Predict Barbel Lifts Execution Based on IOT Devices Sensors Data

Pedro Loes

2022-09-18

Summary

This project is the final assigment Coursera Pratical Machine Learning course and was developed to build a model that predicts between 5 classes of correct and incorrect barbel lifts executions based on features retrieve by **IOT** devices. The project consist of **6** steps. First was data retrieving. Second, the pre processing step that consists of data wrangling and cleaning. Third was implemented a exploratory data analysis in shiny to understand features relation to the outcome response **classe**. A model was developed and the prediction were made to the test data. Finally a prediction on the test set was implemented and submitted to evaluation.

Load Data and Descriptive analysis

The link to the url of the data source is broken making impossible to retrieve a data catalog, meta information or the project design to understand for wich purpose and how the data was collected. Arbitrary decisions were taken about the project intents and data collection interpretations based on the assigment instructions.

- The `pml-training.csv` and `pml-testing` files were loaded using `data.table::fread` and transformed to `tibble` structure. All the missing information was transformed to `NA` data type and the first column index was droped.
- Dimensions
  - Training: **19622** observations, **158** features and **1** target
  - Testing: **20** observations, **158** features and **1** problem id variable

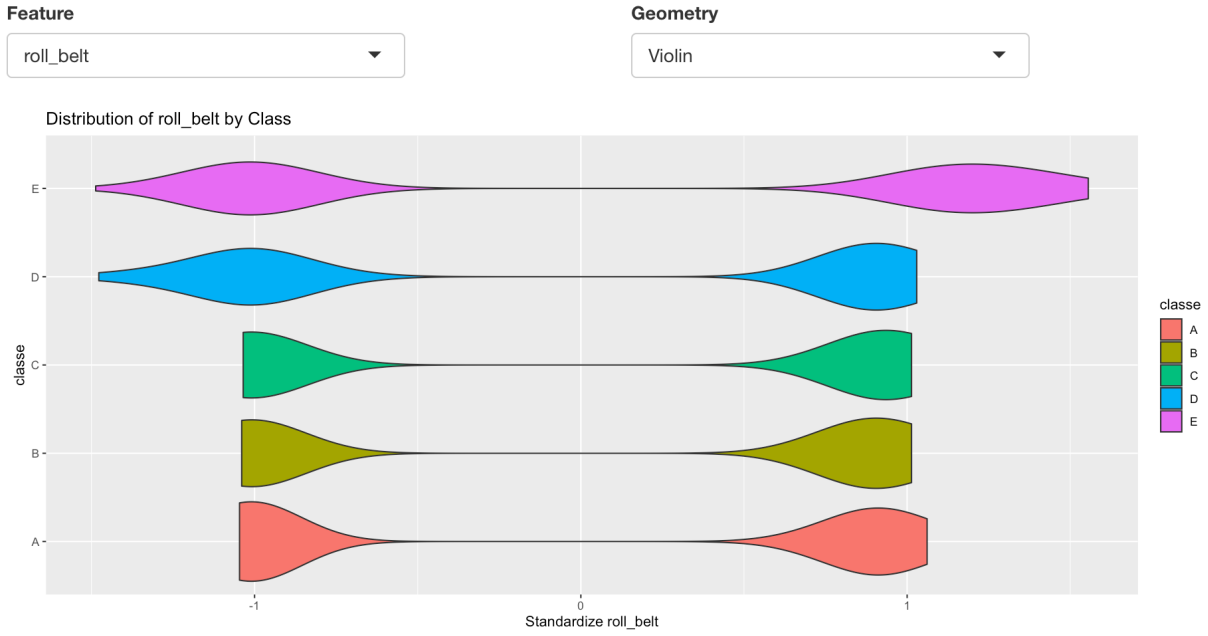
Pre Process

- Hypothesis of write failures on the **IOT** devices were desconsidered and all the missing dat was transformed to **0** representing no measures on the sensors for the exercise.
- Features with all observations containing missing information or features with **zero variance** were droped because no information or improvemet to the model came from these metrics.
- The categorical date feature `cvtd_timestamp` range from **2011-11-28** to **2011-12-05** and represent a week training of the **6** subjects, as well as `raw_timestamp_part_1` and `raw_timestamp_part_2`. Time in hours and minutes was also avaliable, but the exercises are executed in seconds so these features were droped as the main puspose of the project is not to understand pattern based on date and time but based on IOT exercises sensors giving the body position and movement during the exercise execution.

- Also the features `user_name`, `num_window` and `new_window` were droped. The reason was because it's not interesting to predict a class of the exercise execution based on the subject or window but based on the data captured by the sensor at the exercise execution. By this way the sensor could help any people in the future to execute the exercise properly and possible ring a bell if the exercise execution is incorrect and the model predict it.
- Dimensions
  - Training: **19622** observations, **144** features and **1** target
  - Testing: **20** observations, **144** features and **1** problem id variable

Exploratory Analysis

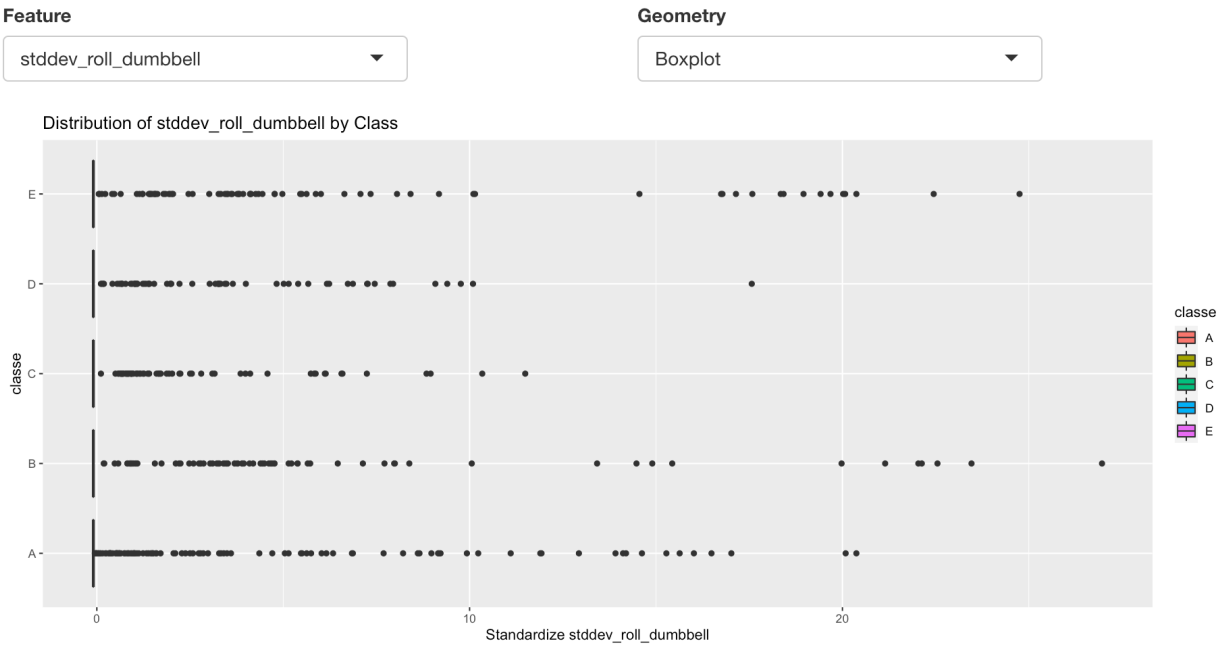
To compare the distribution of classes for each variable an shiny report was built in an separete html shiny report and published at shiny.io https://loes.shinyapps.io/finalProject_shiny with a select input to subset the data on the **IOT sensor** device to enable fast comparison between each metric and the outcome. The scale of the features were also normalized.



https://loes.shinyapps.io/finalProject_shiny/

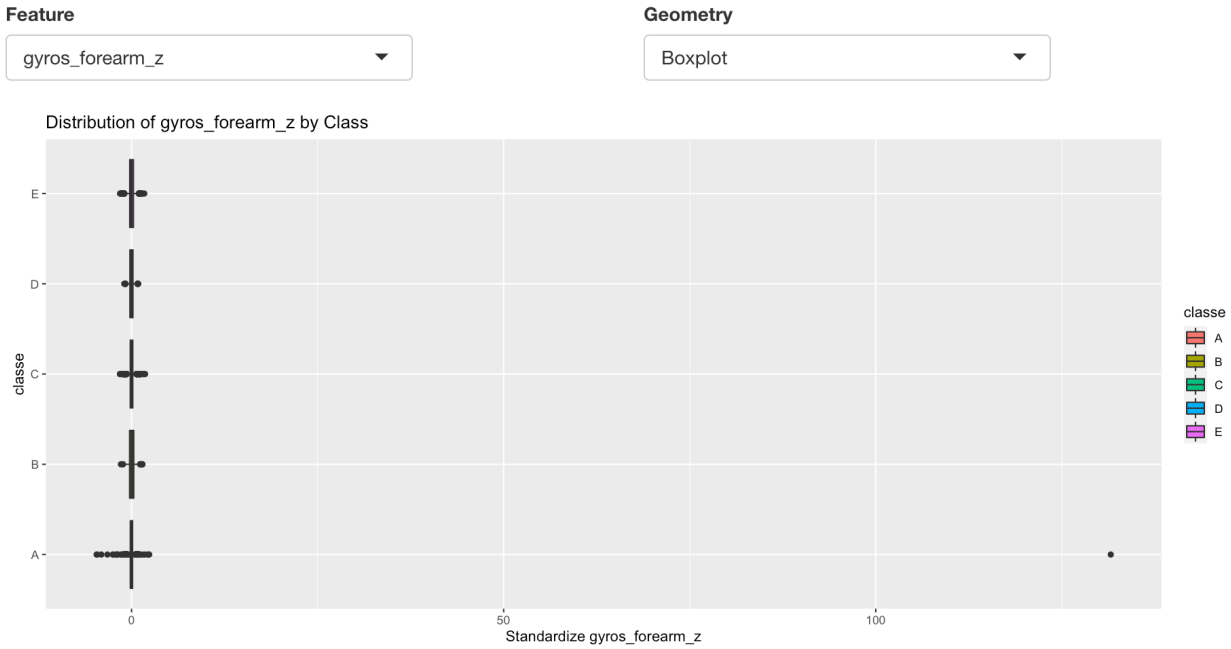
- An example of a potencial good predictor to distinguish `classe` **A** from the other `classe` is the **Roll Belt metric**. Just this class **E** has records above **1.2** treshold and these pattern could be used to correct classify as `classe` **E** when the new observation has values these big.

This tool was also used to spot **outliers** and strange pattern in the data.



https://loes.shinyapps.io/finalProject_shiny/

- An interesting pattern found was that a few metrics has dispersion up to **30** standard deviation above and below the mean like the metric `stddev_roll_dumbbell`. These observations were not treated as outliers and removed because they can help to distinguish between classes.



https://loes.shinyapps.io/finalProject_shiny/

- Some very attipical points were spoted very far from the the distribution center of mass like `giros_forearm_z` with more then **100** standard deviation from the mean. These observartions were

manteined and will be treated by the **PCA** techinic. Although these atipical behavior could be due to measure erros or device malfunction they were held due to the lack of information about the data extraction and the experiment design.

Fit Models

The first step was to reduce the curse of dimensionality of the dataset projecting the predictors on a lower dimensional space. After applying principal component analysis on the `preProcess` step, the original **144** features representing the **IOT** sensors were reduced to **44** features capturing up to **90%** of the data variability.

Three model were implemented and compared considering repeated cross validations with **3** repetitions and folds of size **10**.

- Tree model:
  - Accuracy: **0.4095242**.
  - The best model used the parameters cp = **0.01880074**
- Naive Bayes:
  - Accuray (gaussian): **0.2422454**
  - Accuracy (non parametric kernel): **0.5864673**.
- Random Forest model:
  - Accuracy: **0.9738727**.
  - Mtry parameter used was $\sqrt{44} = 7$

Predictions

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20
B	A	B	A	A	E	D	B	A	A	B	C	B	A	E	E	A	B	B	B

- The final best model **Random Forest** perform much better then the **Naive Bayes** more then 100% better related to the model with just **1** tree. It shows that the bootstrap of trees with random selected predictors results in a much better performance. This fact is probably because on the tree model a good predictor feature was always select on the cross validation process not giving space to other features shiny and help on the overall model.
- There's a down side for the result of the random forest. The model computational cost was at least **20** more expensive compared to the other **2** simpler models.