JH Practical Machine Learning Assignment

Ksenia Sukmanskaya

Executive Summary

This project is aimed to show how different ML models work for solving classification task to predict how well a weight lifting activity was performed. Namley we're investigating the quality of the Unilateral Dumbbell Biceps Curl.

Possible outcomes: - **Class A:** exactly according to the specification - **Class B:** throwing the elbows to the front - **Class C:** lifting the dumbbell only halfway - **Class D:** lowering the dumbbell only halfway - **Class E:** throwing the hips to the front

Exploratory Analysis

Load data

Look up outcome distribution:

```
table(training$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

1. Remove unnecessary variables:

```
training <- training[,-which(names(training) %in%
                                    c('user_name',
                                      'raw_timestamp_part_1',
                                      'raw_timestamp_part_2',
                                      'cvtd_timestamp',
                                      'new_window',
                                      'num_window')
                                  )
                     ]
# # take subset of features:
# # avg, var, stdev, max, min, amplitude, kurtosis, skewness
# features <- names(training)[startsWith(names(training), c('avg',
#                                                            'var',
#                                                            'stdev',
#                                                            'max',
#                                                            'min',
#                                                            'amplitude',
#                                                            'kurtosis',
#                                                            'skewness')) == T]
```

```
#
# names(training)[grep('^avg|var', names(training))]
```

2. Check how many NAs we have for ech variable:

```
vars_to_drop <- c()
for (f in names(training)){
  na_cnt <- sum(is.na(training[,f]))
  if (na_cnt > 0) {
    vars_to_drop <- c(vars_to_drop, f)
    print(paste0(f, '(NA cnt) : ', na_cnt, ';   % of NAs: ', round(mean(is.na(training[,f])),2)))
  }
}
```

```
## [1] "kurtosis_roll_belt(NA cnt) : 19226;   % of NAs: 0.98"
## [1] "kurtosis_picth_belt(NA cnt) : 19248;   % of NAs: 0.98"
## [1] "kurtosis_yaw_belt(NA cnt) : 19622;   % of NAs: 1"
## [1] "skewness_roll_belt(NA cnt) : 19225;   % of NAs: 0.98"
## [1] "skewness_roll_belt.1(NA cnt) : 19248;   % of NAs: 0.98"
## [1] "skewness_yaw_belt(NA cnt) : 19622;   % of NAs: 1"
## [1] "max_roll_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "max_picth_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "max_yaw_belt(NA cnt) : 19226;   % of NAs: 0.98"
## [1] "min_roll_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "min_pitch_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "min_yaw_belt(NA cnt) : 19226;   % of NAs: 0.98"
## [1] "amplitude_roll_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "amplitude_pitch_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "amplitude_yaw_belt(NA cnt) : 19226;   % of NAs: 0.98"
## [1] "var_total_accel_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "avg_roll_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "stddev_roll_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "var_roll_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "avg_pitch_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "stddev_pitch_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "var_pitch_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "avg_yaw_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "stddev_yaw_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "var_yaw_belt(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "var_accel_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "avg_roll_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "stddev_roll_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "var_roll_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "avg_pitch_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "stddev_pitch_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "var_pitch_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "avg_yaw_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "stddev_yaw_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "var_yaw_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "kurtosis_roll_arm(NA cnt) : 19294;   % of NAs: 0.98"
## [1] "kurtosis_picth_arm(NA cnt) : 19296;   % of NAs: 0.98"
## [1] "kurtosis_yaw_arm(NA cnt) : 19227;   % of NAs: 0.98"
## [1] "skewness_roll_arm(NA cnt) : 19293;   % of NAs: 0.98"
## [1] "skewness_pitch_arm(NA cnt) : 19296;   % of NAs: 0.98"
## [1] "skewness_yaw_arm(NA cnt) : 19227;   % of NAs: 0.98"
## [1] "max_roll_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "max_picth_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "max_yaw_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "min_roll_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "min_pitch_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "min_yaw_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "amplitude_roll_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "amplitude_pitch_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "amplitude_yaw_arm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "kurtosis_roll_dumbbell(NA cnt) : 19221;   % of NAs: 0.98"
## [1] "kurtosis_picth_dumbbell(NA cnt) : 19218;   % of NAs: 0.98"
## [1] "kurtosis_yaw_dumbbell(NA cnt) : 19622;   % of NAs: 1"
## [1] "skewness_roll_dumbbell(NA cnt) : 19220;   % of NAs: 0.98"
## [1] "skewness_pitch_dumbbell(NA cnt) : 19217;   % of NAs: 0.98"
## [1] "skewness_yaw_dumbbell(NA cnt) : 19622;   % of NAs: 1"
## [1] "max_roll_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "max_picth_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "max_yaw_dumbbell(NA cnt) : 19221;   % of NAs: 0.98"
## [1] "min_roll_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "min_pitch_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "min_yaw_dumbbell(NA cnt) : 19221;   % of NAs: 0.98"
## [1] "amplitude_roll_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "amplitude_pitch_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "amplitude_yaw_dumbbell(NA cnt) : 19221;   % of NAs: 0.98"
## [1] "var_accel_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "avg_roll_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "stddev_roll_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "var_roll_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "avg_pitch_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "stddev_pitch_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "var_pitch_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "avg_yaw_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "stddev_yaw_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "var_yaw_dumbbell(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "kurtosis_roll_forearm(NA cnt) : 19300;   % of NAs: 0.98"
## [1] "kurtosis_picth_forearm(NA cnt) : 19301;   % of NAs: 0.98"
## [1] "kurtosis_yaw_forearm(NA cnt) : 19622;   % of NAs: 1"
## [1] "skewness_roll_forearm(NA cnt) : 19299;   % of NAs: 0.98"
## [1] "skewness_pitch_forearm(NA cnt) : 19301;   % of NAs: 0.98"
## [1] "skewness_yaw_forearm(NA cnt) : 19622;   % of NAs: 1"
## [1] "max_roll_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "max_picth_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "max_yaw_forearm(NA cnt) : 19300;   % of NAs: 0.98"
## [1] "min_roll_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "min_pitch_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "min_yaw_forearm(NA cnt) : 19300;   % of NAs: 0.98"
## [1] "amplitude_roll_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "amplitude_pitch_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "amplitude_yaw_forearm(NA cnt) : 19300;   % of NAs: 0.98"
## [1] "var_accel_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "avg_roll_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "stddev_roll_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "var_roll_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "avg_pitch_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "stddev_pitch_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "var_pitch_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "avg_yaw_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "stddev_yaw_forearm(NA cnt) : 19216;   % of NAs: 0.98"
## [1] "var_yaw_forearm(NA cnt) : 19216;   % of NAs: 0.98"
```

```r
training <- training[, !(names(training) %in% vars_to_drop)]
dim(training)
```

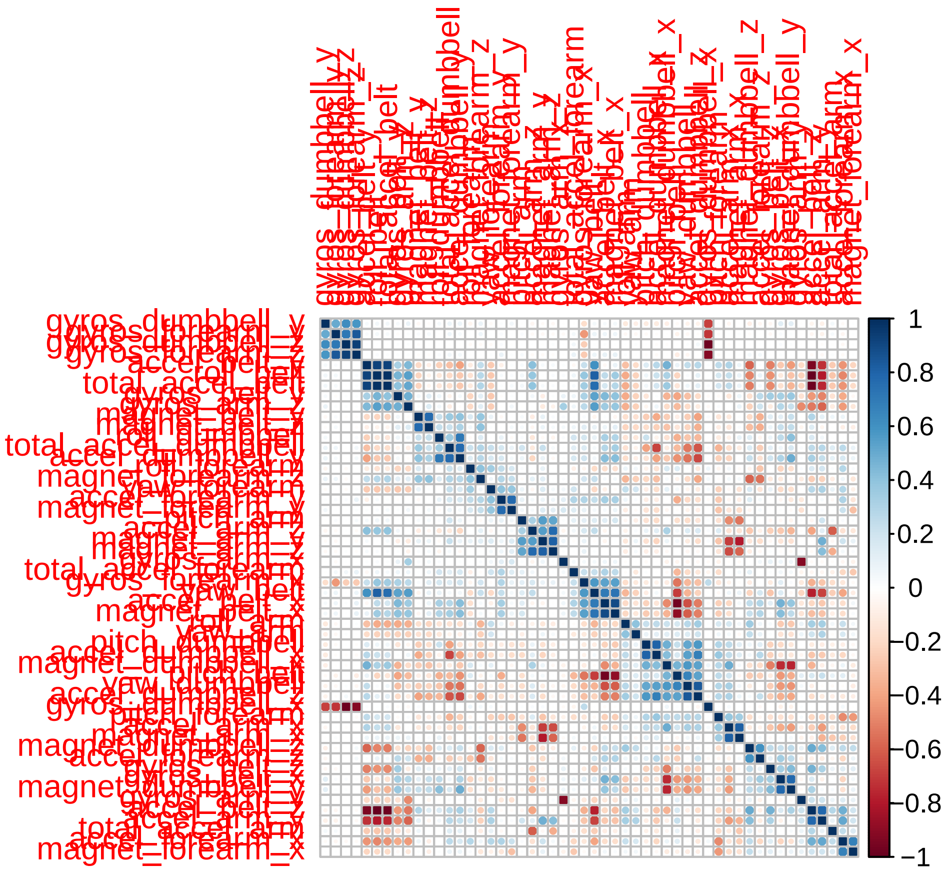
```
## [1] 19622    53
```

3. Clean data from nzv and highly correlated features:

```r
# nzv <- nearZeroVar(training)
# training <- training[,-nzv]
# dim(training)

# correlation analysis
corr_mat <- cor(training[, -53])
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
corrplot(corr_mat, order='hclust')
```



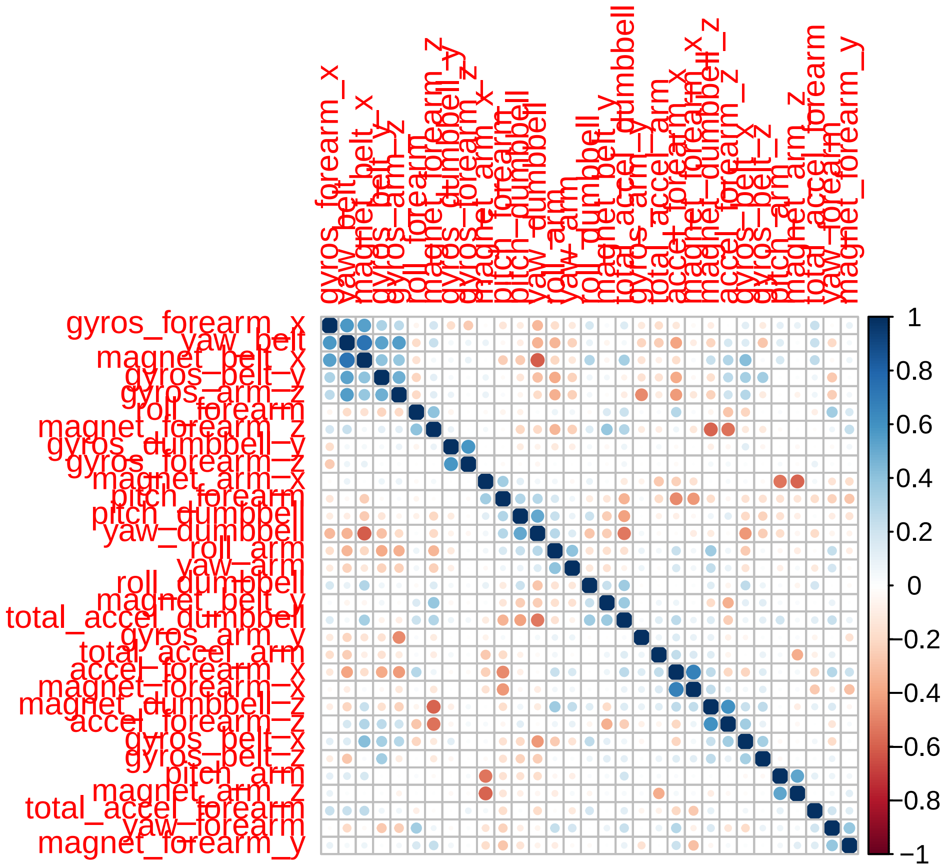
```r
# find attributes that are highly corrected (ideally >0.75)
highlyCorrelated <- findCorrelation(corr_mat, cutoff=0.75, names=F)
# print indexes of highly correlated attributes
print(highlyCorrelated)
```

```
##  [1] 10  1  9 22  4 36  8  2 37 35 38 21 34 23 25 13 48 45 31 33 18
```

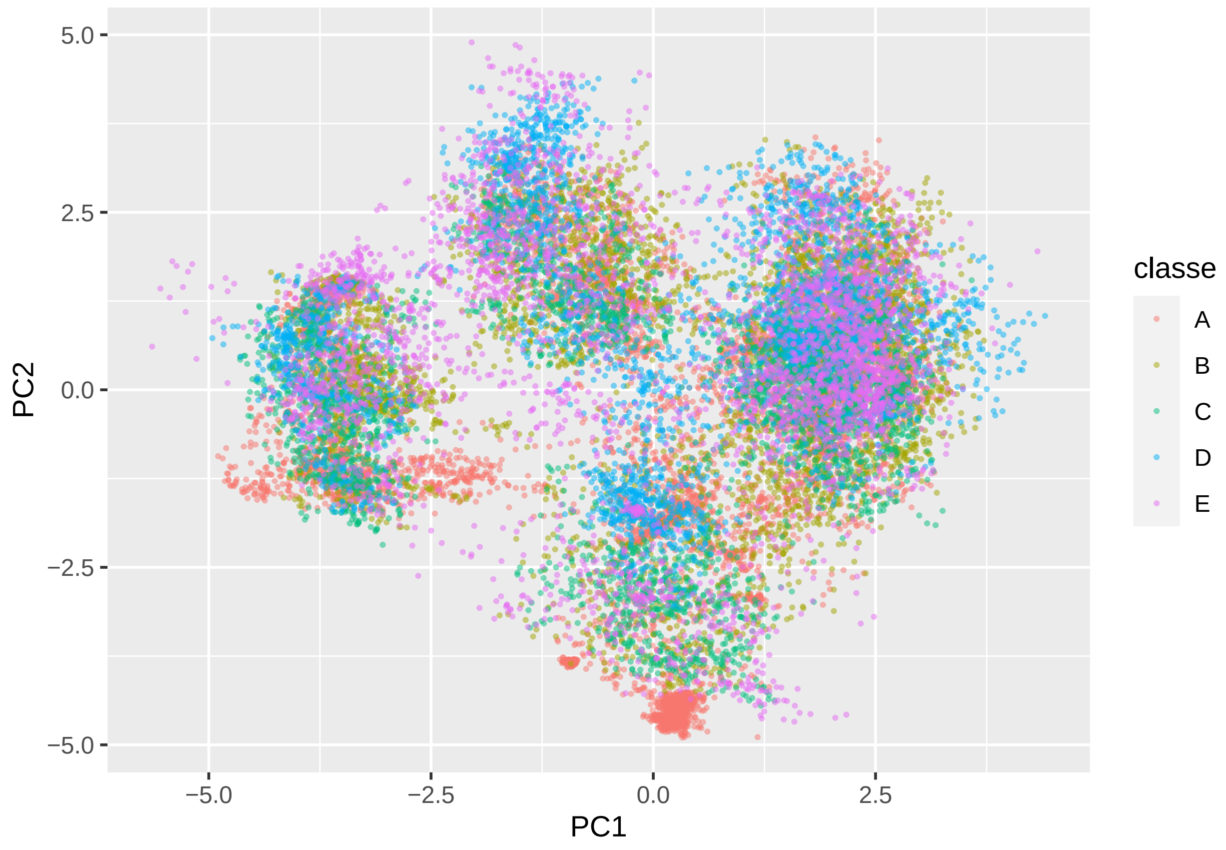
```r
dim(training[,-highlyCorrelated])
```

```
## [1] 19622    32
```

```r
training <- training[,-highlyCorrelated]

corr_mat_clean <- cor(training[,-dim(training)[2]])
corrplot(corr_mat_clean, order='hclust')
```

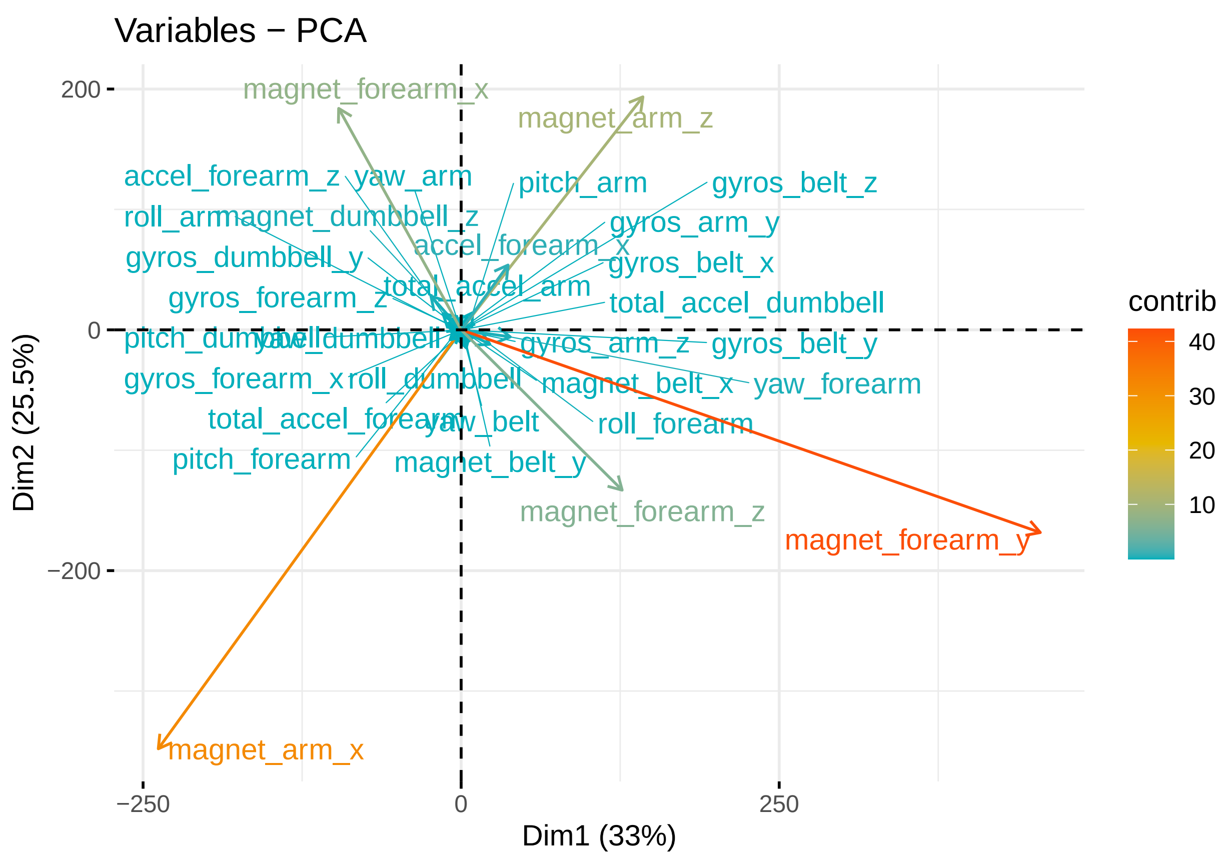


```r
# plot PCA 2d projection:
x_train <- training[,-dim(training)[2]]
y_train <- training$classe
pca <- preProcess(x_train, center=T, scale=T, method = 'pca', pcaComp = 2)
train_pca <- predict(pca, x_train)
train_pca$classe <- y_train
ggplot(train_pca, aes(PC1, PC2, color=classe)) + geom_point(size=0.4, alpha=0.5)
```



PCA

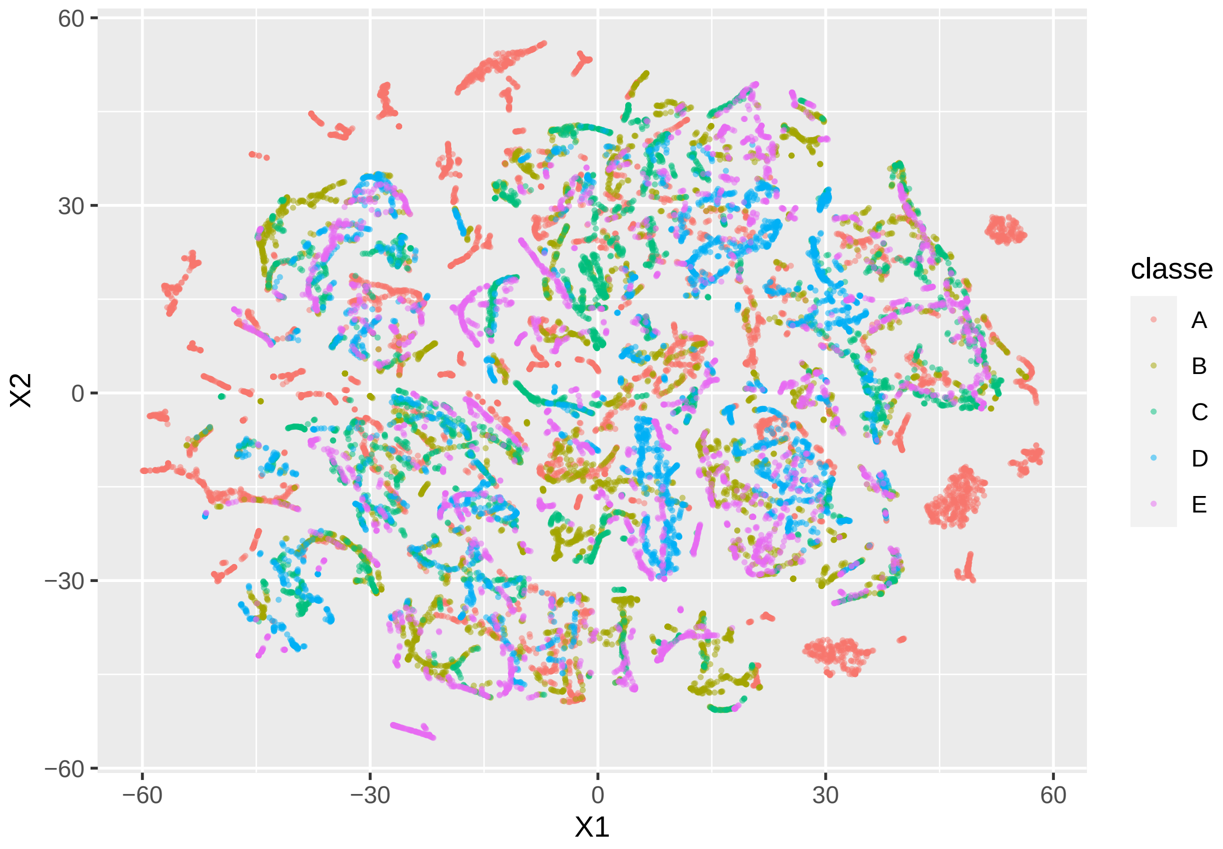
```
res.pca <- prcomp(x_train, scale = F)

fviz_pca_var(res.pca,
             col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE     # Avoid text overlapping
             )
```



```r
library(Rtsne)
res.tsne <- Rtsne(x_train, dims = 2, perplexity=30, verbose=TRUE, max_iter = 1000)
```

```
## Performing PCA
## Read the 19622 x 31 data matrix successfully!
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
##  - point 10000 of 19622
## Done in 4.79 seconds (sparsity = 0.005742)!
## Learning embedding...
## Iteration 50: error is 106.647916 (50 iterations in 4.82 seconds)
## Iteration 100: error is 99.093245 (50 iterations in 6.09 seconds)
## Iteration 150: error is 83.739379 (50 iterations in 5.08 seconds)
## Iteration 200: error is 79.493978 (50 iterations in 4.60 seconds)
## Iteration 250: error is 77.063150 (50 iterations in 4.59 seconds)
## Iteration 300: error is 3.133754 (50 iterations in 4.22 seconds)
## Iteration 350: error is 2.698266 (50 iterations in 4.24 seconds)
## Iteration 400: error is 2.401302 (50 iterations in 3.93 seconds)
## Iteration 450: error is 2.182553 (50 iterations in 3.85 seconds)
## Iteration 500: error is 2.013989 (50 iterations in 3.82 seconds)
## Iteration 550: error is 1.879732 (50 iterations in 3.99 seconds)
## Iteration 600: error is 1.769586 (50 iterations in 3.87 seconds)
## Iteration 650: error is 1.677124 (50 iterations in 4.10 seconds)
## Iteration 700: error is 1.598559 (50 iterations in 3.83 seconds)
## Iteration 750: error is 1.531313 (50 iterations in 4.11 seconds)
## Iteration 800: error is 1.473197 (50 iterations in 4.07 seconds)
## Iteration 850: error is 1.422886 (50 iterations in 3.88 seconds)
## Iteration 900: error is 1.379839 (50 iterations in 3.88 seconds)
## Iteration 950: error is 1.342532 (50 iterations in 3.85 seconds)
## Iteration 1000: error is 1.309882 (50 iterations in 3.91 seconds)
## Fitting performed in 84.73 seconds.
```

```
df_tsne <- data.frame(res.tsne$Y)
df_tsne$classe <- y_train
ggplot(df_tsne, aes(X1, X2, color=classe)) + geom_point(size=0.4, alpha=0.5)
```


X2
X1
classe
A
B
C
D
E