hyperP

Data description

hyperP (*the rotating hyperplane problem*) is a very popular artificial dataset used for analysing concept drift. It represents a 10-dimensional unit cube which is sectioned into two classes by 10-dimensional hyperplane defined as:

$$\sum_{i=1}^{10} w_i x_i = w_o$$

where each variable $x_i \sim U[0, 1]$.

The hyperplane serves as a boundary, it splits the data into two classes (so we have a classification problem). Observations are labeled as 1 if $\sum_{i=1}^{10} w_i x_i \geq w_o$ and as 0 otherwise.

Concept drift is introduced by varying the position and orientation of the hyperplane with time, i.e. by changing weights w_i for each of 10 variables (adding constant $\delta = 0.001$):

$$w_i^{'} = w_i + \delta, \;\; i = 1, \ldots, 10$$

These continuous changes are supposed to produce real gradual (incremental) concept drift [1].

It was introduced in [Mining Time-Changing Data Streams] and used in several works regarding concept drift, with different settings of number of variables, number of classes and speed factor δ:

1) KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift 2 A Taxonomic Look at Instance-based Stream Classifiers
2) Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts
3) MIning Data Streams With Concept Drift In Massive Online Analysis Framework
4) New Ensemble Methods For Evolving Data Streams

Data reading and preparation

Reading the data:

```
hyperP <- read.table(file = paste0(path,'rotatingHyperplane_data.txt'), stringsAsFactors = FALSE)
hyperPLabels <- read.table(file = paste0(path,'rotatingHyperplane_labels.txt'), stringsAsFactors = FALSE

colnames(hyperPLabels) <- c('y')
hyperP <- cbind(hyperP, hyperPLabels) # joining labels with observations
hyperP$y <- factor(hyperP$y)
hyperP <- hyperP[!duplicated(hyperP),]
rm(hyperPLabels)
```

Quick look at first rows of dataset below. It has 10 numerical variables with 200 000 observations and binary target variable y.

The dataset is balanced (equal share of both classes), with no null values or constant variables.

```
prop.table(table(hyperP$y)) # balanced class proportions
```

```
##
##        0        1
## 0.500325 0.499675
```

Table 1: First rows of dataset

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	y
0.40	0.35	0.29	0.51	0.12	0.77	0.66	0.16	0.38	0.14	0
0.28	0.08	0.61	0.95	0.27	0.19	0.00	0.68	0.49	0.49	0
0.23	0.89	0.04	0.59	0.66	0.12	0.65	0.98	0.21	0.37	1
0.28	0.34	0.27	0.12	0.14	0.55	0.49	0.02	0.36	0.24	0
0.25	0.78	0.23	0.98	0.80	0.84	0.16	0.64	0.01	0.63	0

```
sapply(hyperP, function(x){sum(is.na(x))}) # no NA values
```

```
##  V1  V2  V3  V4  V5  V6  V7  V8  V9 V10   y
##   0   0   0   0   0   0   0   0   0   0   0
```

```
colnames(hyperP)[nearZeroVar(hyperP)]  # no nearly constant variables
```

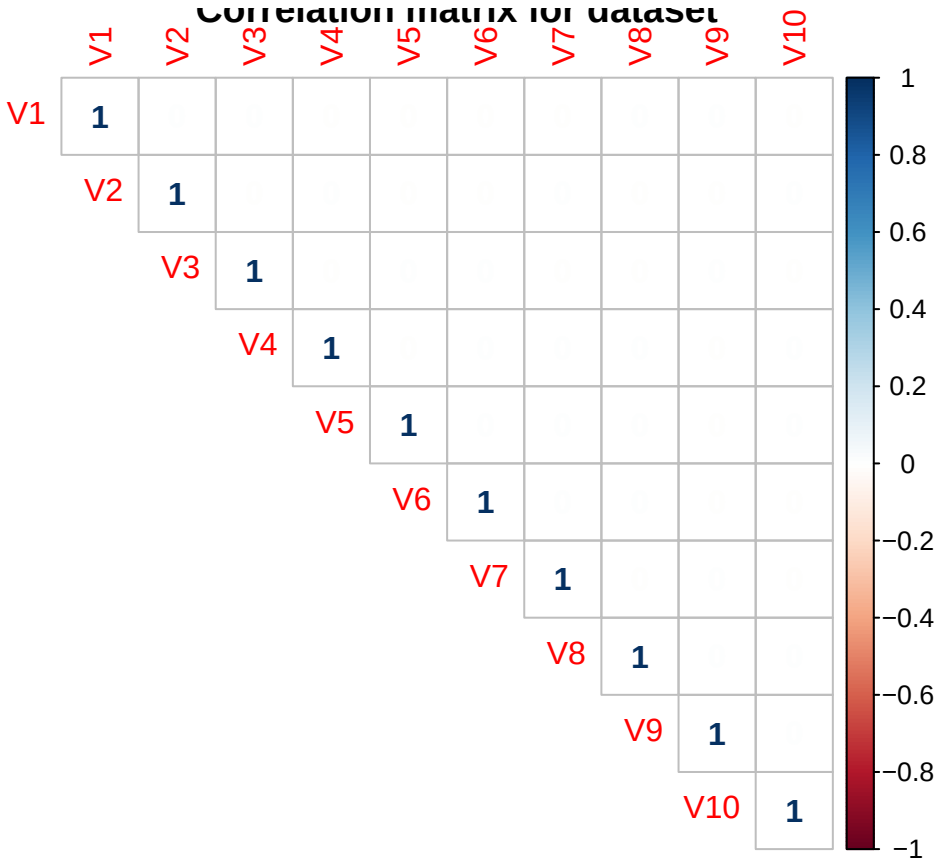
```
## character(0)
```

There is no high correlations between variables or internal linear combinations of them in dataset.

```
findCorrelation(cor(hyperP[ ,-c(11)]), cutoff = 0.7) #no correlated variables
```

```
## integer(0)
```

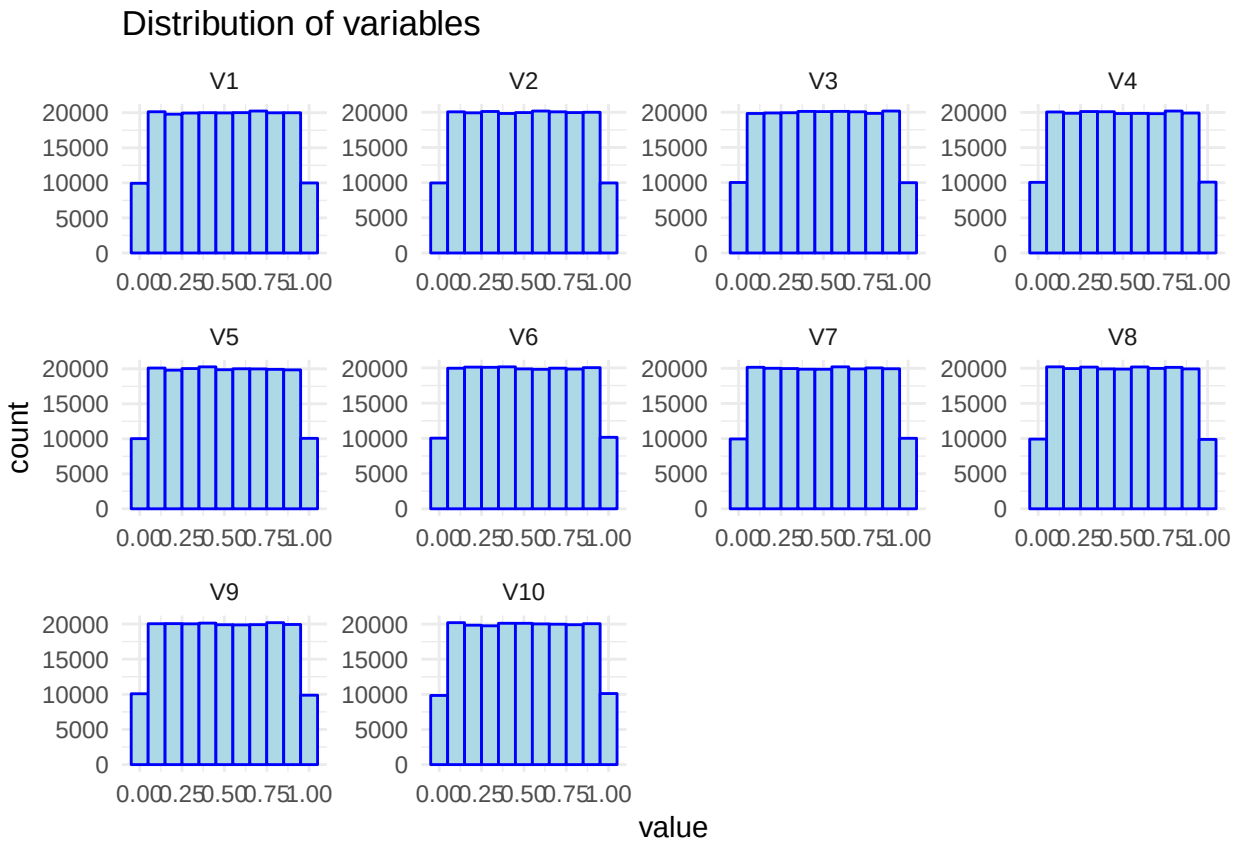
```
corrplot(cor(hyperP[ ,-c(11)]), method = "number", type = 'upper', title = 'Correlation matrix for datas
```



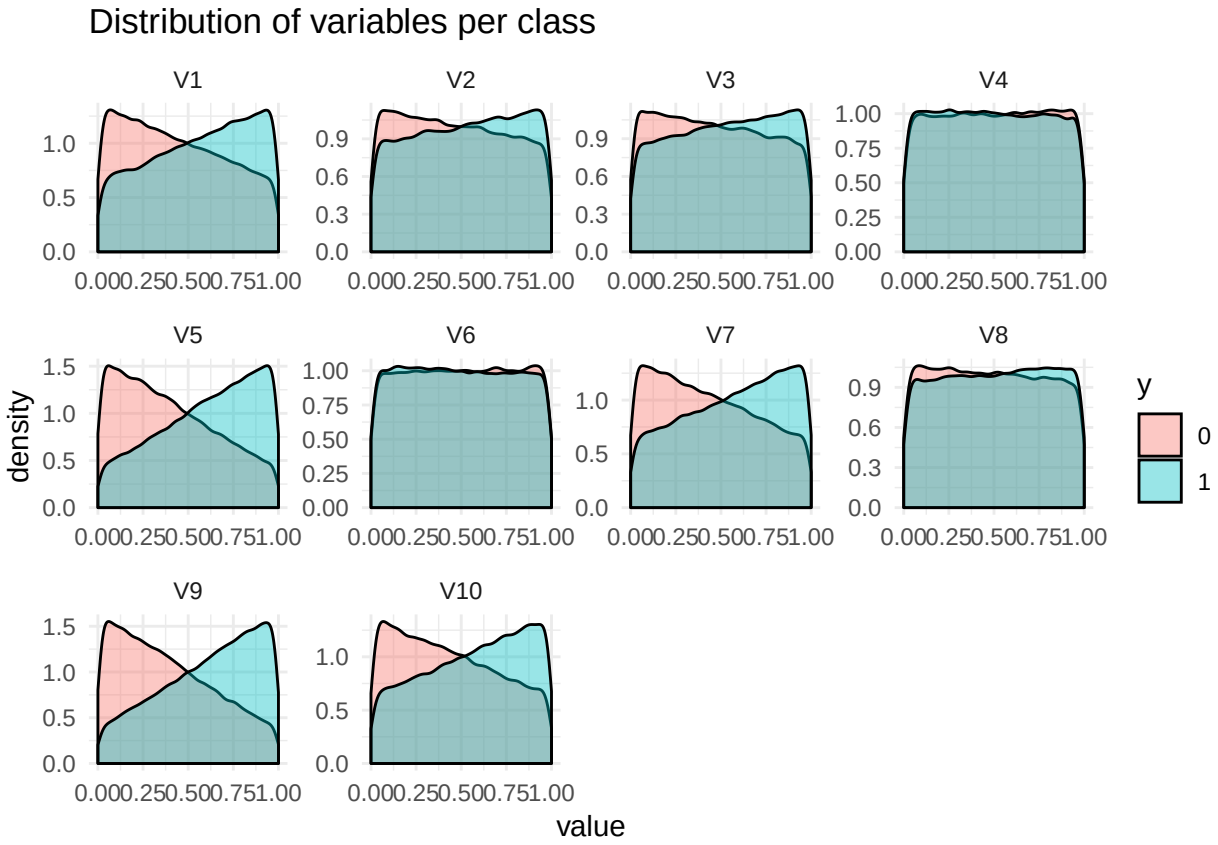

```
findLinearCombos(hyperP[ ,-c(11)]) # no linear combinations
```

```
## $linearCombos
## list()
##
## $remove
## NULL
```

All variables are scaled to interval [0,1] and has almost uniform distribution (beside ends of the interval).



After preliminary visual analysis we can see that variable V5 and V9 have probably the highest discrimination



power.

We assume that observations in dataset are ordered chronologically, so the first rows represent historical data and the last rows actual data. For purpose of our concept drift analysis we partition the dataset into two parts, representing 2 different time periods, by creating artificial wash out period (removing 30% of middle observations). Each new dataset is divided into train (n=56 000) and test set (20% of observations, n=14 000):

```
## List of 4
##  $ train1:'data.frame':  56000 obs. of  11 variables:
##   ..$ V1 : num [1:56000] 0.082 0.812 0.311 0.579 0.728 ...
##   ..$ V2 : num [1:56000] 0.98 0.68 0.82 0.12 0.89 ...
##   ..$ V3 : num [1:56000] 0.84 0.42 0.83 0.89 0.24 ...
##   ..$ V4 : num [1:56000] 0.204 0.825 0.899 0.044 0.115 ...
##   ..$ V5 : num [1:56000] 0.14 0.58 0.74 0.51 0.45 ...
##   ..$ V6 : num [1:56000] 0.042 0.989 0.385 0.642 0.201 ...
##   ..$ V7 : num [1:56000] 0.49 0.28 0.47 0.99 0.83 ...
##   ..$ V8 : num [1:56000] 0.7 0.17 0.14 0.51 0.12 ...
##   ..$ V9 : num [1:56000] 0.51 0.73 0.98 0.26 0.36 ...
##   ..$ V10: num [1:56000] 0.3783 0.0052 0.7839 0.568 0.7054 ...
##   ..$ y  : Factor w/ 2 levels "0","1": 1 1 2 2 1 1 1 2 1 2 ...
##  $ test1 :'data.frame':  14000 obs. of  11 variables:
##   ..$ V1 : num [1:14000] 0.4 0.85 0.43 0.37 0.4 ...
##   ..$ V2 : num [1:14000] 0.35 0.35 0.13 0.57 0.07 ...
##   ..$ V3 : num [1:14000] 0.29 0.6 0.67 0.84 0.17 ...
##   ..$ V4 : num [1:14000] 0.51 0.29 0.52 0.98 0.72 ...
##   ..$ V5 : num [1:14000] 0.12 0.88 0.55 0.43 0.49 ...
##   ..$ V6 : num [1:14000] 0.77 0.74 0.92 0.78 0.63 ...
##   ..$ V7 : num [1:14000] 0.66 0.42 0.38 0.57 0.12 ...
```

```
##   ..$ V8 : num [1:14000] 0.16 0.38 0.17 0.98 0.13 ...
##   ..$ V9 : num [1:14000] 0.38 0.27 0.13 0.93 0.5 ...
##   ..$ V10: num [1:14000] 0.14 0.73 0.48 0.2 0.44 ...
##   ..$ y  : Factor w/ 2 levels "0","1": 1 2 1 2 1 1 1 1 2 1 ...
##  $ train2:'data.frame':  55999 obs. of  11 variables:
##   ..$ V1 : num [1:55999] 0.41 0.82 0.29 0.78 0.49 ...
##   ..$ V2 : num [1:55999] 0.19 0.64 0.97 0.95 0.75 ...
##   ..$ V3 : num [1:55999] 0.257 0.0891 0.4773 0.1302 0.0077 ...
##   ..$ V4 : num [1:55999] 0.51 0.14 0.15 0.31 0.33 ...
##   ..$ V5 : num [1:55999] 0.17 0.45 0.95 0.7 0.84 ...
##   ..$ V6 : num [1:55999] 0.6 0.12 0.27 0.19 0.17 ...
##   ..$ V7 : num [1:55999] 0.32 0.62 0.5 0.81 0.47 ...
##   ..$ V8 : num [1:55999] 0.022 0.85 0.239 0.666 0.374 ...
##   ..$ V9 : num [1:55999] 0.073 0.184 0.482 0.736 0.508 ...
##   ..$ V10: num [1:55999] 0.11 0.64 0.66 0.68 0.56 ...
##   ..$ y  : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 2 2 2 ...
##  $ test2 :'data.frame':  14000 obs. of  11 variables:
##   ..$ V1 : num [1:14000] 0.93 0.98 0.21 0.74 0.35 ...
##   ..$ V2 : num [1:14000] 0.4 0.18 0.61 0.89 0.46 ...
##   ..$ V3 : num [1:14000] 0.25 0.65 0.65 0.99 0.53 ...
##   ..$ V4 : num [1:14000] 0.28 0.84 0.41 0.55 0.14 ...
##   ..$ V5 : num [1:14000] 0.58 0.89 0.39 0.39 0.18 ...
##   ..$ V6 : num [1:14000] 0.13 0.8 0.93 0.62 0.6 ...
##   ..$ V7 : num [1:14000] 0.52 0.12 0.54 0.96 0.56 ...
##   ..$ V8 : num [1:14000] 0.026 0.572 0.402 0.255 0.836 ...
##   ..$ V9 : num [1:14000] 0.059 0.059 0.027 0.777 0.164 ...
##   ..$ V10: num [1:14000] 0.38 0.43 0.24 0.92 0.37 ...
##   ..$ y  : Factor w/ 2 levels "0","1": 1 1 1 2 1 2 2 1 1 2 ...
```

We can see that class share in each of new datasets is still balanced.

```
# the same class balance in each dataset
cat("\n","train1 ", round(prop.table(table(hyperPWithBreak$train1$y)), 2) , "\n",
    "test1 ", round(prop.table(table(hyperPWithBreak$test1$y)), 2) , "\n",
    "train2 ", round(prop.table(table(hyperPWithBreak$train2$y)), 2), "\n",
    "test2 ", round(prop.table(table(hyperPWithBreak$test2$y)), 2) )
```

```
##
##  train1  0.5 0.5
##  test1  0.49 0.51
##  train2  0.5 0.5
##  test2  0.5 0.5
```

Models preparation

We will fit 2 models for this dataset; random forest and logistic regression. (TODO dodać te knn)

```
# random forest
predict_function1 <- function(m,x,...) predict(m, x, ..., probability=TRUE)$predictions[,2]
model_old1 <- ranger(y ~ ., data = hyperPWithBreak$train1, probability=TRUE, importance = 'permutation'
model_new1 <- ranger(y ~ ., data = hyperPWithBreak$train2, probability=TRUE, importance = 'permutation'
```

```
## Growing trees.. Progress: 86%. Estimated remaining time: 4 seconds.
```

```
# logistic regression
```

```
predict_function2 <- function(m,x,...) predict(m, x, ..., type ='response')
model_old2 <- glm(y ~.,family=binomial(link='logit'),data=hyperPWithBreak$train1)
model_new2 <- glm(y ~.,family=binomial(link='logit'),data=hyperPWithBreak$train2)
```

We can check their accuracy on their corresponding test data (TODO: porównać do literatury) which is above 80% for both datasets and both models.

```
# random forest
# old model old test
pred <- predict(model_old1, hyperPWithBreak$test1, probability=TRUE)$predictions
pred <- apply(pred,1,which.max)
test <-  as.numeric(hyperPWithBreak$test1$y)
table(pred, test)
```

```
##     test
## pred    1    2
##    1 5865 1171
##    2 1060 5904
```

```
a <- data.frame(cbind(pred, test))
sum(a$pred == a$test)/nrow(a)
```

```
## [1] 0.8406429
```

```
# new model new test
pred <- predict(model_new1, hyperPWithBreak$test2, probability=TRUE)$predictions
pred <- apply(pred,1,which.max)
test <-  as.numeric(hyperPWithBreak$test2$y)
table(pred, test)
```

```
##     test
## pred    1    2
##    1 5951 1142
##    2 1074 5833
```

```
a <- data.frame(cbind(pred, test))
sum(a$pred == a$test)/nrow(a)
```

```
## [1] 0.8417143
```

```
# logistic regresion
# old model old test
pred <-ifelse(predict(model_old2, hyperPWithBreak$test1, type='response') >= 0.5, 1,0)
test <-  as.numeric(as.character(hyperPWithBreak$test1$y))
table(pred, test)
```

```
##     test
## pred    0    1
##    0 5957 1055
##    1  968 6020
```

```
a <- data.frame(cbind(pred, test))
sum(a$pred == a$test)/nrow(a)
```

```
## [1] 0.8555
```

```
# new model new test
pred <-ifelse(predict(model_new2, hyperPWithBreak$test2, type='response') >= 0.5, 1,0)
```

```
test <-  as.numeric(as.character(hyperPWithBreak$test2$y))
table(pred, test)
```

```
##     test
## pred    0    1
##    0 5976 1136
##    1 1049 5839
```

```
a <- data.frame(cbind(pred, test))
sum(a$pred == a$test)/nrow(a)
```

```
## [1] 0.8439286
```

We can also check how old model can predict new data. It dropped to 70% of accuracy for both models.

```
# random forest
# old model new test
pred <- predict(model_old1, hyperPWithBreak$test2, probability=TRUE)$predictions
pred <- apply(pred,1,which.max)
test <-  as.numeric(hyperPWithBreak$test2$y)
table(pred, test)
```

```
##     test
## pred    1    2
##    1 4981 2137
##    2 2044 4838
```

```
a <- data.frame(cbind(pred, test))
sum(a$pred == a$test)/nrow(a)
```

```
## [1] 0.7013571
```

```
# logistic regression
# old model new test
pred <-ifelse(predict(model_old2, hyperPWithBreak$test2, type='response') >= 0.5, 1,0)
test <-  as.numeric(as.character(hyperPWithBreak$test2$y))
table(pred, test)
```

```
##     test
## pred    0    1
##    0 4928 2139
##    1 2097 4836
```

```
a <- data.frame(cbind(pred, test))
sum(a$pred == a$test)/nrow(a)
```

```
## [1] 0.6974286
```

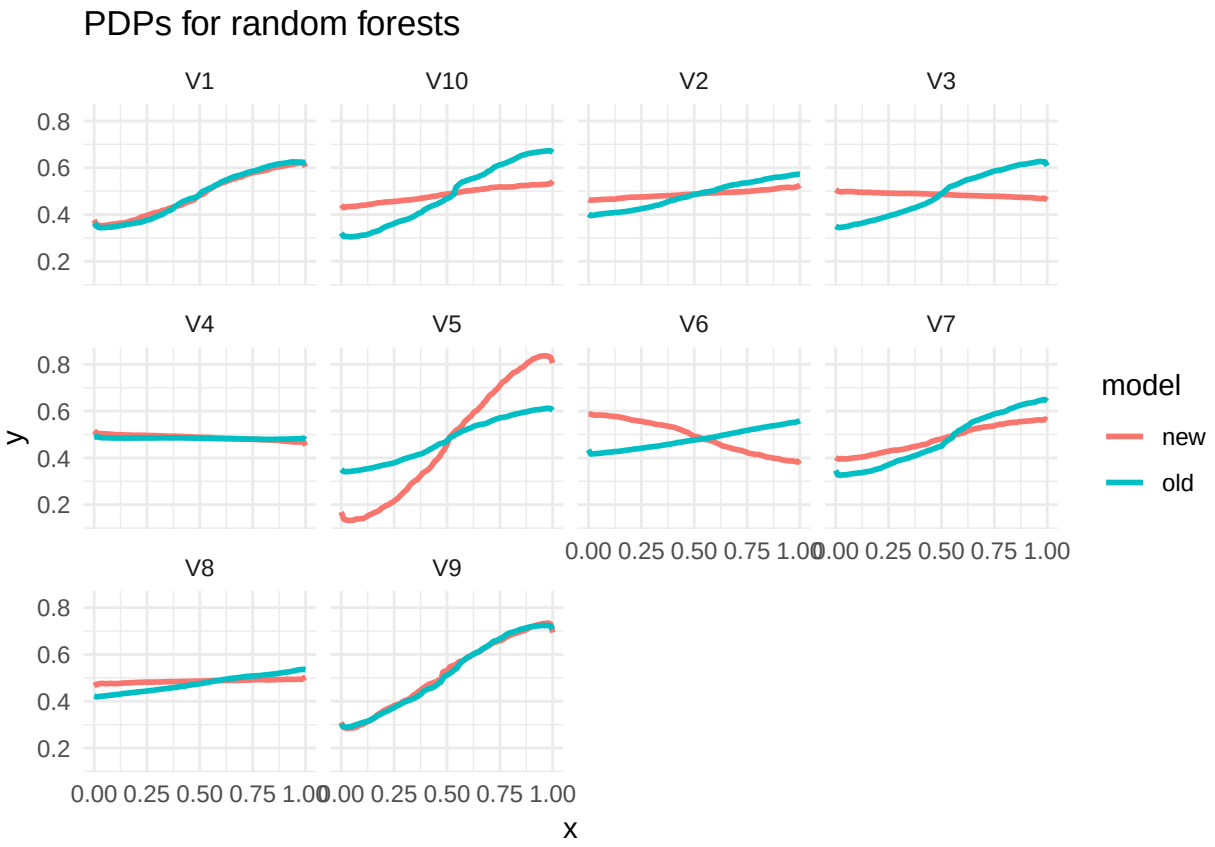
PDP

Now we can plot PDP for both models and both datasets. We will choose 200 random observations from test dataset to create PDPs.

First random forest.

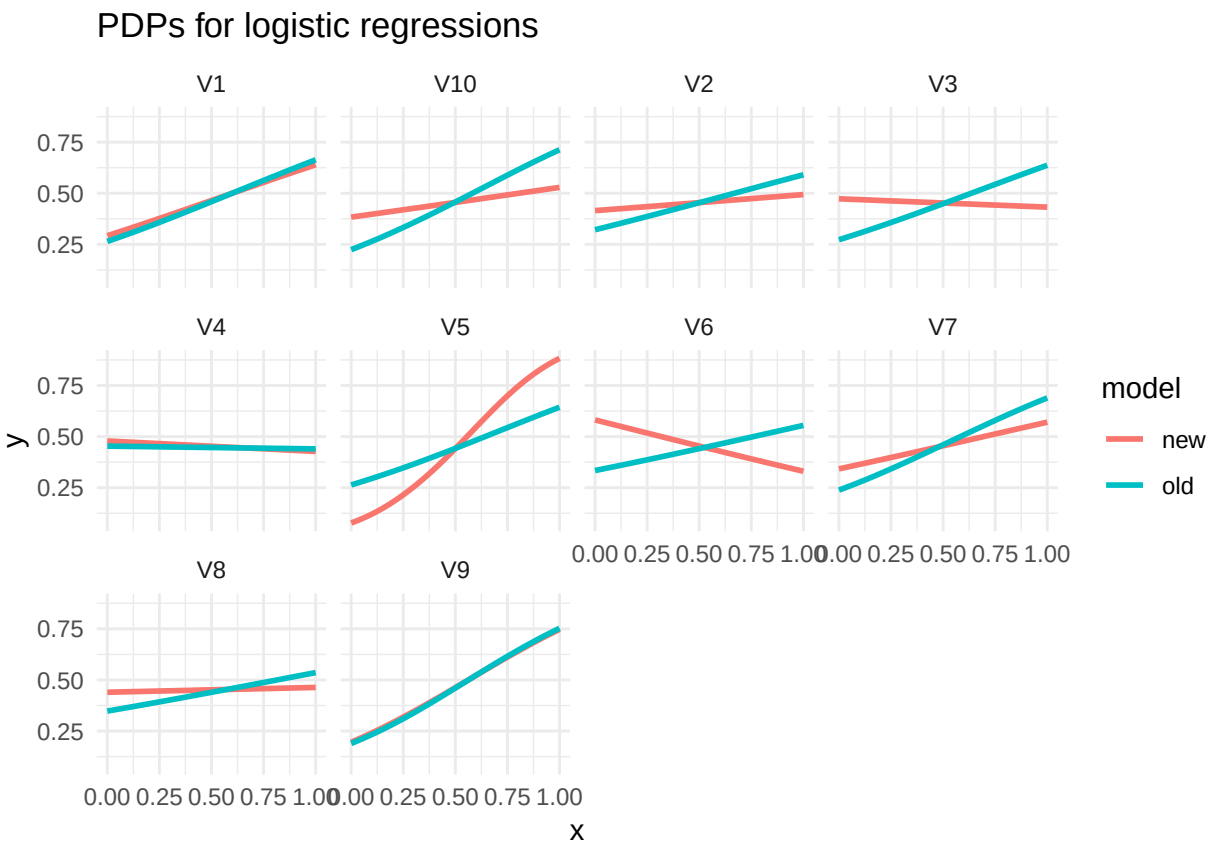
We can see that for each variable there is at least slight difference in PDPs shape and position between two models (old, new) (TODO check if drift was really set for each variable), but change for variable V1 and V4 seems least significant. The most significant (visually) change is for variable V5.

We can see that for each variable PDP for old and for new dataset are centered in the same point, so there is no significant vertical shift, but there is change in shape (like V10, from sigmoid to linear shape) and orientation of curves (like V6, for old model probability is increasing along the increase of variable values, for new model is decreasing). All curves are smooth.



Then for logistic regression.

Here we can see exactly the same trends as for random forests, but curves are more linear.



Concept drift detection - intro

We will test the following hypothesis:

$H_o : d(PDP_1, PDP_2) = 0$ vs. $H_a : d(PDP_1, PDP_2) \neq 0$

For above hypothesis, we assume d is any non-negative, symmetric dissimilarity measure. In this hypothesis curves don't have to be exactly equal but have to be similair with respect to chosen d.

Distance 1: L2 (drifter)

Defined as:

$$\int_0^\infty f(x)\, dx$$

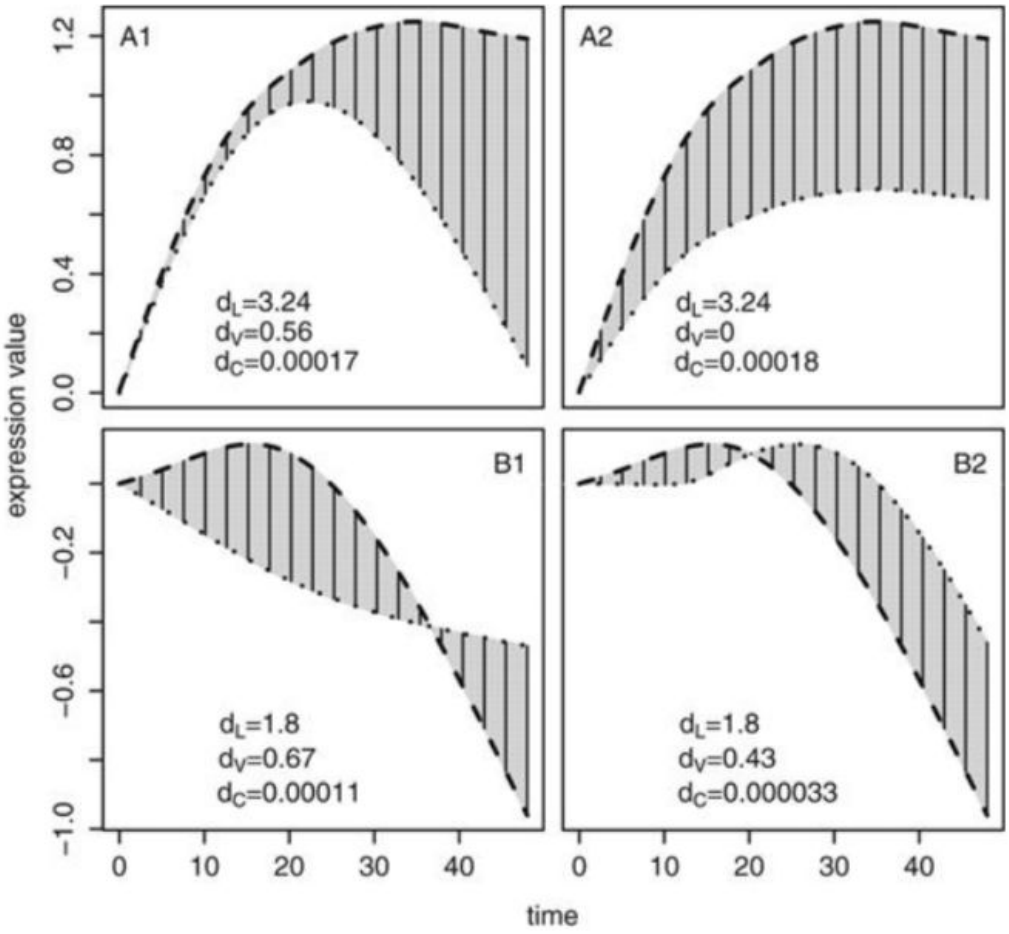
$$d_{L2}(PDP_1(x), PDP2(x)) = (\int_{x \in X} (PDP_1(x) - PDP_2(x))^2 dx)^{1/2}$$

d_{L2} explores vertical differences between two curves, curves shifted horizontally (in x domain) are treated as different curves, thus curves with similair shape but shifted vertically can be treated in the same way as curves with completely different shape, see Figure 1 from Minas C., Waddell S.J., Montana G.:

Random forest: variable V5 is detected as the one that changes the most, then, V10, V6, V3 and V2, V7, which corresponds to area under between both curves.

```
calculate_model_drift(model_old1, model_new1,
                      hyperPWithBreak$test2[,-c(11)],
```

Fig. 1.



View large Download slide

Four different comparisons between two gene curves illustrating the effects of using the d_L, d_V and d_C distances. The curves in A1 and A2 have the same d_L distances (represented by the shaded regions with vertical lines) despite clearly visibly differences in the temporal gene expression patterns. Similarly, the curves in B1 and B2 have the same d_L distances, although the curves in B1 have quite different time-varying behaviour while those in B2 have the same shape but are time-delayed. These shape-related differences are better captured by d_V and d_C.

Figure 1: Figure 1

```
                      hyperPWithBreak$test2$y == "1", # odp do tego co wybrane w prediction_function
                      max_obs = 200,
                      predict_function = predict_function1)
```

```
##                   Variable    Shift  Scaled
##   ---------------------------------------------
##                         V1     0.01     2.4
##                         V2     0.04     8.7
##                         V3     0.10    20.7  *
##                         V4     0.02     4.4
##                         V5     0.15    29.7  *
##                         V6     0.12    24.5  *
##                         V7     0.06    12.4  .
##                         V8     0.04     7.1
##                         V9     0.02     3.0
##                        V10     0.10    20.5  *
```

Logistic regression: the same is detected using logistic regression model.

```
calculate_model_drift(model_old2, model_new2,
                      hyperPWithBreak$test2[,-c(11)],
                      hyperPWithBreak$test2$y == "1", # odp do tego co wybrane w prediction_function
                      max_obs = 200,
                      predict_function = predict_function2)
```

```
##                   Variable    Shift  Scaled
##   ---------------------------------------------
##                         V1     0.01     2.2
##                         V2     0.05    10.1  .
##                         V3     0.12    23.5  *
##                         V4     0.01     2.5
##                         V5     0.16    31.3  **
##                         V6     0.13    26.8  *
##                         V7     0.06    11.2  .
##                         V8     0.04     8.6
##                         V9     0.00     0.8
##                        V10     0.10    19.7  .
```

Distance 2: visual L2

Used in Minas C., Waddell S.J., Montana G., proposed in Marron J.S., Tsybakov A.B..

Takes into accounts both vertical and horizontal differences between curves (treats them as points on the plane (X,Y)) and is more similair to natural, „naked eye" comparison of functions.

Distance 3: DTW

Distance 4: Hausdorff distance

Distance 5: Frechet distance

Distance 6: . . .