Machine Learning Final Project

Kyle Ward

January 4, 2019

Executive Summary

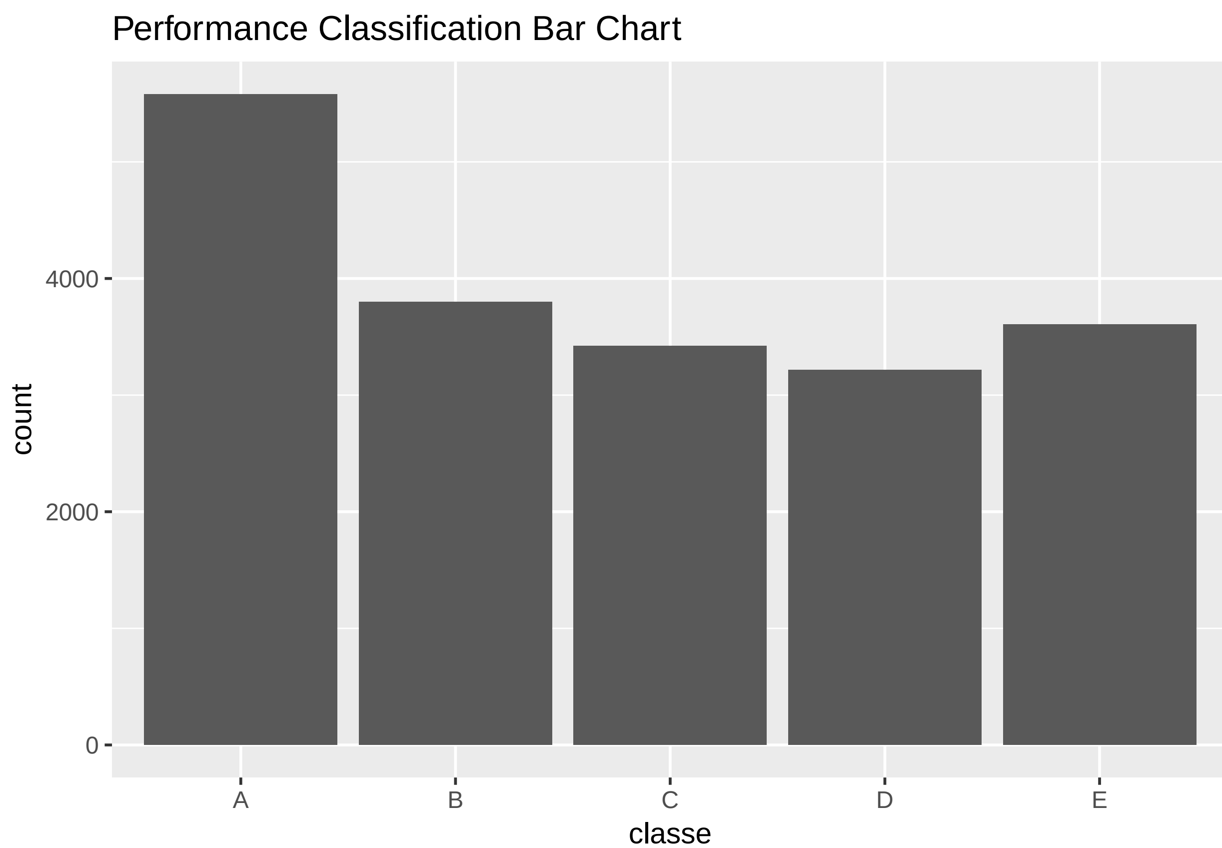
The purpose of this analysis is to estimate a prediction model to determine how well a weight lifting exercise is performed based on performance data collected from the use of various sensors. The performance of the weight lifting exercise is classified according to five different classifications with class A representing performing the exercise exactly according to the specified instructions. Data for this analysis was obtained from a Human Activity Recognition dataset licensed under the Creative Commons license (CC BY-SA) (Read more: http://groupwar%3Ce.les.inf.puc-rio.br/har#dataset#ixzz5bfIWAJVO).

Two competing machine learning models will be evaluated and the model that predicts the correct classification with greater accuracy will be selected.

Exploratory Data Analysis

Weight Lifting Exercise Performance Classification Definitions (Classe):

- A: Exactly according to specification
- B: Elbows in front
- C: Dumbell lifted halfway
- D: Dumbell lowered halfway
- E: Hips in front



Classification Frequency Percentages (Training Data)

```
## classe
##         A         B         C         D         E 
## 0.2843747 0.1935073 0.1743961 0.1638977 0.1838243
```

Pre-Processing Data

Removing Zero Covariates

Displaying first 10 non zero variables

```
##                         freqRatio percentUnique zeroVar  nzv
## new_window               47.33005    0.01019264   FALSE TRUE
## kurtosis_roll_belt     1921.60000    2.02323922   FALSE TRUE
## kurtosis_picth_belt     600.50000    1.61553358   FALSE TRUE
## kurtosis_yaw_belt        47.33005    0.01019264   FALSE TRUE
## skewness_roll_belt     2135.11111    2.01304658   FALSE TRUE
## skewness_roll_belt.1    600.50000    1.72255631   FALSE TRUE
## skewness_yaw_belt        47.33005    0.01019264   FALSE TRUE
## max_yaw_belt            640.53333    0.34654979   FALSE TRUE
## min_yaw_belt            640.53333    0.34654979   FALSE TRUE
## amplitude_yaw_belt       50.04167    0.02038528   FALSE TRUE
```

Tidying Data Set

Removed variables where vast majority of values were NA's and removed first 6 columns used primary as identifiers.

The remaining pre-processed data sets have the following dimensions:

```
dim(train)
```

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

```
dim(test)
```

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

Splitting Data

```
inTrain <- createDataPartition(y=train$classe,
                               p=0.8, list=FALSE)
training <- train[inTrain,]
validation <- train[-inTrain,]
```

Machine Learning Model Formulation

Model 1: Random Forest Cross Validation Method

Fitting a Random Forest Model based on training data

```
##
## 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: 0.62%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4461    3    0    0    0 0.000672043
## B   18 3013    7    0    0 0.008229098
## C    0   17 2716    5    0 0.008035062
## D    0    0   38 2533    2 0.015546055
## E    0    0    2    5 2879 0.002425502
```

Evaluating random forest cv model performance based on validation dataset

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1115    0    0    0    1
##          B    4  755    0    0    0
##          C    0    4  679    1    0
##          D    0    0    8  634    1
##          E    0    0    0    0  721
##
## Overall Statistics
##
##                Accuracy : 0.9952
##                  95% CI : (0.9924, 0.9971)
##     No Information Rate : 0.2852
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9939
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9947   0.9884   0.9984   0.9972
## Specificity            0.9996   0.9987   0.9985   0.9973   1.0000
## Pos Pred Value         0.9991   0.9947   0.9927   0.9860   1.0000
## Neg Pred Value         0.9986   0.9987   0.9975   0.9997   0.9994
## Prevalence             0.2852   0.1935   0.1751   0.1619   0.1843
## Detection Rate         0.2842   0.1925   0.1731   0.1616   0.1838
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9980   0.9967   0.9934   0.9978   0.9986
```

Model 2: Gradient Boosting Method

Fitting a Gradient Boosting Model with trees based on training data

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 41 had non-zero influence.
```

Evaluating Gradient Boosting model performance based on validation dataset

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1102    9    1    4    0
##          B   28  711   17    2    1
##          C    0   16  655   12    1
##          D    0    4   15  621    3
##          E    0    7    3   10  701
##
## Overall Statistics
##
##                Accuracy : 0.9661
##                  95% CI : (0.9599, 0.9715)
##     No Information Rate : 0.288
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9571
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9752   0.9518   0.9479   0.9569   0.9929
## Specificity            0.9950   0.9849   0.9910   0.9933   0.9938
## Pos Pred Value         0.9875   0.9368   0.9576   0.9658   0.9723
## Neg Pred Value         0.9900   0.9886   0.9889   0.9915   0.9984
## Prevalence             0.2880   0.1904   0.1761   0.1654   0.1800
## Detection Rate         0.2809   0.1812   0.1670   0.1583   0.1787
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9851   0.9683   0.9695   0.9751   0.9934
```

Conclusion

Random Forest Model yields creater accuracy compared with the Gradient Booting method (99%>96%)

Selecting Random Forest Model to use for prediction submission using test dataset

```
FinalPred<-predict(RFmodeltrain,newdata = test)
FinalPreddf<-data.frame(test_ID=test$problem_id,PredictClass=FinalPred)
FinalPreddf
```

```
##    test_ID PredictClass
## 1        1            B
## 2        2            A
## 3        3            B
## 4        4            A
## 5        5            A
## 6        6            E
## 7        7            D
## 8        8            B
## 9        9            A
## 10      10            A
## 11      11            B
## 12      12            C
## 13      13            B
## 14      14            A
## 15      15            E
## 16      16            E
## 17      17            A
## 18      18            B
## 19      19            B
## 20      20            B
```

Appendix

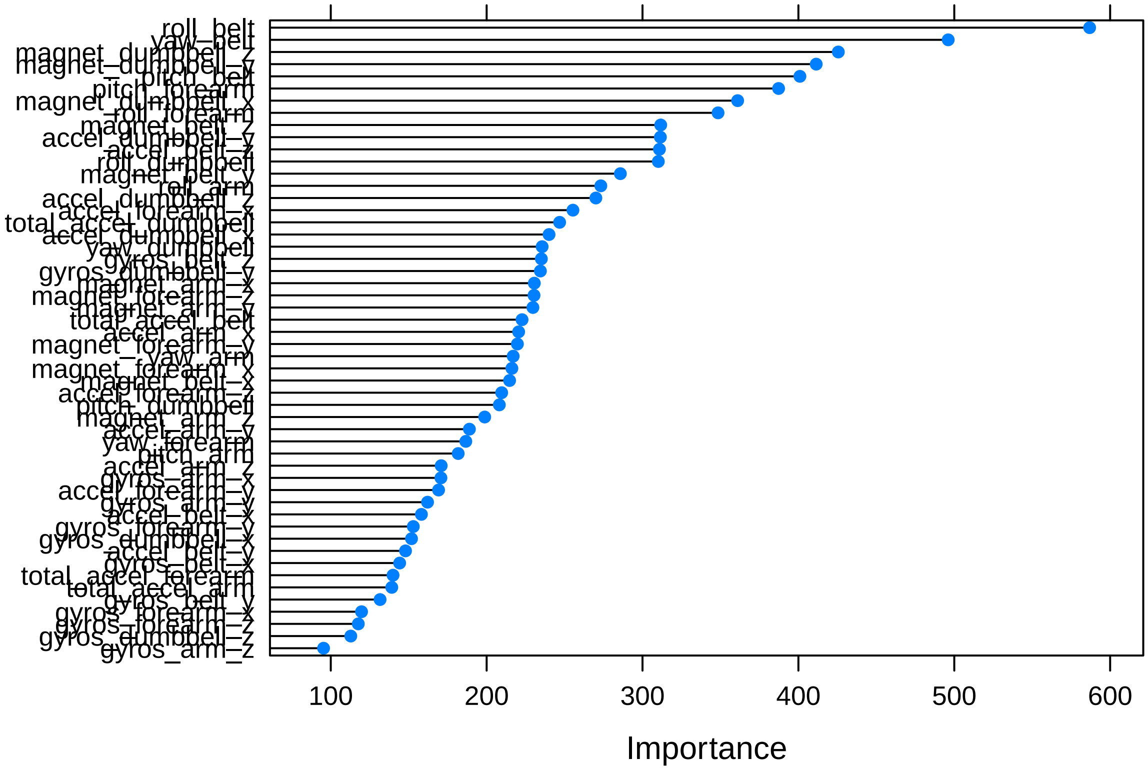
Additional Data Exploration

Most relevant 20 variables on the model

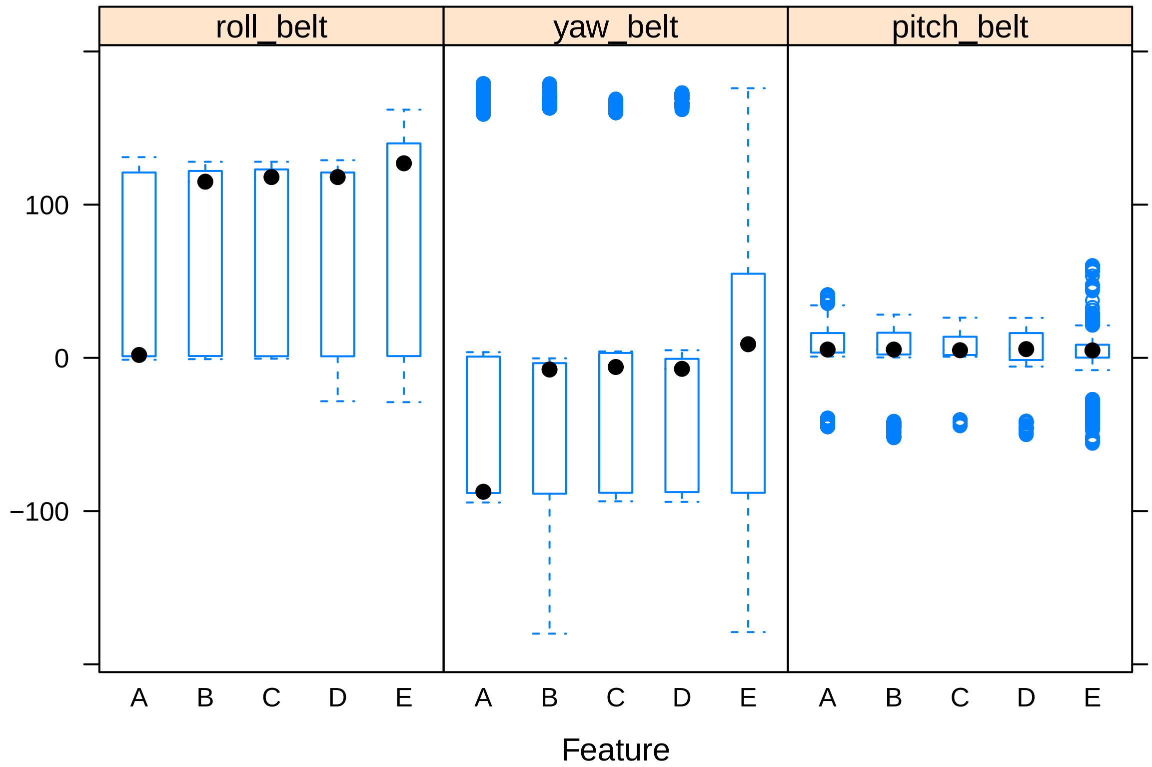
```
importance <- varImp(RFmodeltrain, scale=FALSE)
print(importance)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt              586.9
## yaw_belt               496.1
## magnet_dumbbell_z      425.6
## magnet_dumbbell_y      411.4
## pitch_belt             401.0
## pitch_forearm          387.3
## magnet_dumbbell_x      361.0
## roll_forearm           348.5
## magnet_belt_z          311.7
## accel_dumbbell_y       311.5
## accel_belt_z           310.9
## roll_dumbbell          310.1
## magnet_belt_y          285.8
## roll_arm               273.3
## accel_dumbbell_z       270.1
## accel_forearm_x        255.4
## total_accel_dumbbell   246.8
## accel_dumbbell_x       240.0
## yaw_dumbbell           235.6
## gyros_belt_z           235.1
```

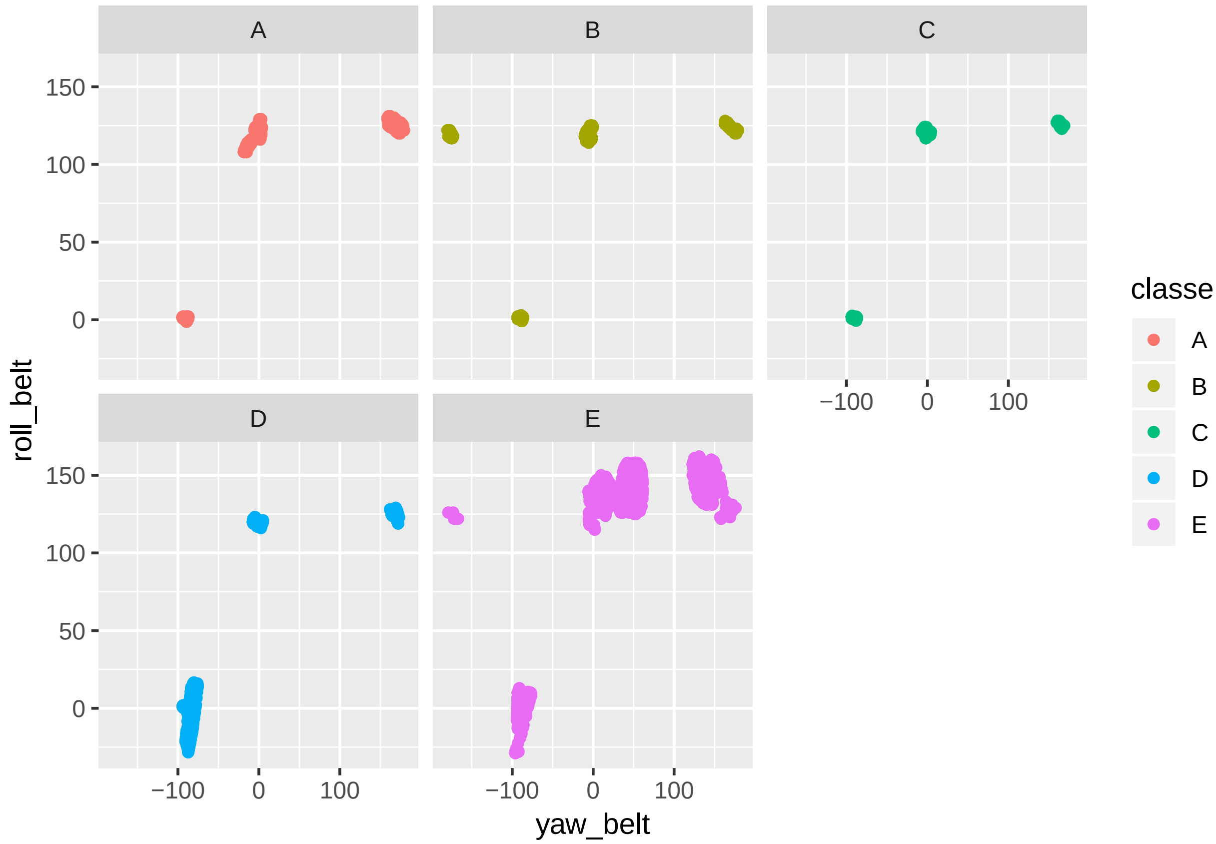
```
plot(importance)
```



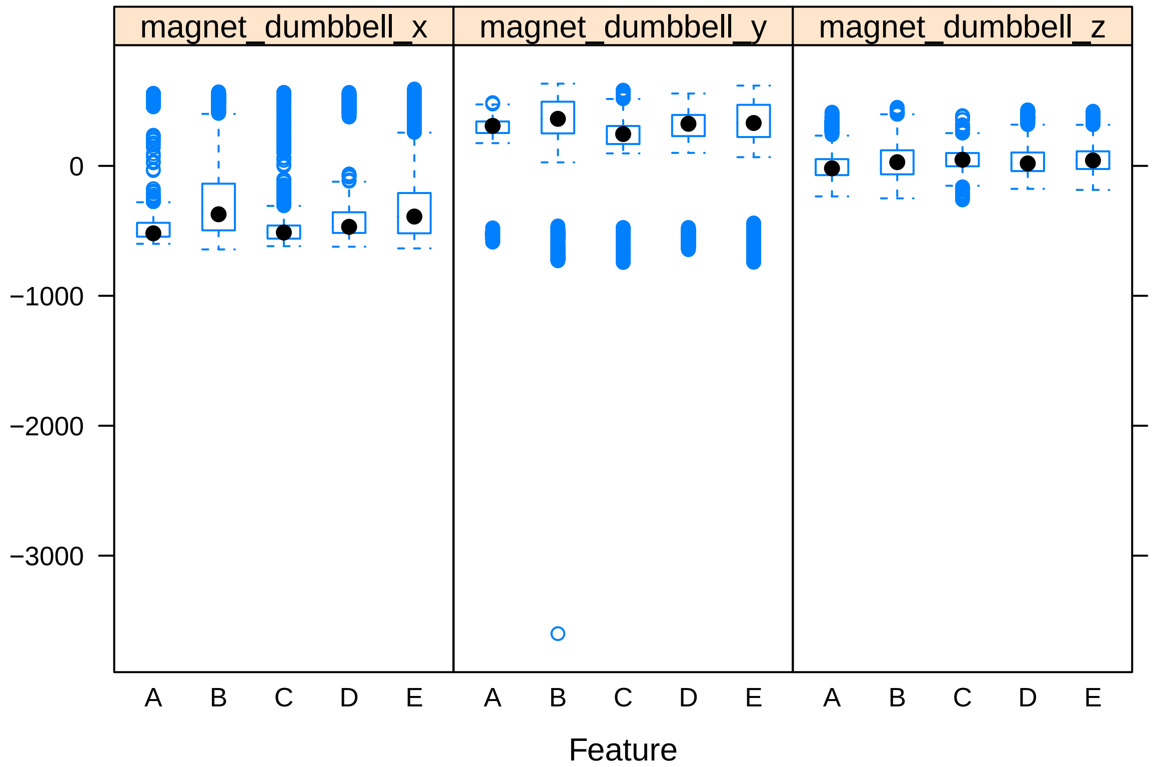
Feature Plot - Belt Euler Angles



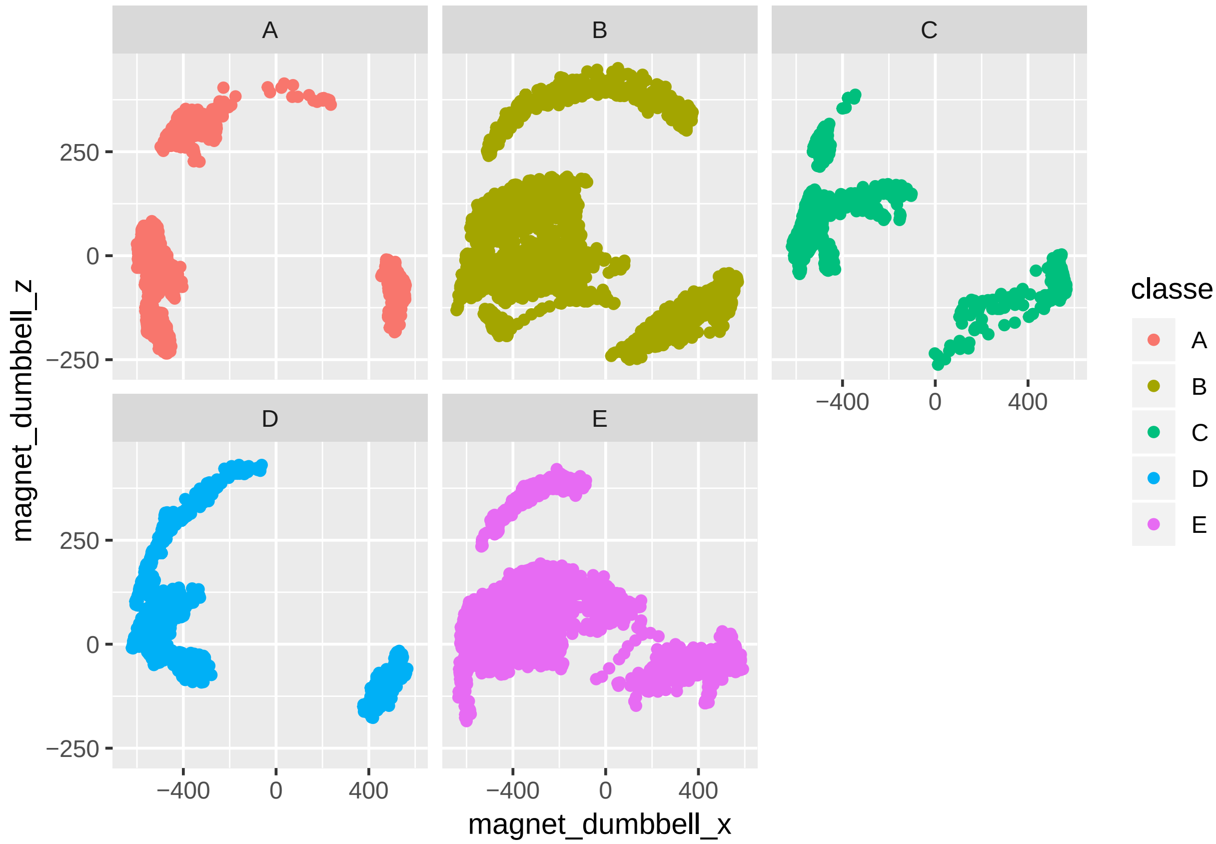
Belt Euler Angle Scatterplot by Classe



Feature BoxPlot - Dumbbell magnometer



Dumbbell Magnometer Scatterplot by Classe


A
B
C
D
E
250
0
−250
−400
0
400
magnet_dumbbell_z
magnet_dumbbell_x
classe