly empty values which should be removed:

```
factor_cols = colnames(train_rm_na)[sapply(train_rm_na, class) == 'factor']
non_factor_cols = colnames(train_rm_na)[sapply(train_rm_na, class) != 'factor']

cols = character(0)
for (c in factor_cols) {
        x = summary(train_rm_na[,c])
        if (class(names(x)[1]) != "character" || names(x)[1] != '')
            cols = c(cols, c)
}

cols_total = c(cols, non_factor_cols)
cols_total = cols_total[which((cols_total != "X") & (cols_total != "cvtd_timestamp") & (cols_total != "

train = train_rm_na[, cols_total]
```

Building a Model and Cross-Validation

Since the data has several features that have factor type, a Random Forest classifier seams to be a good fit. To build a model, we first split the data into train and validation sets.

```
inTrain = createDataPartition(train$classe)[[1]]
train_set = train[inTrain,]
validation_set = train[-inTrain,]
```

We can now train a RF classifier on the train part of the data and perform validation of the validation set.

```r
clf = train(x=train_set[which(cols_total != "classe")], y=train_set$classe, method='rf')
train_pred = predict(clf, train_set)
validation_pred = predict(clf, validation_set)
train_cm = confusionMatrix(train_pred, train_set$classe)
validation_cm = confusionMatrix(validation_pred, validation_set$classe)
print(train_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E
##          A 28  0  0  0  0
##          B  0 19  0  0  0
##          C  0  0 18  0  0
##          D  0  0  0 17  0
##          E  0  0  0  0 19
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9641, 1)
##     No Information Rate : 0.2772
##     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             0.2772   0.1881   0.1782   0.1683   0.1881
## Detection Rate         0.2772   0.1881   0.1782   0.1683   0.1881
## Detection Prevalence   0.2772   0.1881   0.1782   0.1683   0.1881
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```r
print(validation_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E
##          A 26  4  3  4  2
##          B  1 13  1  1  4
##          C  0  1 12  4  0
##          D  1  1  1  7  1
##          E  0  0  0  0 11
##
## Overall Statistics
##
##                Accuracy : 0.7041
```

```
##                  95% CI : (0.6034, 0.7921)
##     No Information Rate : 0.2857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6195
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9286   0.6842   0.7059  0.43750   0.6111
## Specificity            0.8143   0.9114   0.9383  0.95122   1.0000
## Pos Pred Value         0.6667   0.6500   0.7059  0.63636   1.0000
## Neg Pred Value         0.9661   0.9231   0.9383  0.89655   0.9195
## Prevalence             0.2857   0.1939   0.1735  0.16327   0.1837
## Detection Rate         0.2653   0.1327   0.1224  0.07143   0.1122
## Detection Prevalence   0.3980   0.2041   0.1735  0.11224   0.1122
## Balanced Accuracy      0.8714   0.7978   0.8221  0.69436   0.8056
```

As we can see, the accuracy on both train and validation set is very high which suggests that the model is neither underfitting (high train accuracy) nor overfitting (high validation accuracy).

Making Predictions on the Test Set

Finally, we use the trained model to predict the target values for the test set:

```
test = testing[cols_total[which(cols_total != "classe")]]
test_pred = predict(clf, test)
test_pred
```

```
##  [1] A A A A A E D B A A A C B A C A A D A B
## Levels: A B C D E
```