Term Project - Group 80

Sari Ropponen, Outi Boman, Loic Dreano

2022-12-09

Description of data

The training data npf_train.csv and testing data npf_test_hidden.csv were downloaded. The data includes 104 features. The number of observations in training data is 464 and 965 in testing data.

Table 1: Summary of the dataset

	npf_train	npf_test
Measurements	464	965
Variables	104	104

The features include a lot of daily measurements taken in Hyytiälä forestry field station. Some of the features like temperature T and CO2 are measured in different heights. The height is indicated in the name of the feature, for example, T84.mean is the mean temperature at 8.4 meters above the mast base.

Preprocessing data

The summary of the first four features in training data is: — I changed this to be the 4 first because only they are needed to do the next steps of data cleaning.

```
##        id            date              class4          partlybad
##  Min.   :  1.0   Length:464         Length:464         Mode :logical
##  1st Qu.:116.8   Class :character   Class :character   FALSE:464
##  Median :232.5   Mode  :character   Mode  :character
##  Mean   :232.5
##  3rd Qu.:348.2
##  Max.   :464.0
```

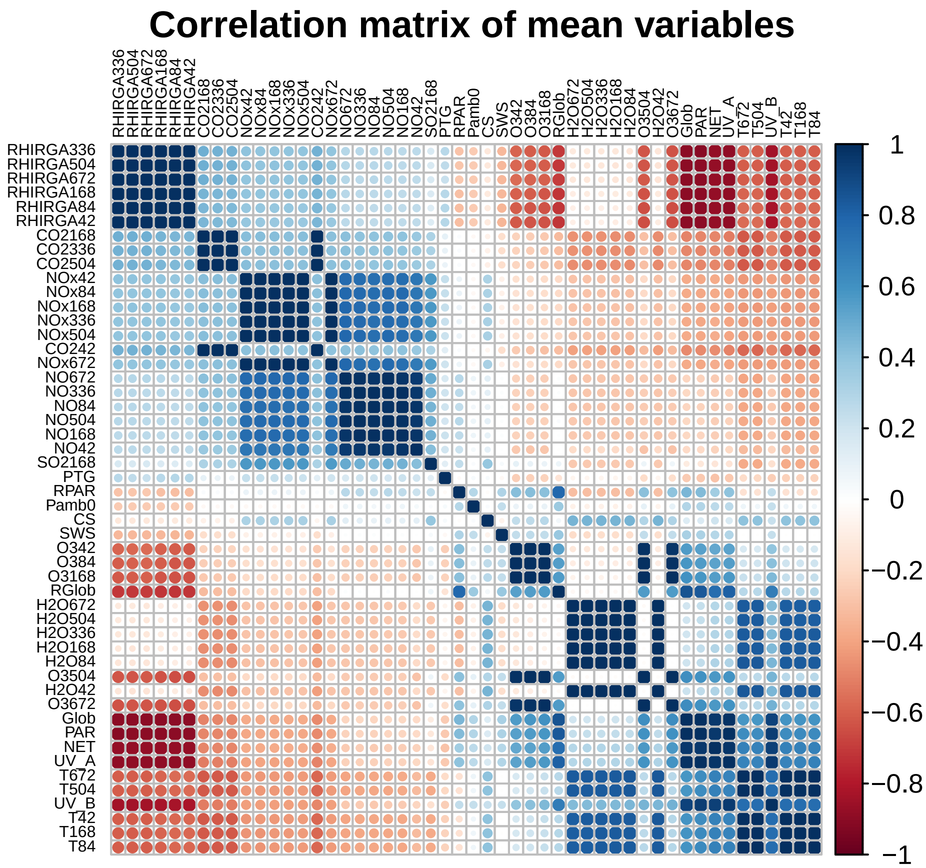
The column "date" was set to be the row names. Columns "id", "date" and "partlybad" were removed from the data. Because the value of the logical variable "partlybad" is FALSE for all the observations, it doesn't give any information.

A qualitative variable "class2" is added to the training data. It gets either value "event" or "nonevent" according to "class4". Variable "class4" indicates the type of the event if it has happened, values "Ia", "Ib" or "II", or "nonevent" if no event has happened during the day.

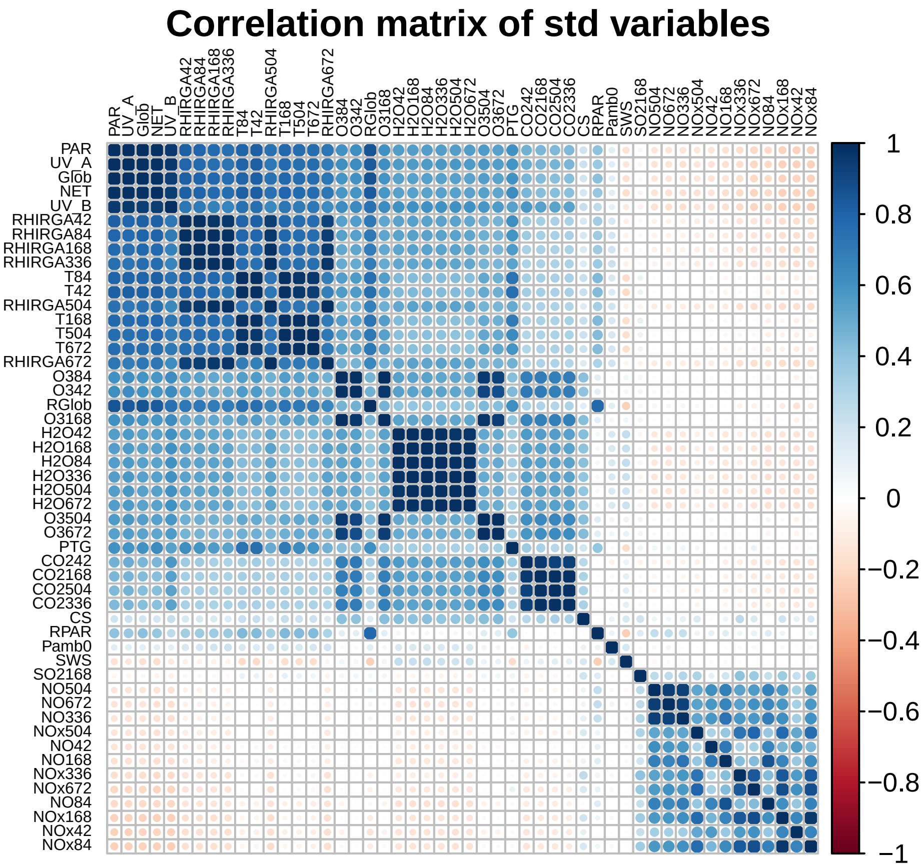
The task is to build a binary classifier which predicts if an NPF event will happen or not during the day according to the observed measurements.

Data exploration

Because the data includes same measurements at different heights it is expected that there are correlation between variables. Correlations between different mean values are shown in the matrix below:


Correlation matrix of mean variables

The same picture for different standard deviation values is

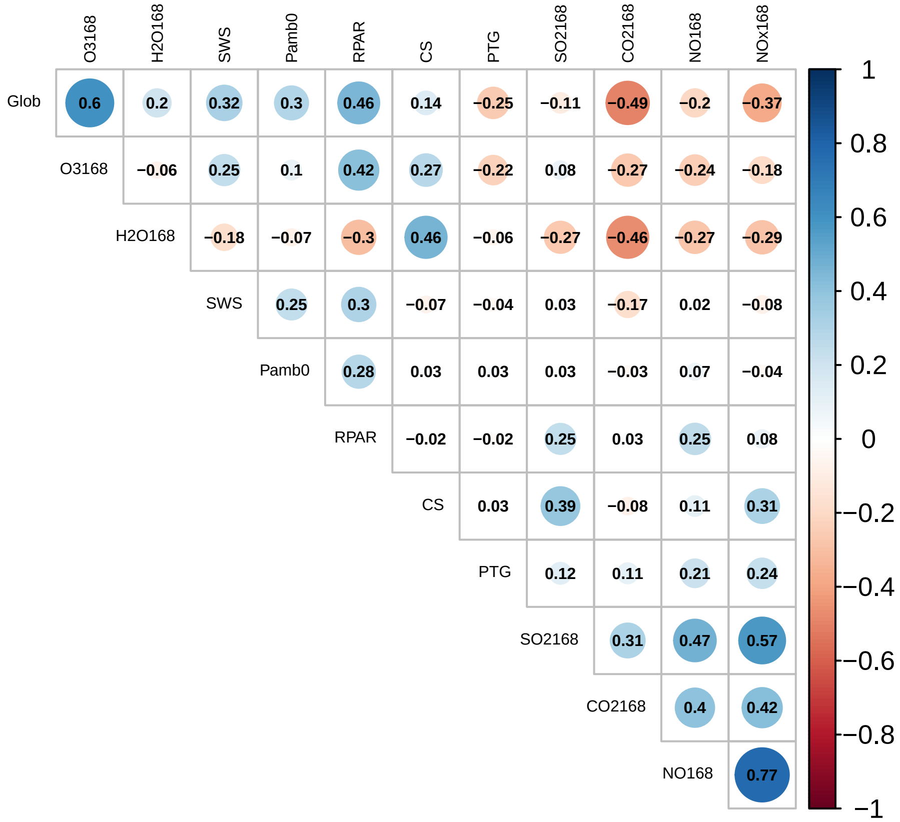


As we can see, there are a lot of correlated variables which can cause problems when using machine learning methods. — I think we should explain more deeply what kind of problems. . .

The variables which have an absolute correlation greater than 0.8 with some other variable is removed from the data. — Loic, can you write above a better description of what is done (I mean, how the code selects which one are left in the data and which one are removed?)

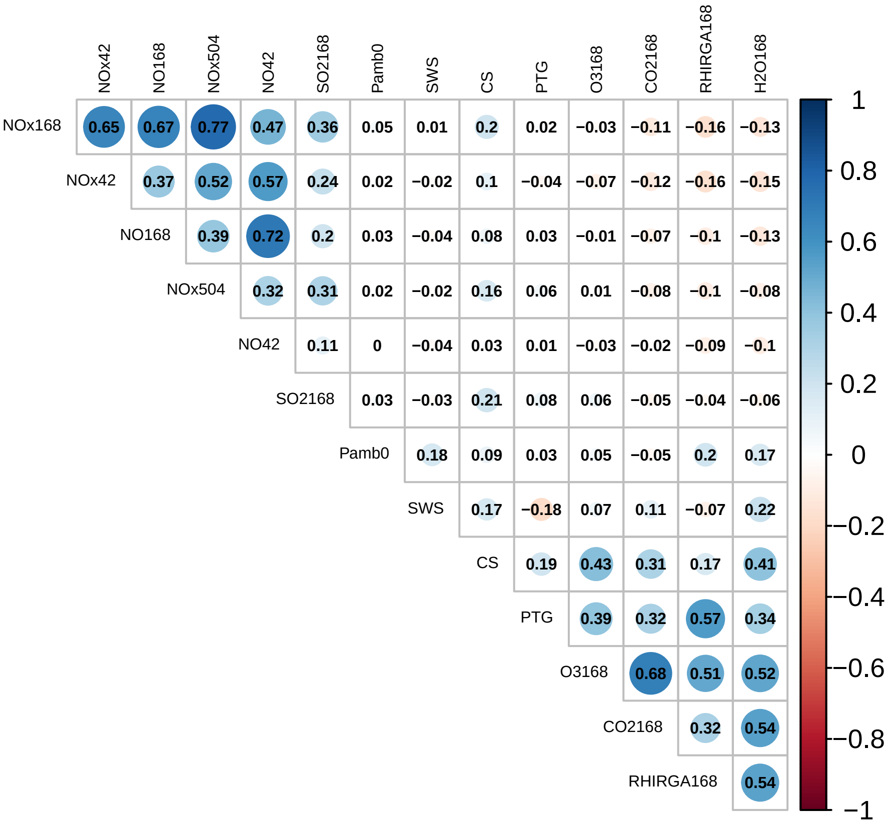
The correlations for mean and standard deviations after cleaning the data are shown in the pictures below as well as a table of variables left in the data (table 2). Table 3 shows that the data includes only 26 variables. — should we put the means and std in the same correlation matrix to see if there are correlation between them and to save space?

Correlation matrix of mean variables for the clean dataset



	O3168	H2O168	SWS	Pamb0	RPAR	CS	PTG	SO2168	CO2168	NO168	NOx168
Glob	0.6	0.2	0.32	0.3	0.46	0.14	−0.25	−0.11	−0.49	−0.2	−0.37
O3168		−0.06	0.25	0.1	0.42	0.27	−0.22	0.08	−0.27	−0.24	−0.18
H2O168			−0.18	−0.07	−0.3	0.46	−0.06	−0.27	−0.46	−0.27	−0.29
SWS				0.25	0.3	−0.07	−0.04	0.03	−0.17	0.02	−0.08
Pamb0					0.28	0.03	0.03	0.03	−0.03	0.07	−0.04
RPAR						−0.02	−0.02	0.25	0.03	0.25	0.08
CS							0.03	0.39	−0.08	0.11	0.31
PTG								0.12	0.11	0.21	0.24
SO2168									0.31	0.47	0.57
CO2168										0.4	0.42
NO168											0.77

Correlation matrix of std variables for the clean dataset



	NOx42	NO168	NOx504	NO42	SO2168	Pamb0	SWS	CS	PTG	O3168	CO2168	RHIRGA168	H2O168
NOx168	0.65	0.67	0.77	0.47	0.36	0.05	0.01	0.2	0.02	−0.03	−0.11	−0.16	−0.13
NOx42		0.37	0.52	0.57	0.24	0.02	−0.02	0.1	−0.04	−0.07	−0.12	−0.16	−0.15
NO168			0.39	0.72	0.2	0.03	−0.04	0.08	0.03	−0.01	−0.07	−0.1	−0.13
NOx504				0.32	0.31	0.02	−0.02	0.16	0.06	0.01	−0.08	−0.1	−0.08
NO42					0.11	0	−0.04	0.03	0.01	−0.03	−0.02	−0.09	−0.1
SO2168						0.03	−0.03	0.21	0.08	0.06	−0.05	−0.04	−0.06
Pamb0							0.18	0.09	0.03	0.05	−0.05	0.2	0.17
SWS								0.17	−0.18	0.07	0.11	−0.07	0.22
CS									0.19	0.43	0.31	0.17	0.41
PTG										0.39	0.32	0.57	0.34
O3168											0.68	0.51	0.52
CO2168												0.32	0.54
RHIRGA168													0.54

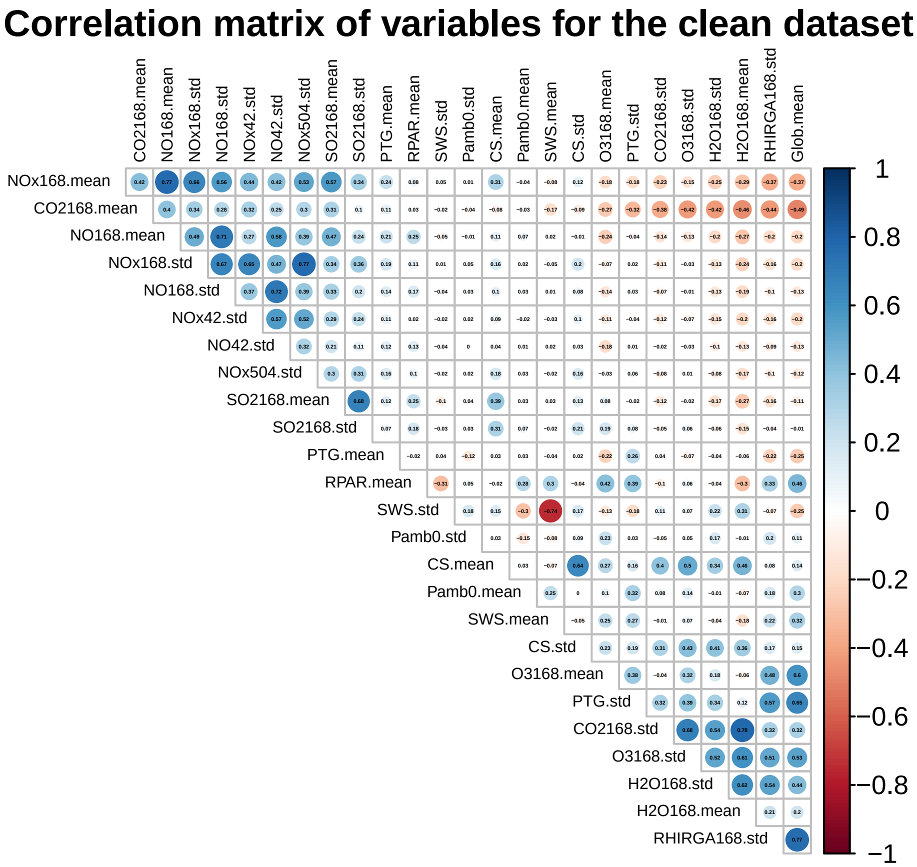


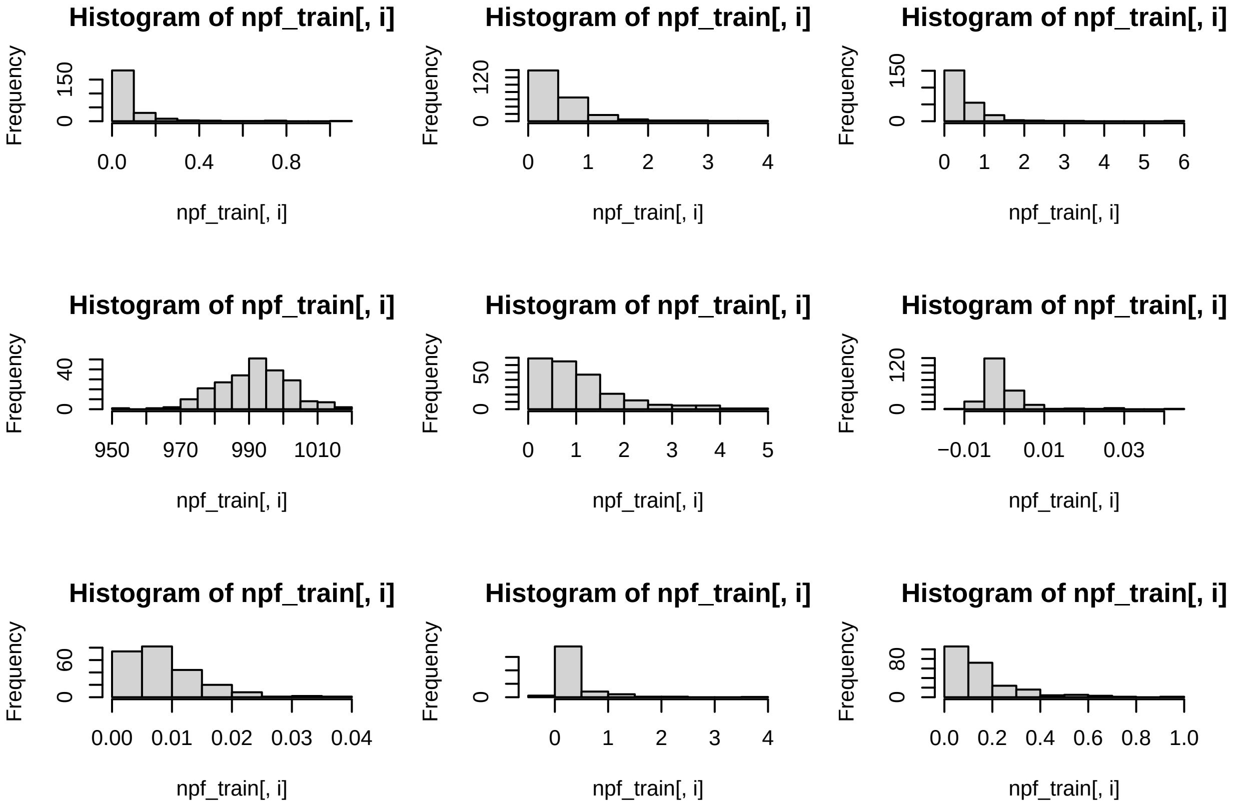
Table 2: List of kept variables

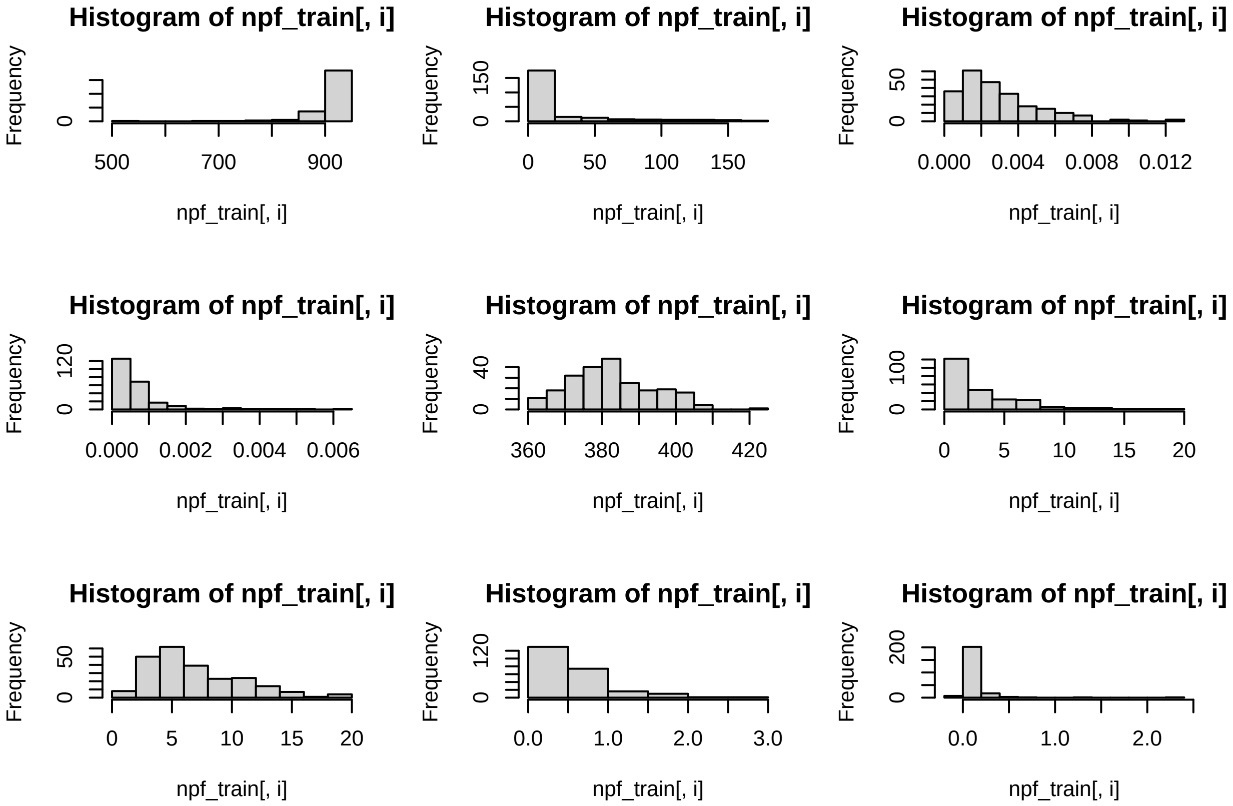
NO42.std	SO2168.mean	CO2168.std	NOx168.std
NOx42.std	SO2168.std	H2O168.mean	O3168.mean
NOx504.std	SWS.mean	H2O168.std	O3168.std
Pamb0.mean	SWS.std	NO168.mean	RHIRGA168.std
Pamb0.std	CS.mean	NO168.std	RPAR.mean
PTG.mean	CS.std	NOx168.mean	Glob.mean
PTG.std	CO2168.mean	NOx168.std	

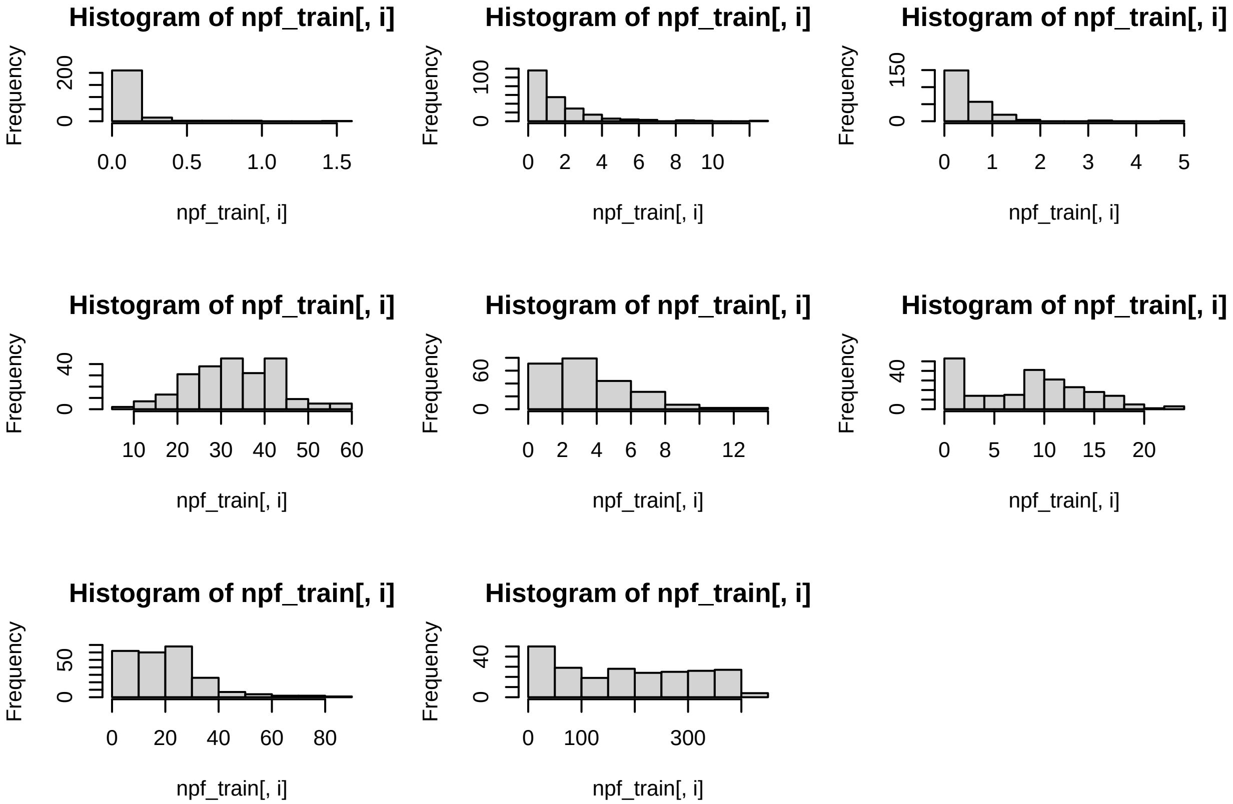
Table 3: Summary of clean dataset

	npf_train	npf_test
Measurements	464	965
Variables	26	26

— I think we should put here some plot or graphical summary of the variables left after cleaning to get an overview of the values (for example to justify scaling in PCA). — Is there any good “summary picture” we could use? I tried histograms but there are too many of them…


Histogram of npf_train[, i]
Frequency
0
150
0.0
0.4
0.8
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
120
0
1
2
3
4
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
150
0
1
2
3
4
5
6
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
40
950
970
990
1010
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
50
0
1
2
3
4
5
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
120
−0.01
0.01
0.03
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
60
0.00
0.01
0.02
0.03
0.04
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
0
1
2
3
4
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
80
0.0
0.2
0.4
0.6
0.8
1.0
npf_train[, i]


Histogram of npf_train[, i]
Frequency
0
500
700
900
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
150
0
50
100
150
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
50
0.000
0.004
0.008
0.012
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
120
0.000
0.002
0.004
0.006
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
40
360
380
400
420
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
100
0
5
10
15
20
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
50
0
5
10
15
20
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
120
0.0
1.0
2.0
3.0
npf_train[, i]
Histogram of npf_train[, i]
Frequency
0
200
0.0
1.0
2.0
npf_train[, i]



Performance measures

To compare different classifiers two measures are used: accuracy and perplexity. Accuracy is the proportion of the observations that has been classified correctly. Perplexity is a rescaled variant of log-likelihood. If perplexity equals to 1 the classifier predicts always the probability of an observation to an actual class. Perplexity of 2 corresponds to coin flipping.

For Random Forest and decision tree perplexity is not calculated but instead ROC curves are looked at. —- Did we agree on this? :) ROC should still be added to code below

If the method predicts a probability of an event, observation is classified as "event" if the estimated posterior probability is more than 0.5. Otherwise the observation is classified as "nonevent".

Performance measures are calculated using validation method and 10-fold Cross-Validation. For validation method the training data is randomly divided into 2 equally large data sets, training data to fit the model and validation data to estimate the accuracy and perplexity.

Investigation of features

To find the most important variables logistic regression with Lasso, decision tree ("basic"), random forest and Principal Component Analysis (PCA) are used to cleaned data. The accuracy of the approaches are also calculated as well as perplexity of logistic regression with Lasso.

1) Logistic regression with Lasso, lambda selected by Cross-Validation:

Logistic regression is a discriminant classifier which assumes that the log odds is linear in variables. When combined with Lasso a subset selection of variables are done by adding so called penalty term to residual sum of squares which is minimized in parameter estimation process. The bigger the estimated coefficients of the variables are the bigger the penalty. The penalty term forces some of the coefficients to zero. The amount of penalty depends on a coefficient called lambda. The value of lambda is selected by cross-validation so that the value of the test error is minimized. The lambda is

```
## [1] 0.009059896
```

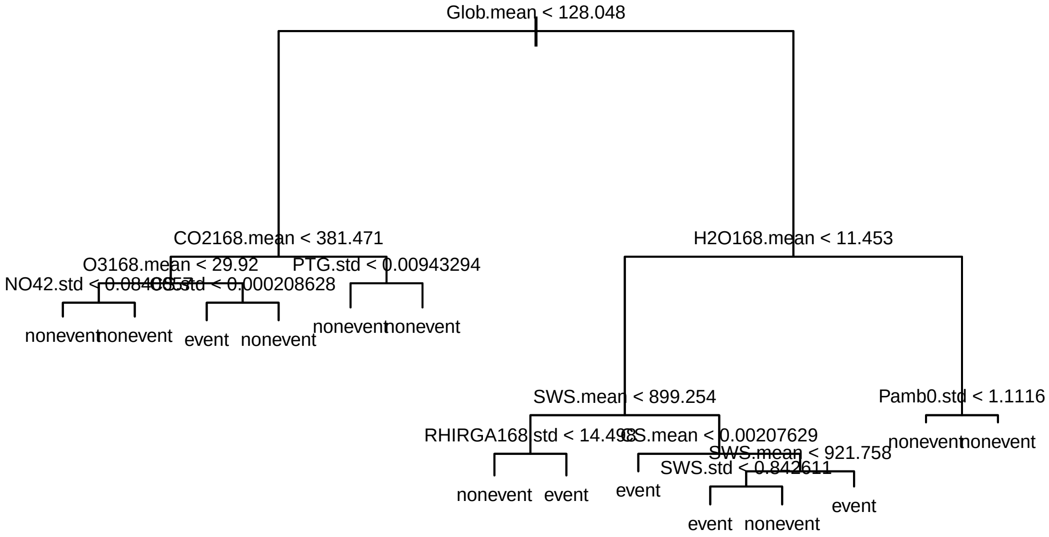
The estimated coefficients are

```
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                          s0
## (Intercept)    5.178482e-01
## NO42.std       7.355450e-02
## NOx42.std      .
## NOx504.std     .
## Pamb0.mean     .
## Pamb0.std      2.284548e-01
## PTG.mean       .
## PTG.std        4.088522e+01
## SO2168.mean    .
## SO2168.std     4.954213e-01
## SWS.mean       1.451465e-02
## SWS.std        .
## CS.mean       -7.942649e+02
## CS.std         3.210184e+02
## CO2168.mean   -4.745895e-02
## CO2168.std     7.771859e-02
## H2O168.mean   -2.422650e-01
## H2O168.std     .
## NO168.mean     .
## NO168.std      2.096218e+00
## NOx168.mean    .
## NOx168.std     .
## O3168.mean     1.449695e-01
## O3168.std      9.487847e-02
## RHIRGA168.std  6.634250e-02
## RPAR.mean      .
## Glob.mean      7.590128e-03
```

As we can see, some of the variables have a zero coefficient meaning that Lasso has done variable selection.

2) A “normal” decision tree selects the following variables with the misclassification as follows

```
##
## Classification tree:
## tree(formula = class2 ~ ., data = npf_train)
## Variables actually used in tree construction:
##  [1] "Glob.mean"     "CO2168.mean"   "O3168.mean"    "NO42.std"
##  [5] "CS.std"        "PTG.std"       "H2O168.mean"   "SWS.mean"
##  [9] "RHIRGA168.std" "CS.mean"       "SWS.std"       "Pamb0.std"
## Number of terminal nodes:  14
## Residual mean deviance:  0.2277 = 49.64 / 218
## Misclassification error rate: 0.05172 = 12 / 232
```



Accuracy of the tree can be calculated from the confusion matrix. Predicted classes vs. actual classes for validation data:

```
##
## tree.pred  nonevent event
##   nonevent      104    21
##   event          13    94
```

Predicted classes vs. actual classes for training data: — this can actually be seen from the results above (summary(tree.npf)) - do we want to leave this confusion matrix away?

```
##
## tree.pred_train nonevent event
##        nonevent      113    10
##        event           2   107
```

The accuracies are respectively

```
## [1] 0.948
```

```
## [1] 0.853
```

3) Random Forest with 5 variables (square root of the number of features) used in each run

Confusion matrix for training data

```
##
## rf.pred_train nonevent event
##      nonevent      115     0
##      event           0   117
```

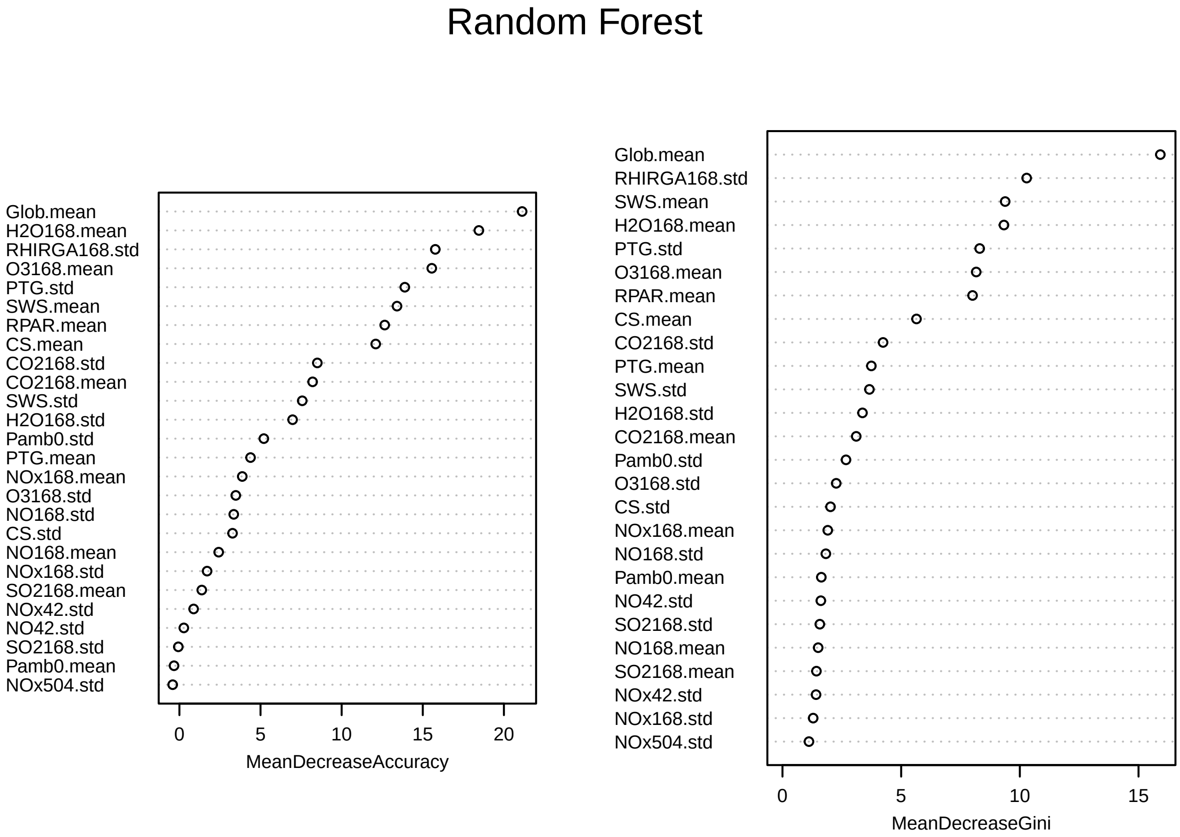
Confusion matrix for validation

```
##
## rf.pred    nonevent event
##   nonevent      101    20
##   event          16    95
```

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

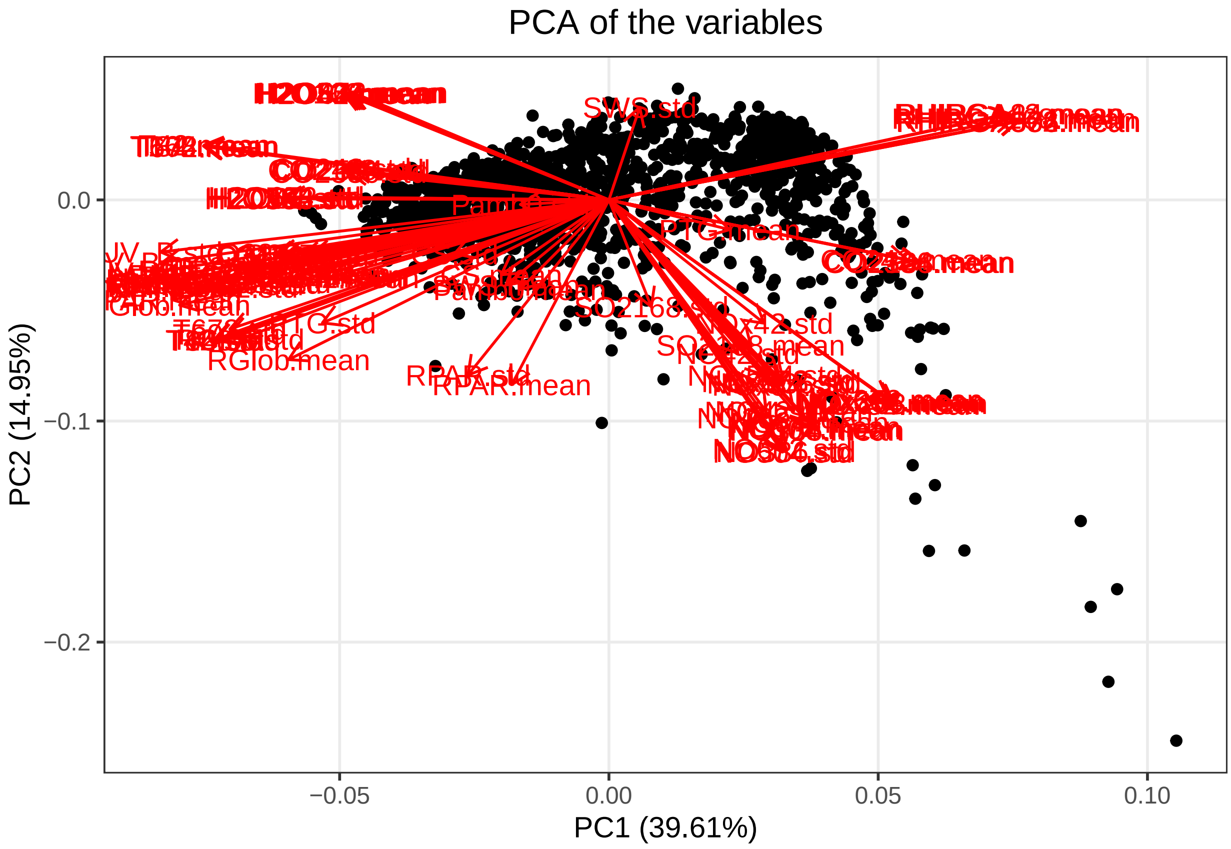
```
## [1] 0.8534483
```

The importance of the variables are:



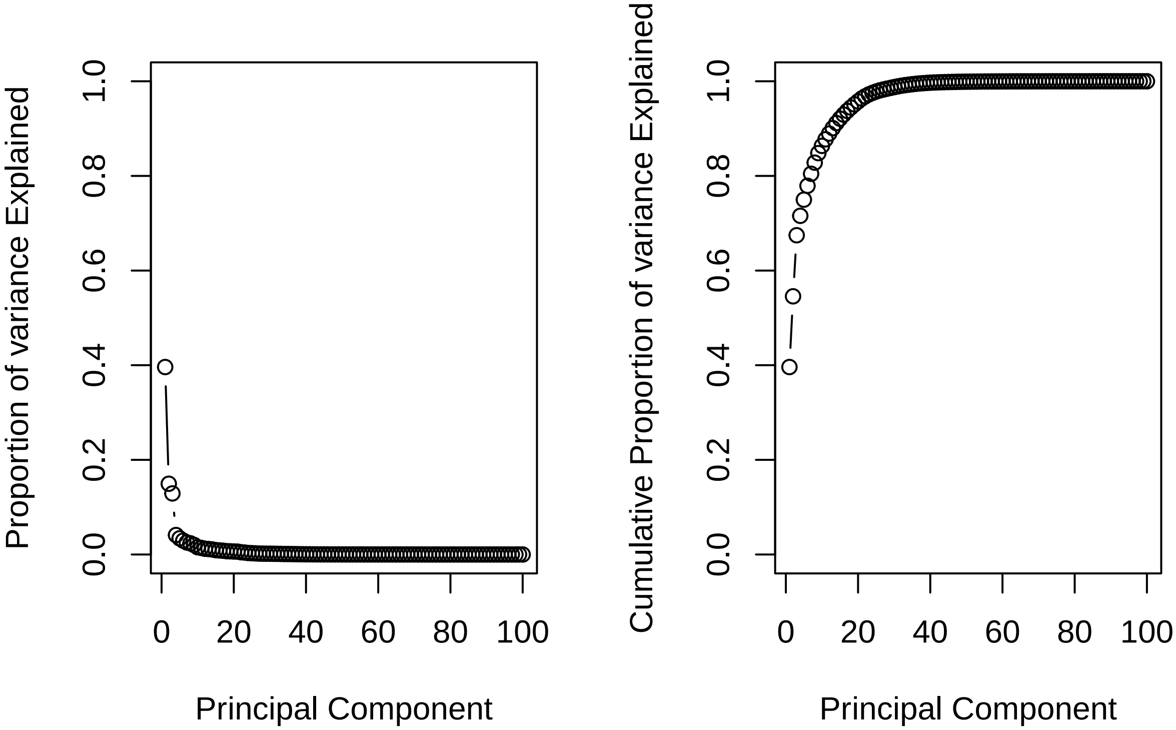
```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
```

4) PCA PCA is used to the original training data (npf_train.csv with 464 observations) together with the original testing data (npf_hidden.csv) where the variables are centered to have zero mean and scaled to have standard deviation one. The responses variables are removed from the original training data because we are using unsupervised learning method. The first two principal components are:



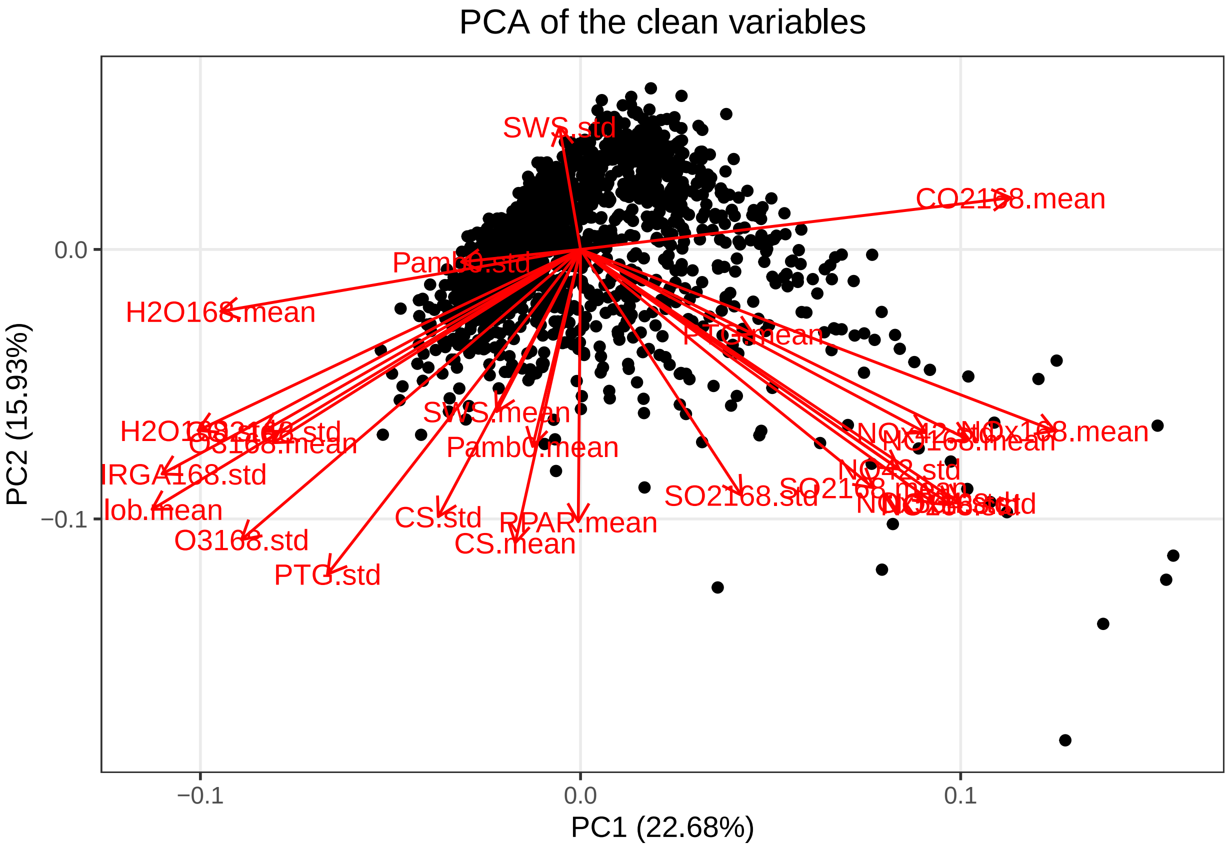
—Can we conclude from the picture above if there are any significant outliers?

The proportion of variance explained (PVE) by each principal component and the cumulative PVE is shown

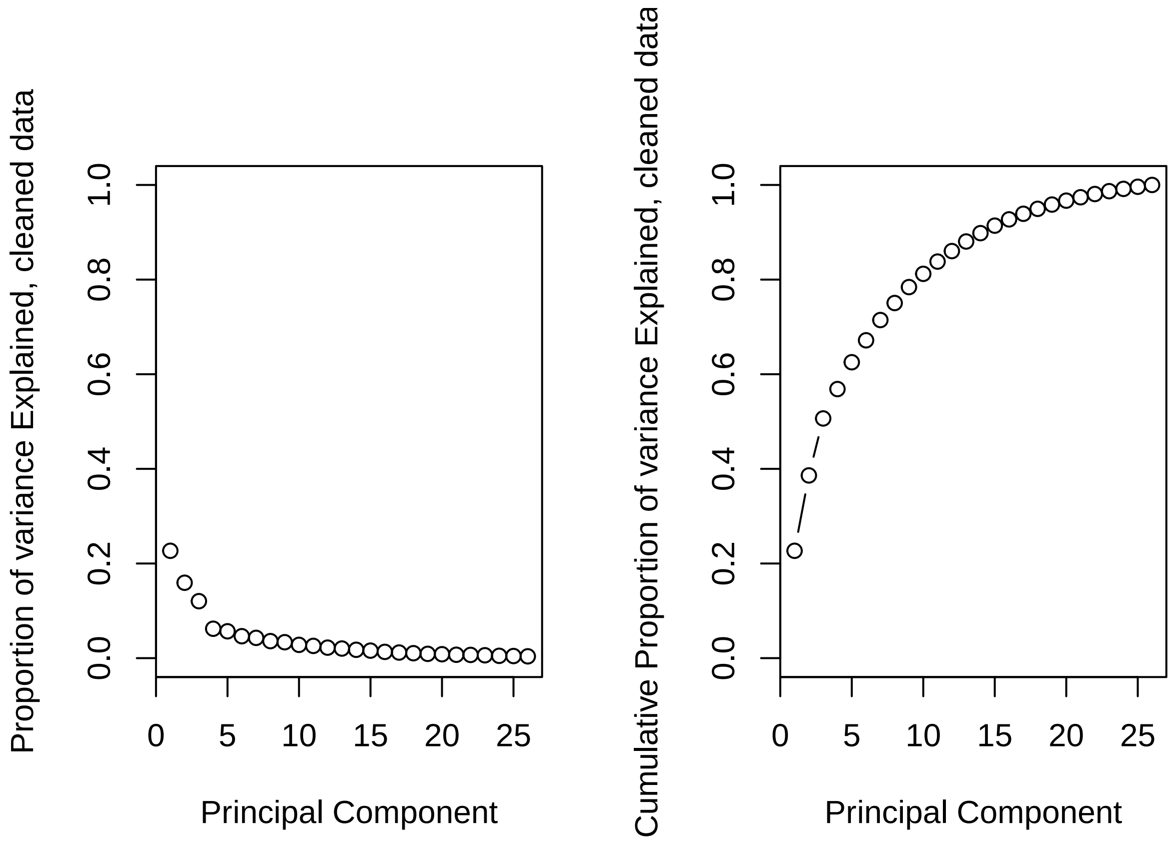


in the figures below:

PCA is also done for the data where variables with high correleation to other variables are removed (cleaned data). The first two principal components are:



The proportion of variance explained (PVE) by each principal component and the cumulative PVE is shown in the figures below:



According to the results above at least 15 first principal components should be used to explaina bout 90 % of the variance. — If you can conclude something else from the results, you can add some text :)

The performance of the classifiers investigated so far are:

```
##             Model Train Accuracy Validation Accuracy CV Accuracy
## 1 Log reg Lasso CV          0.918               0.836       0.886
## 2             Tree          0.853               0.948
## 3    Random Forest              1               0.853
##   Train Perplexity Validation Perplexity CV Perplexity
## 1             1.24                 1.348         1.311
## 2
## 3
```

Conclusions and feature selection

— !!! This whole part can be removed if it's not needed. The text is not updated after cleaning the data so at least the conclusions should be updated. The models in the next session is set to use cleaned data instead of the selection previously done here. !!!

The accuracy is best for Random Forest, but also Logistic Regression with Lasso where lambda is selected using CV ("log reg CV") performs very well. —- Should we investigate diagnostic plots for log reg? Should be give weight to well performed classifiers when selecting the variables?

All models use the following variables - RHIRGA: mean is more important than std according to tree and RF - H2O mean - O3 mean (Logistic Regression with Lasso, lambda = CV ("log reg CV") gives small value to std) - CO2: tree and RF uses only std, log regs give value to mean too - T std - SWS.mean

The following variables are used in all other models but RF - NO.std - Pamb0.mean or .std

The following variables are used in all other models but "normal" tree - CS.mean (.std only in log reg CV)

These variables are used in some models - UV_B.std is used only on log reg CV - UV_A.mean is used only in RF - RGlob.mean and RGlob.std (RF, mean also in log reg 0.1) - PTG.std (tree, RF) - NET.mean - RPAR.mean - SO2

Because the measures in different heights/levels are highly correlated, we select just one of them. —- which ones? Should we put more weight to the selections in random forest and log reg CV? —- Should be produce boxplots, histograms and/or scatterplots to selected variables

We select the following variables to be used in models: —- selected for testing the same variables than in Log Reg Lasso CV but only once if there are multiple values with different heights ————

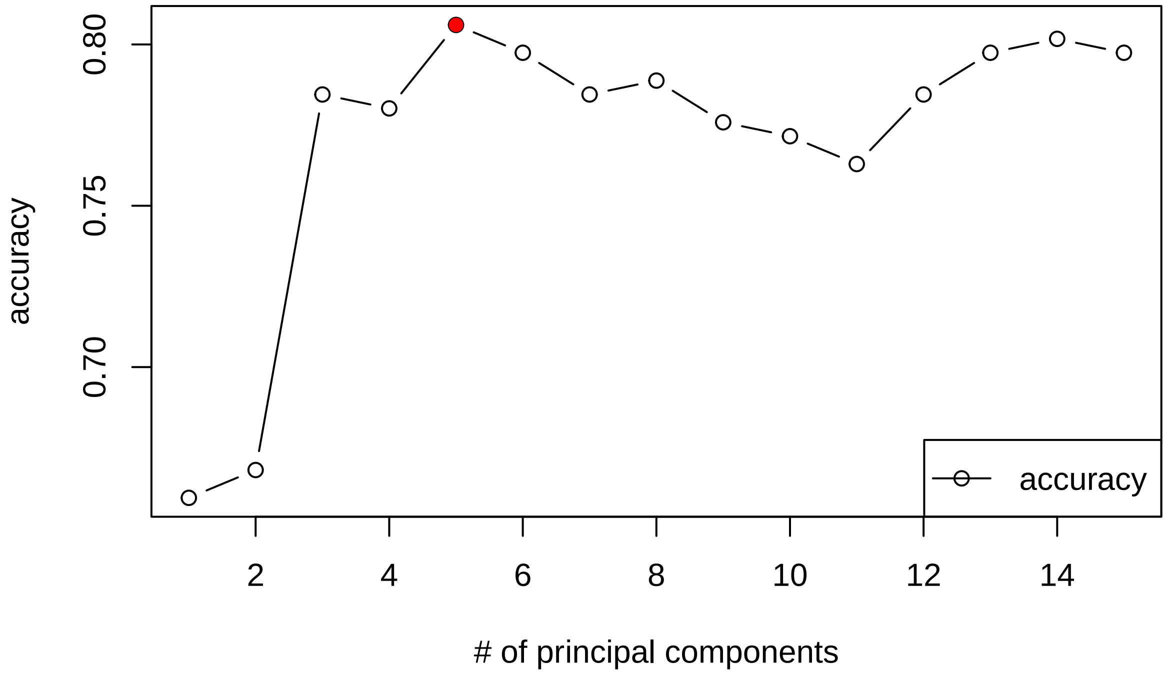
Let's check if there are any correlation left:

Model selection

The models tested in addition to the ones already tested are

1) Dummy: —- write a description of the method and the code itself…
2) Naive Bayes

Accuracy of Naive Bayes when using the first 15 principal components to reduce the dimensionality of the data. The highest accuracy is marked with red.



The highest accuracy is received when using 5 first PCs so this is added to be one of the possible methods.

3) Logistic regression — with or without interactions? now it's without

4) k-NN K nearest neighbor is tried with different values of k: 1, 5, 10, 15, 20 and 50. Accuracy on validation set for each of them are respectively

```
## [1] 0.780 0.836 0.836 0.828 0.823 0.780
```

When k = 5 (or 10), the accuracy is highest so 5-NN is added

Performance measures for the methods are:

```
##                  Model Train Accuracy Validation Accuracy CV Accuracy
## 1     Log reg Lasso CV          0.918               0.836       0.886
## 2                 Tree          0.853               0.948            
## 3        Random Forest              1               0.853            
## 4          Naive Bayes          0.853               0.746       0.754
## 5 Naive Bayes with PCA          0.875               0.806       0.817
## 6  Logistic regression           0.94               0.866       0.877
## 7                 5-NN          0.892               0.836         0.5
##   Train Perplexity Validation Perplexity CV Perplexity
## 1             1.24                 1.348         1.311
## 2                                                     
## 3                                                     
## 4            2.106                   Inf         9.353
## 5             1.46                 1.471         1.628
## 6            1.184                 1.411          1.37
## 7                                                     
```