Exoplanet Hunting

Grzegorz Wierzchowski

2019-07-24

License

Prerequisites

R Packages

The document is prepared in **R Markdown** language and should be compiled using the **rmarkdown** package. This package can be installed from CRAN:

```
install.packages("rmarkdown")
```

To create this report I used following programs and R packages:

- R Base - version 3.6.1
- `tidyverse` - version 1.2.1
- `gridExtra` - version 2.3
- `h2o` - version 3.22.1.1
- R Studio
- Java - Oracle version "9.0.4"
- Linux based operating system

If some package is not installed on the system, report's build script will download and install it. Some packages require C compiler to be installed on system, `h2o` package require Java in version 8 or higher to be installed.

H2O

In my work I wanted to use machine learning. Having bad experiece from previous MovieLens project, where some of my calculations were running for several hours using only 1 CPU core, and having very limited time for this project (about one week), I decided to try package which utilizes multithreading. One of the world leading frameworks for machine learning is *H2O*: https://www.h2o.ai/. This framework is writen in Java and there is tiny wrapper for R language available as `h2o` package. It works in following way. User from his R code loads the package and initalizes engine by invoking `h2o.init()` call. This either starts on local computer *H2O* server or with proper configuration may connect to server working on remote machine. Then user using *H2O* methods can load data or perform operations on it including tidying data, model creation, training and validation. Every such operation is being done at server side (by multithreaded Java code) and computer resources are taken at server side. This means that if all this is run on one box, it must have enough RAM memory to support both *H2O* server and R local script. I have run it on my personal laptop with 8 CPU cores and 16GB RAM - and it was quite sufficient for the task.

When *H2O* server is running, it is possible to connect to it from web browser and inspect objects, lookup job statuses, or even perform machine learning operations entirely from GUI, and at the end also export selected

model as Java sorce code possible to deploy on other systems. During this project I however did not used any of those operations, doing everything completly in R.

Data

I decided to use one of the data files published on *Kaggle* web service. Data are described and can be downloaded from following page: https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data. It is required to create free personal account on *Kaggle* service before it allows you to download data. I'm not authorized to re-distribute those data. So if somebody wishes to run the code enbedded into this document, he should download the data files by himself. Data are available for download as zip-file which containes two files: `exoTrain.csv` and `exoTest.csv`. As names suggest they store training and test data and are in CSV format. My program assumes that they are placed in `Data` subfolder of folder where this document source (i.e. .Rmd file) is stored.

Introduction

Problem description

The goal of the project is to try programatically recognize stars which could potentially have one or more exoplanets. Here is extracted description from *Kaggle*:

> The data describe the change in flux (light intensity) of several thousand stars. Each star has a binary label of 2 or 1. 2 indicated that that the star is confirmed to have at least one exoplanet in orbit; some observations are in fact multi-planet systems.
>
> As you can imagine, planets themselves do not emit light, but the stars that they orbit do. If said star is watched over several months or years, there may be a regular 'dimming' of the flux (the light intensity). This is evidence that there may be an orbiting body around the star; such a star could be considered to be a 'candidate' system. Further study of our candidate system, for example by a satellite that captures light at a different wavelength, could solidify the belief that the candidate can in fact be 'confirmed'.

Model Evaluation Metrics

Evaluation metric measures how good is particular model in estimating results. There are many kinds of them in use. We have to select which one is most important for us, and which is most appropriate for our problem. Here we predict if particular star may have an orbiting exoplanet or not. This is so called binary clasification problem. Clasification - because our result belongs to fixed set of values. Binary - because we can have only two values of result. Additional characteristic of our problem is very big disproportion of one result over other in data (called *prevalence* in course materials) due to fact that that only small fraction of stars are suspected to have planets.

I have reviewed different metrics and found out that most recommended for such case is metric called MCC (Matthews correlation coefficient). Quoting Wikipedia:

> The MCC is in essence a correlation coefficient between the observed and predicted binary classifications; it returns a value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 no better than random prediction and -1 indicates total disagreement between prediction and observation.

It can be calculated from confusion matrix using formula (abbreviations as in course material):

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Meaning "star has exoplanet" as positive I will be also paying attention to *Precision*. I imagine that a stream of observations is being at first exposed to "Exoplanet Hunting" models, and then stars appointed by models are further investigated. This investigation engages time of scientists, devices, power etc. so is costfull. Therefore it would be money lost if model would predict too much false positives. Precision can be calculaed as:

$$\text{PPV} = \frac{TP}{TP + FP}$$

Symbols

In this report I will use following symbols:

- μ - mean of star flux observations
- σ - standard deviation of star flux observations

Work plan

As noted earlier I had very limited time for this project, so I did not spent too much time on searching how others approach the problem. I started to work rigth away with the data. From problem nature I knew that exoplanet should manifest as periodical and short flux level fluctuations (peaks to down). Here I specify my approach:

- load the data and review the graphs that shows measured flux levels; check both stars labeled as having and not having exoplanets
- try to find any pattern in exoplanet charts
- try to matematically describe this pattern, and code respective additional predictors.
- try to use machine learning algorithms to transform those predictors into clasification result
- select algorithm that up-front (with default parameters) produces best results and try to further tune it changing some parameters or using cross-validation to estimate them; use ensembling if there are few best models.

Data contain large number of columns (predictors). And those columns represent time-series observations, so I did not meant that some of them are generally more important then others or some relate on others. So I did not considered that any kind of processes like factorization, dimension reduction etc. would be approprate here. Also because of number of columns I did not think that using them directly as imputs for machine learning might give any good results.

Data Exploration

List of parameters used in predictors calculation process with assigned values:

```
# Parameters that affect document flow, performance etc.
H2O_NTHREADS <- -1        # (int) Number of threads for H2O Server (-1 - auto from OS)
H2O_MEM <- "12G"          # (text) Memory for H2O Server
WORK_WITH <- 1.0          # (float) Which fraction of data to work with
                          #   (1.0 = work with full training data + load test data)
RANDOM_SAMPLES <- T       # (T/F) Set this to TRUE if you want to see different flux graphs
                          #   at every time you generate document
SAMPLES_NO <- 18          # (int) Number of samples to present in document
SKIP_CALC <- T            # (T/F) Set this if you previously run calculations and saved them in file
                          #   .RData files needs to be manually placed in RData subfolder
ML_ALGORITHM_TEST_SIZE <- 1000 # We will check different ML algorithms with this size sample
```

```
# Parameters that affect model, calculations, etc.
OUTLINER_THRES_HI <- 5.0 # (float) Outliner if higher then OUTLINER_THRES_HI*sd
OUTLINER_THRES_LO <- 5.0 # (float) Outliner if lower then OUTLINER_THRES_LO*sd
NORMAL_THRES <- 1.7      # (float) Planet suspition if below NORMAL_THRES*sd
LEVELS_NO <- 4           # (int) We create predictors for this number of levels
                         #   from NORMAL_THRES to OUTLINER_THRES_LO
DISPLAY_LEVEL <- 1       # (int) Prepare charts for this level (1=NORMAL_THRES)
WINDOW_H <- 50           # (int) Minimal distance between fluxes minima to consider them as different
TEETH_DIST_ROUND <- 12   # (int) Used to round distances between teeths to group values
# Should be:
#   0 < WORK_WITH <= 1.0
#   0 < NORMAL_THRES <= OUTLINER_THRES_LO
#   LEVELS_NO >= 1
#   1 <= DISPLAY_LEVEL <= LEVELS_NO
#   WINDOW_H > 1
#   TEETH_DIST_ROUND > 0 (1=no rounding)
#   ML_ALGORITHM_TEST_SIZE <= 5087 (no of records in training data)
```

Data retrieval and preparation

To prevent this document expanding to large size I include here very little code listings. Full program code is avaialble in source file of this document (the .Rmd file).

Data were loaded using `h2o.importFile()` function which transformed it into `H2OFrame` object. It is possible to operate directly on such object, but I found it very slow when it comes to setting particular data inside records (probaly bacause of network message exchange associated to every such operation). Such H2O objects are rather intended to running bulk calculation operations at once which could be done entirely at H2O server side (like model training). So I then converted `H2OFrame` object into regular R `data.frame`, and performed operations on it using "well known" `dplyr` package. Once I performed all calculations and created new set of predictors I then transformed new `data.frame` object to `H2OFrame` and used this as input for creating and training models.

Basic characteristic of data

Structure of our input, raw data:

```
h2o.str(exoTrainData_ext, list.len = 5)
```

```
## Class 'H2OFrame' <environment: 0x3902c98>
##  - attr(*, "op")= chr ":="
##  - attr(*, "eval")= logi TRUE
##  - attr(*, "id")= chr "RTMP_sid_990b_6"
##  - attr(*, "nrow")= int 5087
##  - attr(*, "ncol")= int 3198
##  - attr(*, "types")=List of 3198
##   ..$ : chr "enum"
##   ..$ : chr "real"
##   ..$ : chr "real"
##   ..$ : chr "real"
##   ..$ : chr "real"
##   .. [list output truncated]
##  - attr(*, "data")='data.frame': 10 obs. of  3198 variables:
##   ..$ LABEL    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
```

```
##   ..$ FLUX.1   : num  93.8 -38.9 532.6 326.5 -1107.2 ...
##   ..$ FLUX.2   : num  83.8 -33.8 535.9 347.4 -1112.6 ...
##   ..$ FLUX.3   : num  20.1 -58.5 513.7 302.4 -1119 ...
##   ..$ FLUX.4   : num  -27 -40.1 496.9 298.1 -1095.1 ...
##   .. [list output truncated]
```

Notes:

- `LABEL` is our result data; I did some pre-processing here and changed label 2 into 0, so now "0" stands for "is exoplanet", 1" - "no exoplanet"; this was done to force H2O models treat "exoplanet" as positive result.
- all other fields: `FLUX.*` are predictors - i.e. measured flux levels
- I did not investigated if they were pre-processed somehow, or why there are negative values, etc. My feeling was that absolute values does not matter that much, instead important is how they are changing - i.e. the shape of flux graph
- for this reason I also decided to not normalize them
- information: "10 obs. of 3198 variables" is little confusing, it is some kind of H2O cached value, real muber of observations is given in "attr(*, "nrow")"

Check if there are invalid observations inside the data:

```
sum(is.na(exoTrainData_ext_df))
```

```
## [1] 0
```

Proportion of positive vs. negative results:

```
sum(exoTrainData_ext_df$LABEL == 0) / sum(exoTrainData_ext_df$LABEL == 1)
```

```
## [1] 0.007326733
```

Initial overview of data

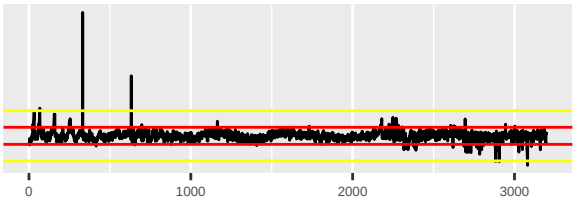
Now I wanted to see how the data looks like. Below I present samples from flux values of stars of each category: without and with exoplanets. 18 stars of each category was randomly selected from training data set.

Actually I have done those charts several times, each time selecting different stars. I noticed that there are some quite rate observations that are unordinary high or low. I treated them as outliners possibly caused by some measure errors. I decided to remove them from data - but only those very high (i.e. bigger than 5 times standard deviation), because I was worying that I can accidentally remove important "down peaks" which may designate a present exoplanet. I replaced outliner value with overall mean value. Better approach would be to take a local mean, but I did not do it for performance reasons, and also because - it does not really matter with this number of observations.

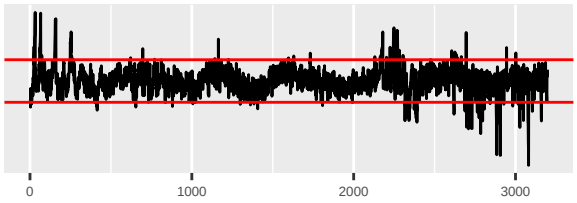
Charts at left side present data before outliners removal, at right side - after. Yellow lines mark outliners cut-off level: $\mu \pm 5\sigma$. Red lines mark level that I will use later as cut-off for creating new predictors: $\mu \pm 1.7\sigma$

No Exoplanet

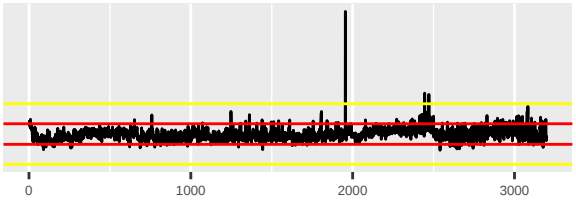
NEx # 1498



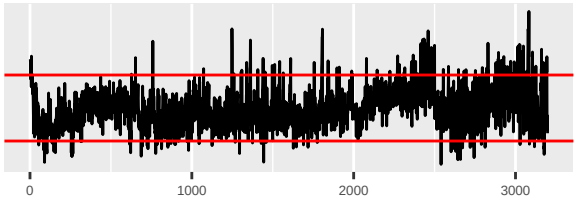
Cleaned



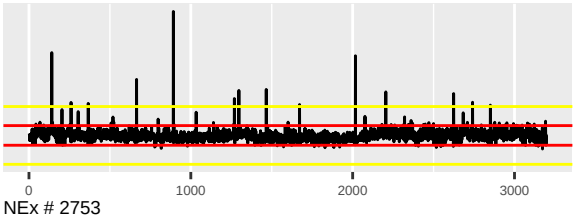
NEx # 2064



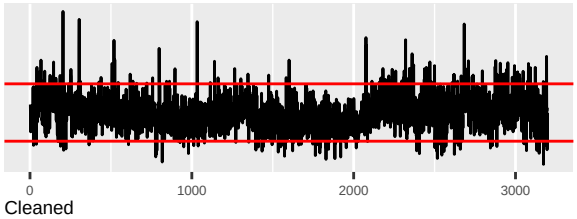
Cleaned



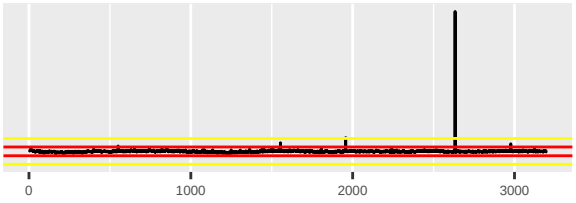
NEx # 3994



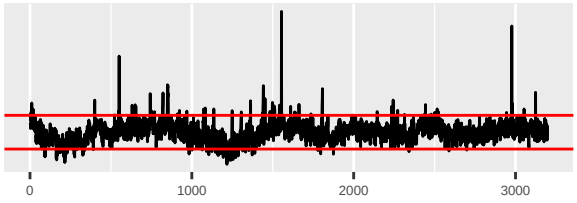
Cleaned



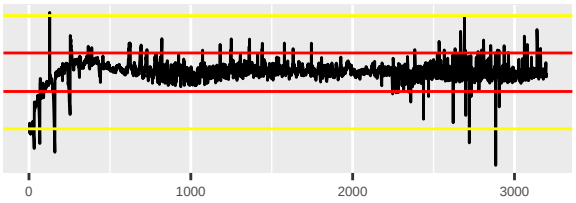
NEx # 2753



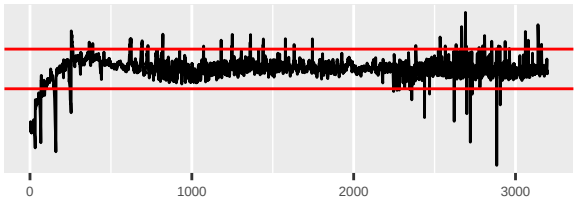
Cleaned



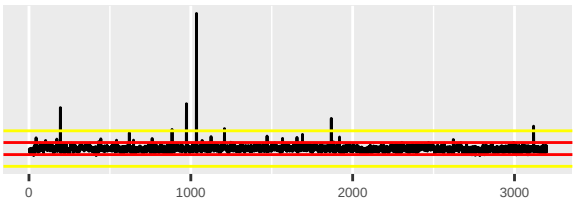
NEx # 882



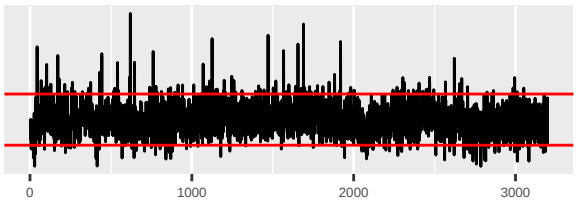
Cleaned



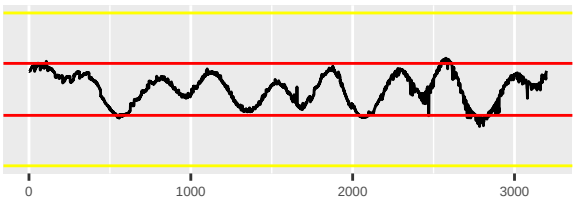
NEx # 3616



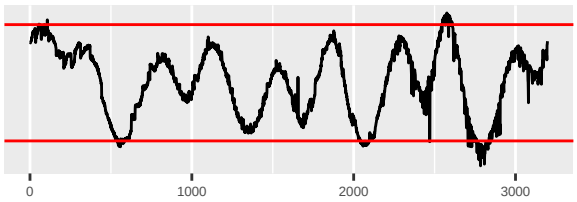
Cleaned



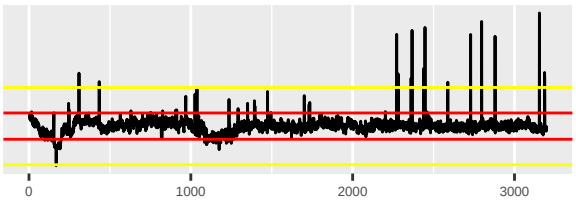
NEx # 3431



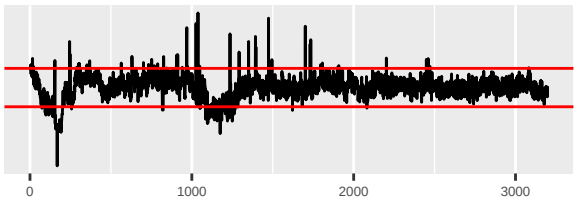
Cleaned



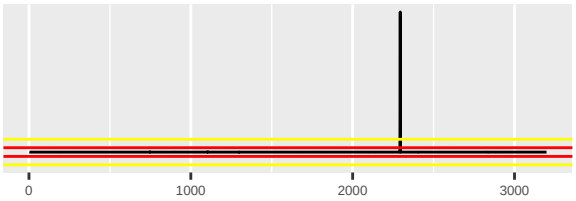
NEx # 2796



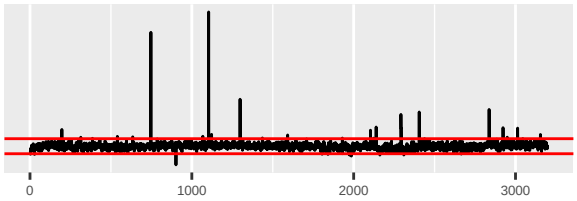
Cleaned



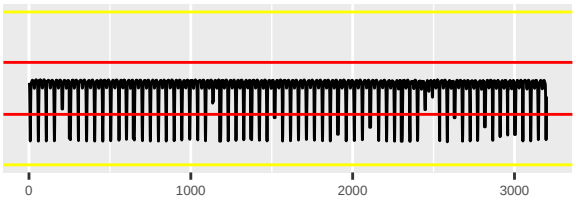
NEx # 2263



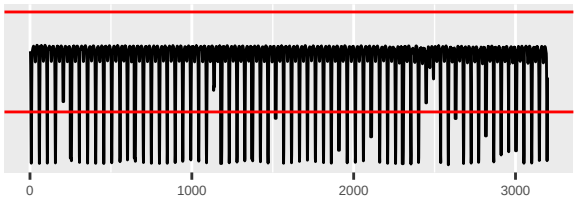
Cleaned



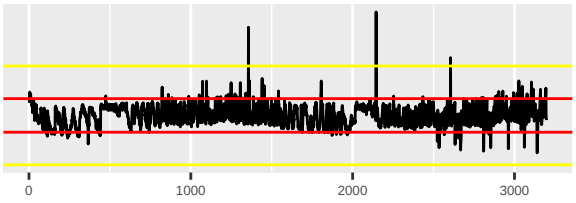
NEx # 2374



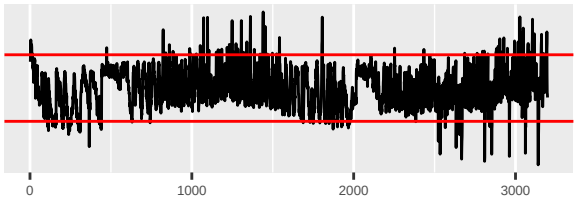
Cleaned



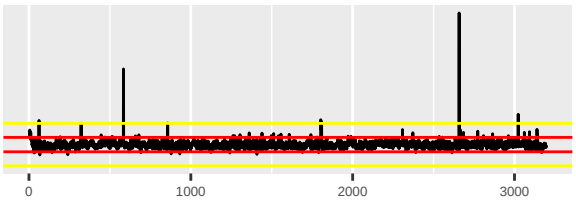
NEx # 4605



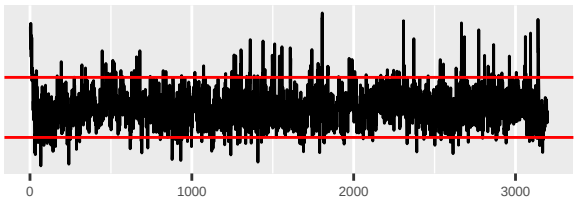
Cleaned



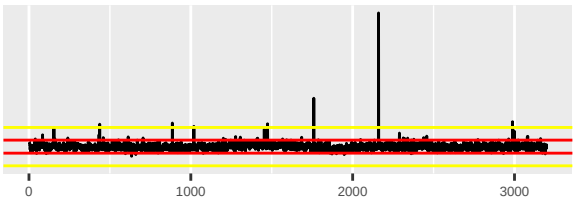
NEx # 77



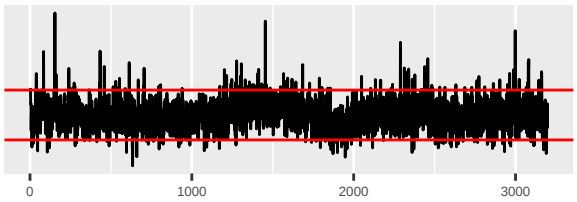
Cleaned



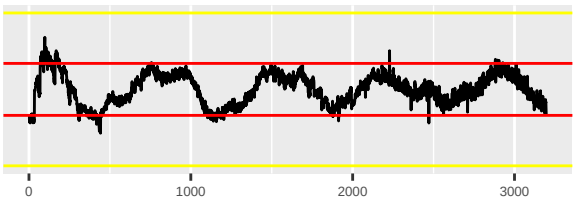
NEx # 5018



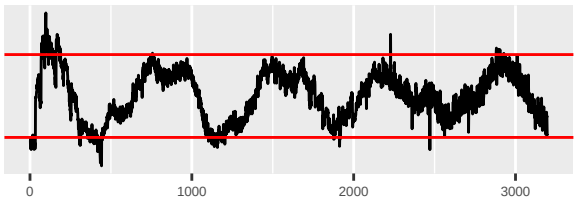
Cleaned



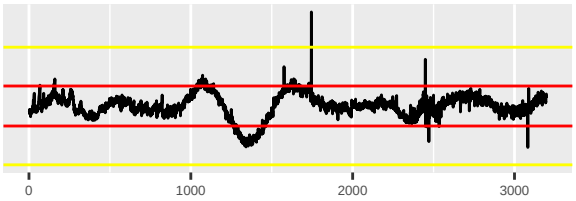
NEx # 2408



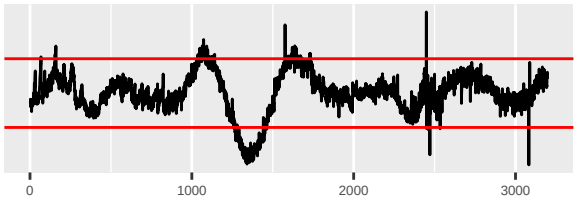
Cleaned



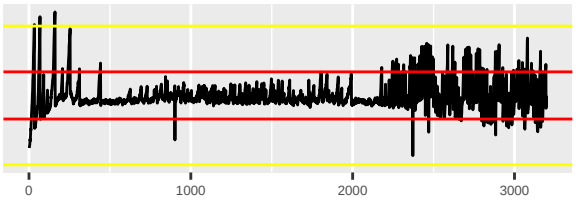
NEx # 1806



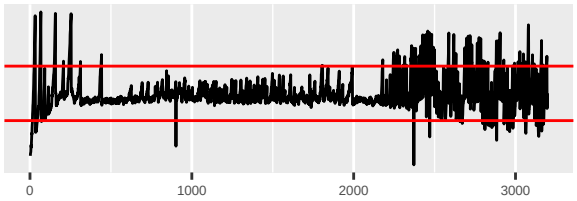
Cleaned



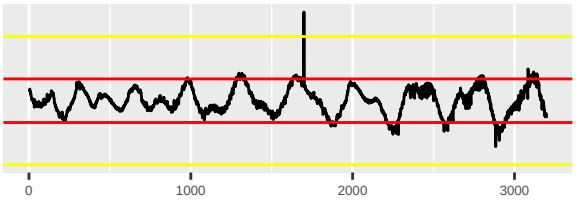
NEx # 2180



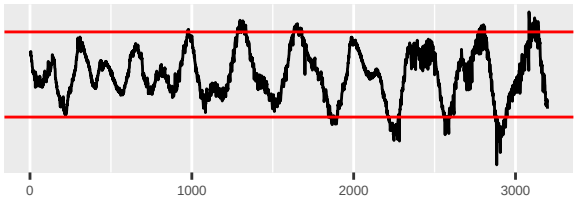
Cleaned



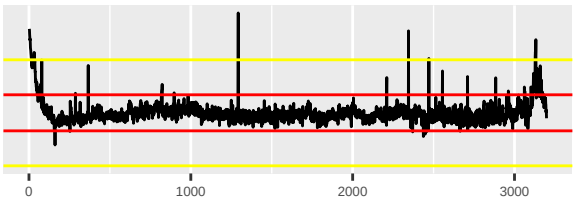
NEx # 3722



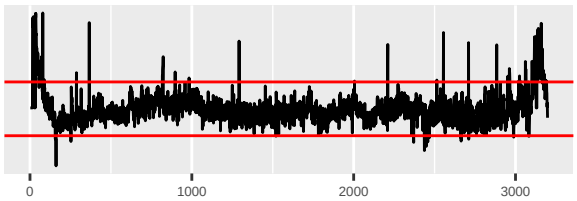
Cleaned



NEx # 4863

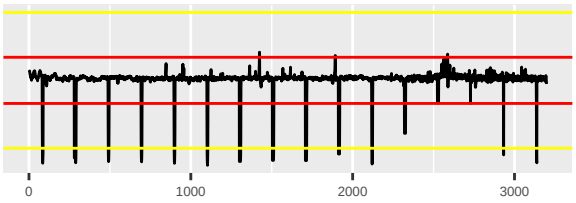


Cleaned

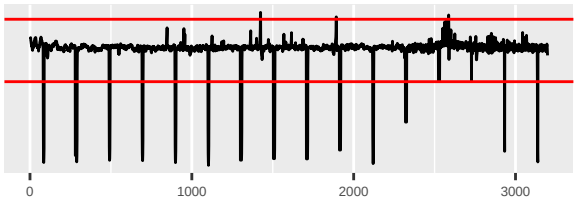


With Exoplanet

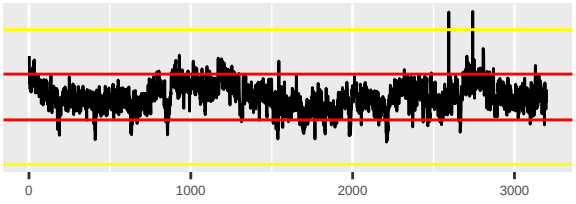
Ex # 8



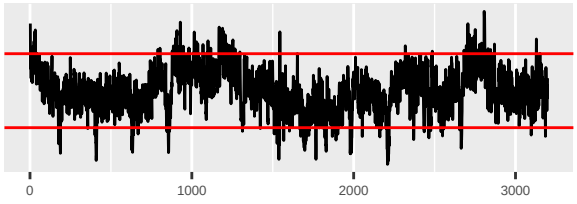
Cleaned



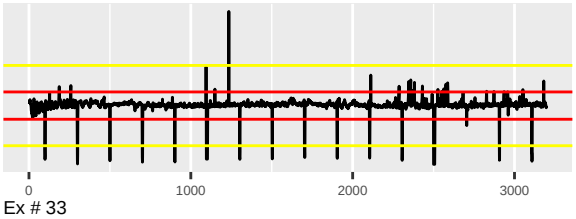
Ex # 20



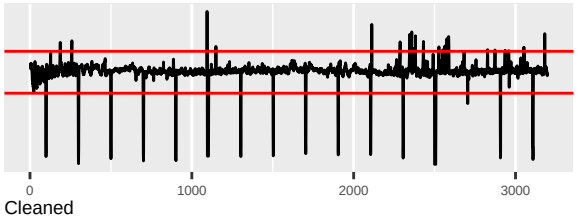
Cleaned



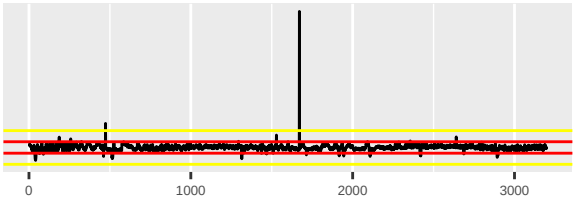
Ex # 36



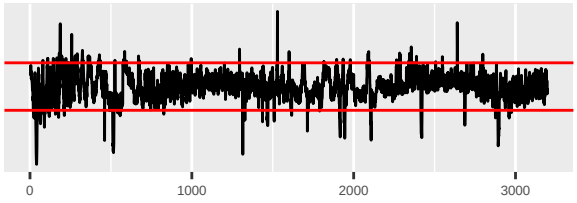
Cleaned



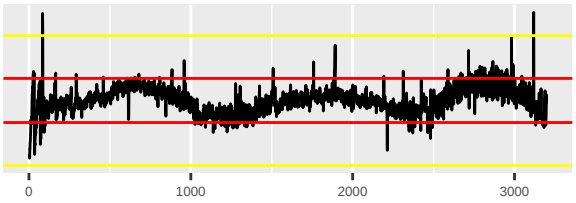
Ex # 33



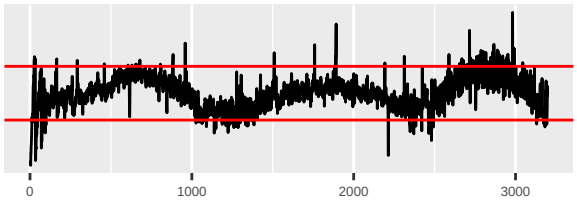
Cleaned



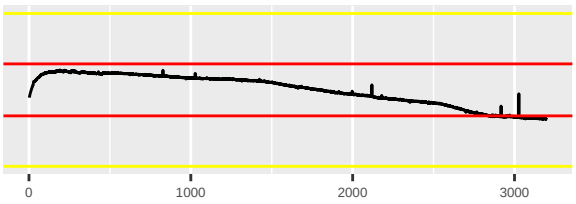
Ex # 17



Cleaned

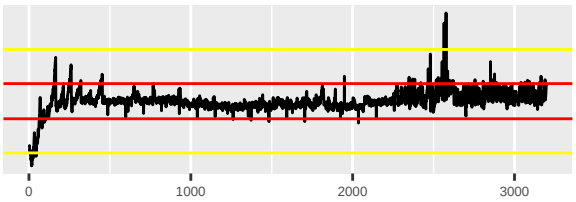


Ex # 15

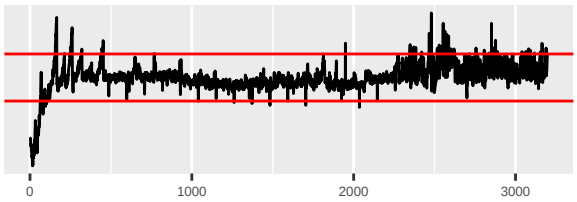


Cleaned

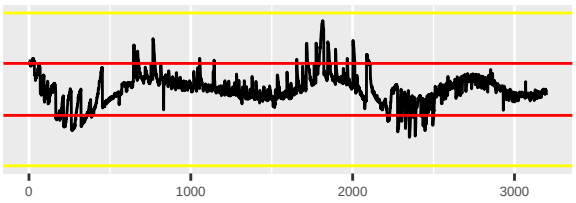
Ex # 21



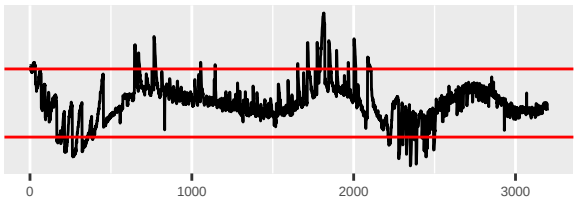
Cleaned



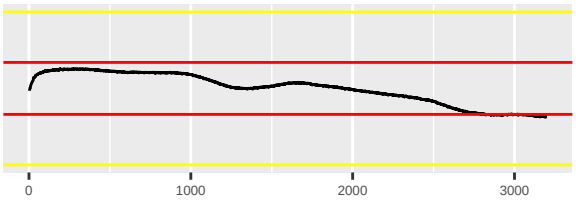
Ex # 3



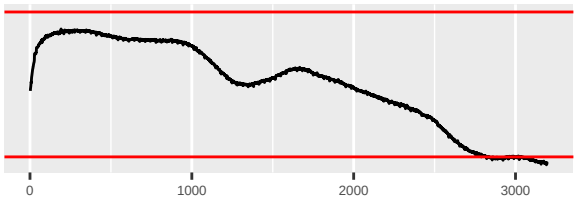
Cleaned



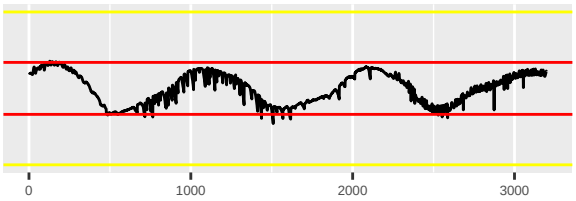
Ex # 31



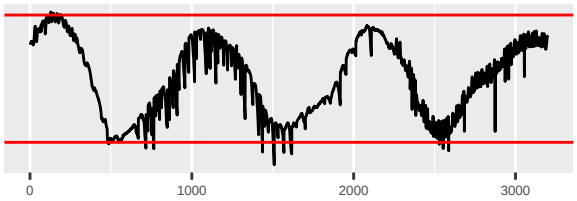
Cleaned



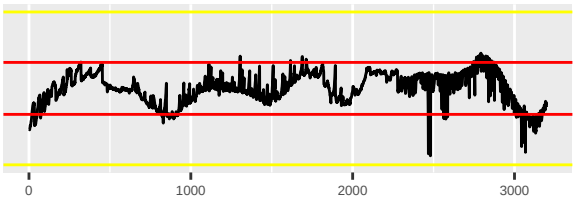
Ex # 22



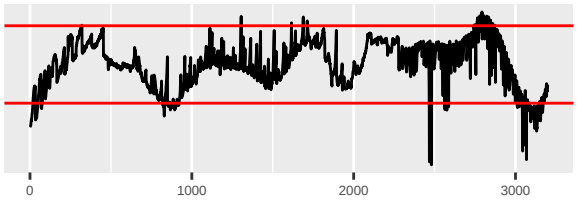
Cleaned



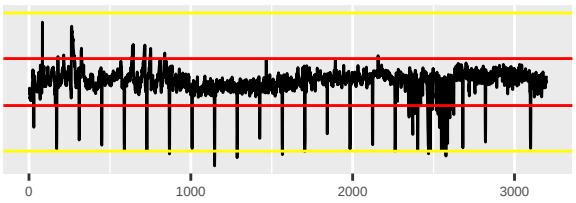
Ex # 5



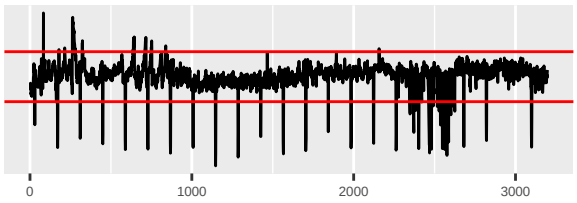
Cleaned



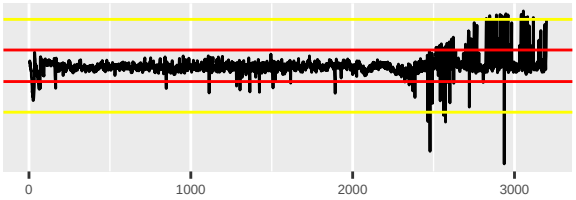
Ex # 2



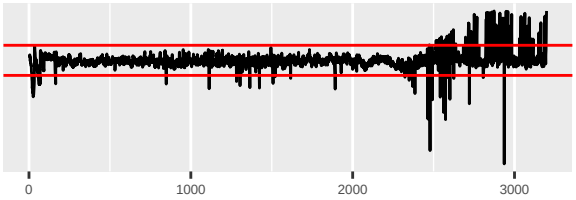
Cleaned



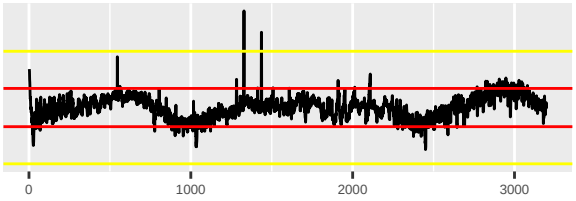
Ex # 11



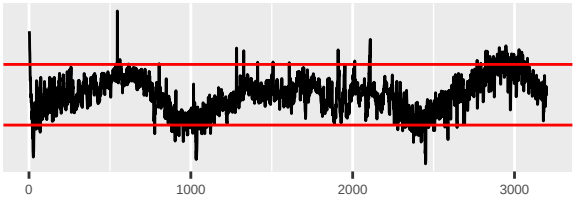
Cleaned



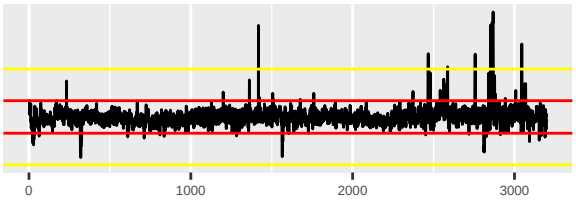
Ex # 25



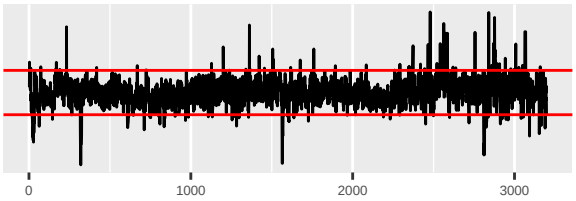
Cleaned



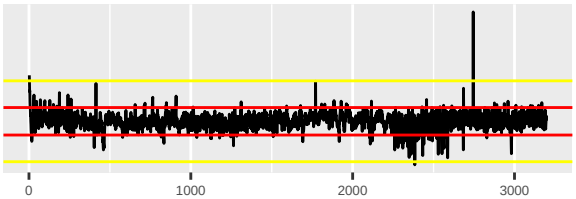
Ex # 7



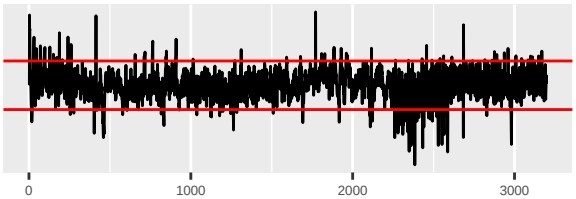
Cleaned



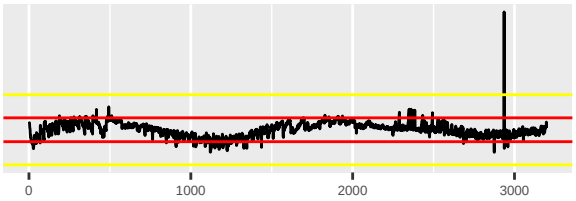
Ex # 32



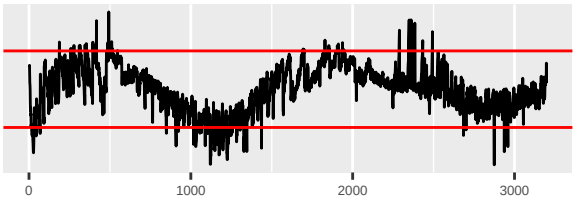
Cleaned



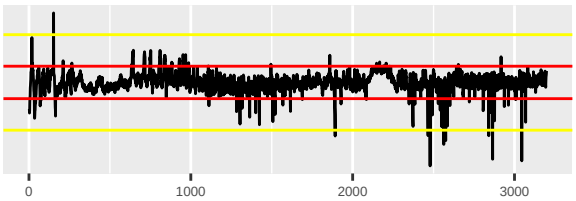
Ex # 34



Cleaned



Ex # 9



Cleaned

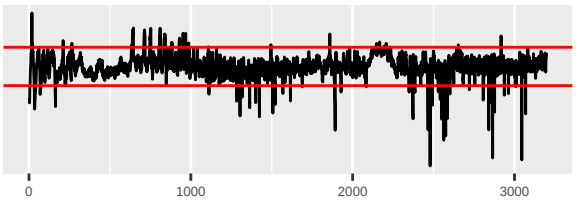


Figure 1: The Rake (my own 'device' :)

Custom Predictors

Looking at graphs presented in previous section I noticed that flux chart of stars that have exoplanets little bit resemble shape of "rake with teeth", where star flux decreases periodically (the teeths). So I will try to calculate predictors which will try to catch this effect. I will select abservations which are below certain level, then try to group the ones which are close together (parameter WINDOW_H) (meaning one planet pass before the star). Then I capture 3 statistics:

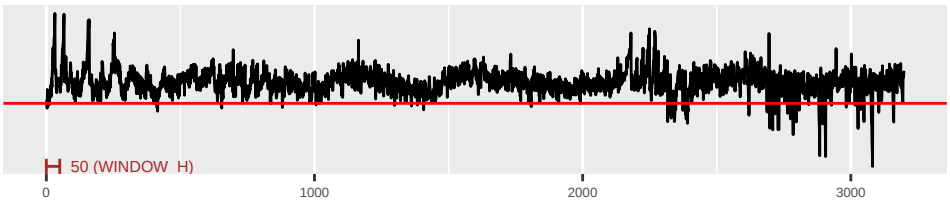
- number of such groups (aka teeths)
- standard deviation of distances between groups (the less, the more regular "teeths" are)
- number of rounded different distances between groups (the less, the more regular "teeths" are)

Then add them as my predictors. I'm doing this for configurable number of levels (parameter LEVELS_NO) from $\mu - 1.7\sigma$ to $\mu - 5\sigma$ in equal steps. For more details about used algorithm please look at provided source code.

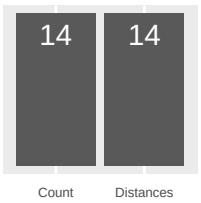
Below charts present sample of stars (the same stars as in previous section) with marked one selected level: level #1. Red line mark level, and on the right there are listed first and third predicate (as above). One may optically verify it against the chart. Comparing non-exoplanet and exoplanet stars one may notice that those predictors used to be equal each other for non-exoplanet while are more frequently inequal for stars with exoplanets. This is promising result which I hope made those predictors usable by ML models.

No exoplanet

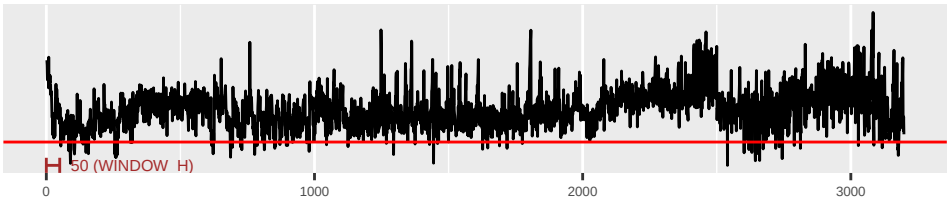
NEx # 1498



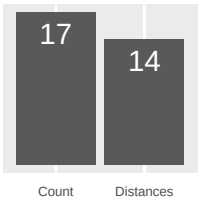
Teeths, sd: 80.73



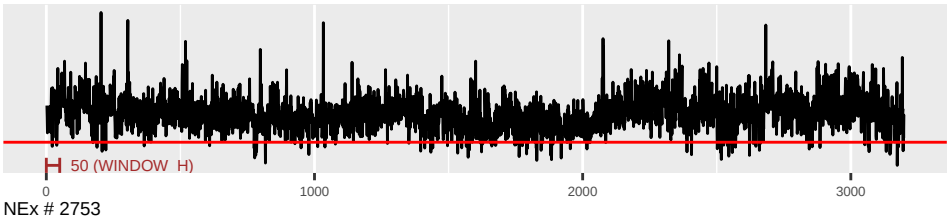
NEx # 2064



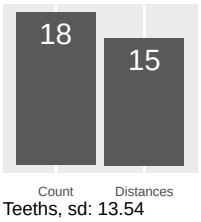
Teeths, sd: 4.36



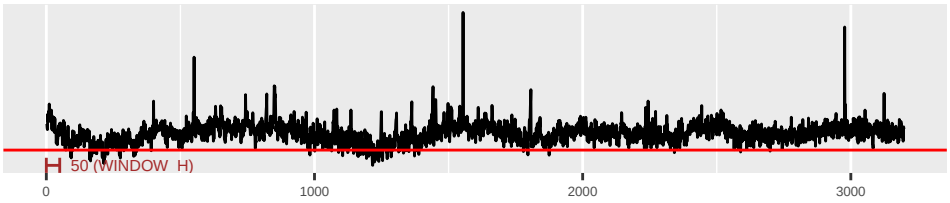
NEx # 3994



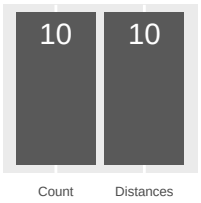
Teeths, sd: 1.76



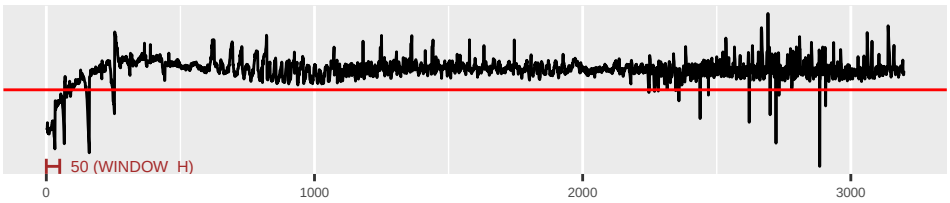
NEx # 2753



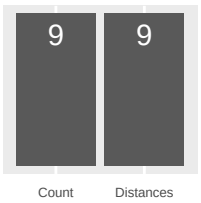
Teeths, sd: 13.54



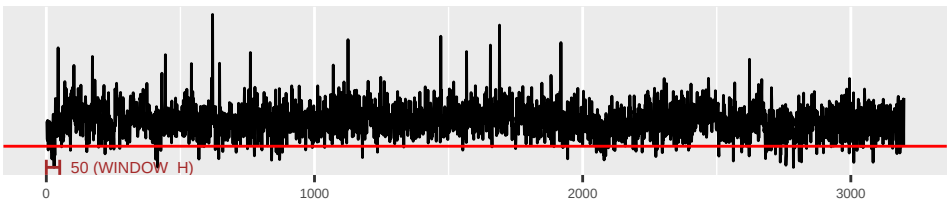
NEx # 882



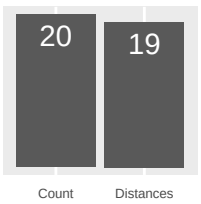
Teeths, sd: 90.01



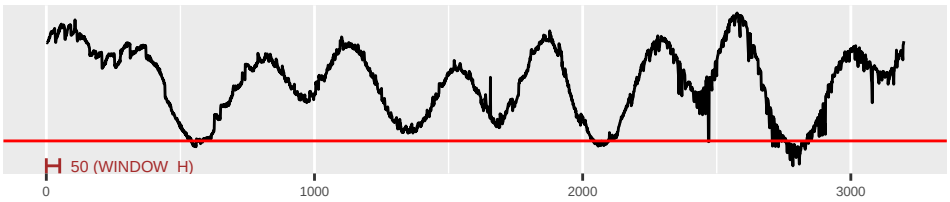
NEx # 3616



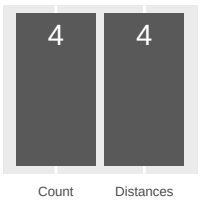
Teeths, sd: 1.84



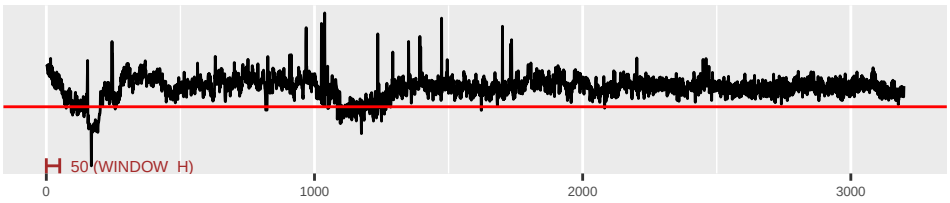
NEx # 3431



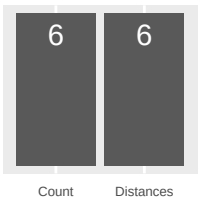
Teeths, sd: 73.31



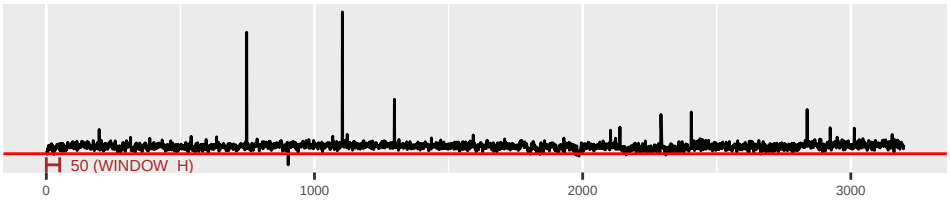
NEx # 2796



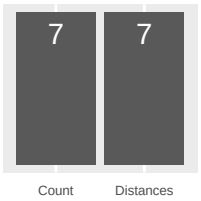
Teeths, sd: 22.97



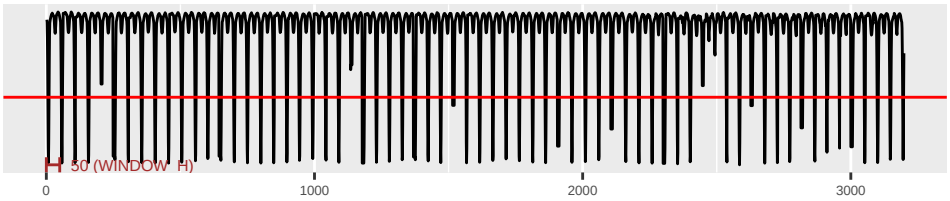
NEx # 2263



Teeths, sd: 10.62

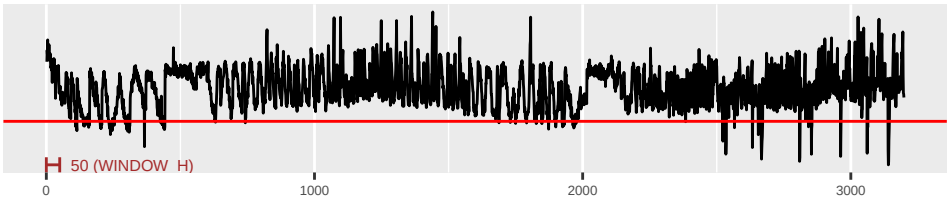


NEx # 2374

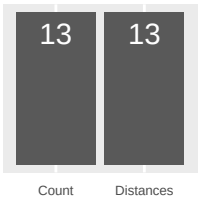


Teeths, sd: 3741.47

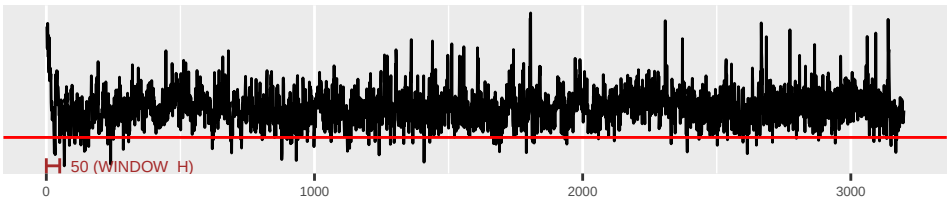
NEx # 4605



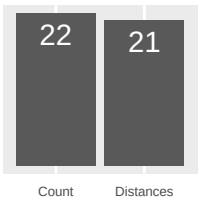
Teeths, sd: 12.44



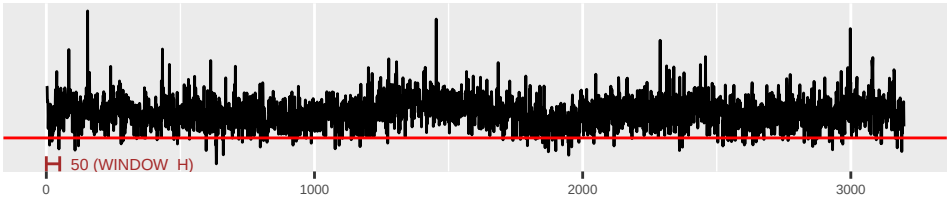
NEx # 77



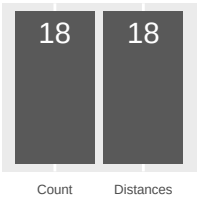
Teeths, sd: 2.48



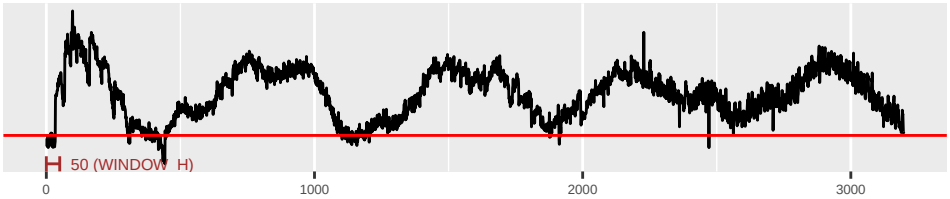
NEx # 5018



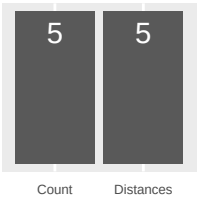
Teeths, sd: 1.96



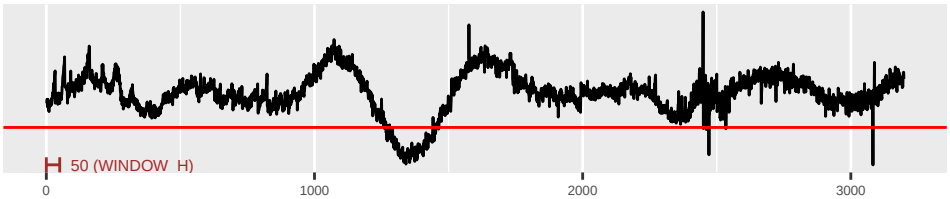
NEx # 2408



Teeths, sd: 46.99

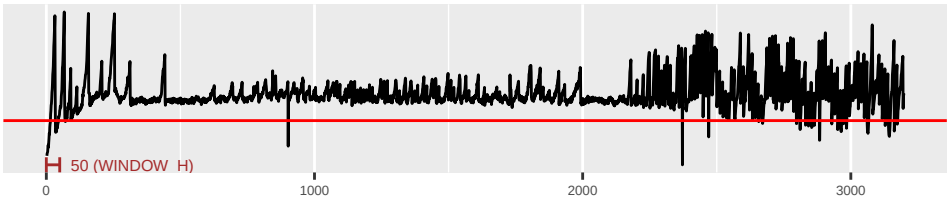


NEx # 1806

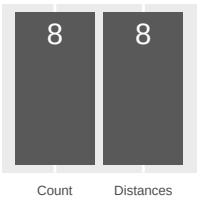


Teeths, sd: 71.98

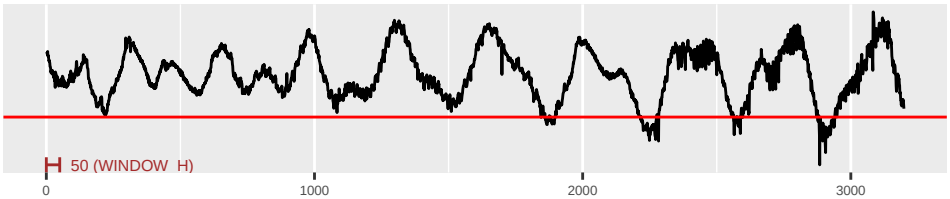
NEx # 2180



Teeths, sd: 64.76

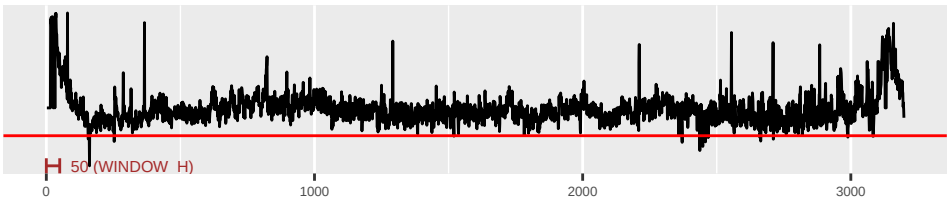


NEx # 3722

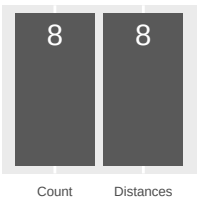


Teeths, sd: 156.49

NEx # 4863

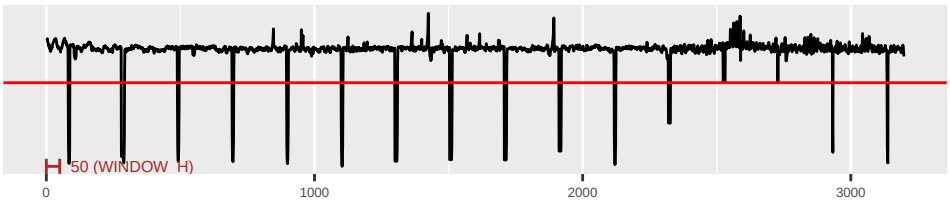


Teeths, sd: 24.06

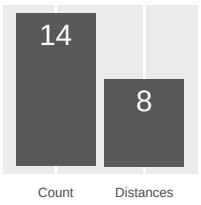


With Exoplanet

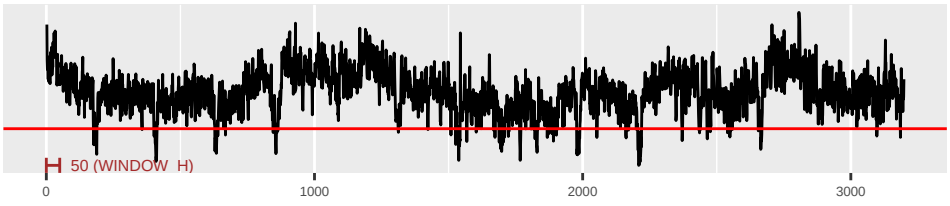
Ex # 8



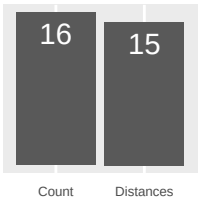
Teeths, sd: 314.63



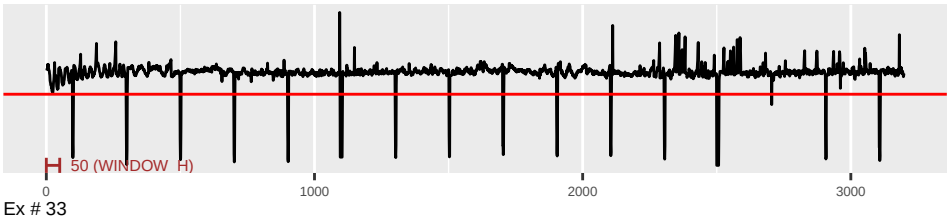
Ex # 20



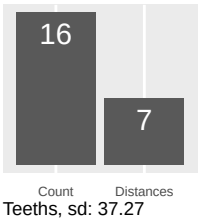
Teeths, sd: 95.03



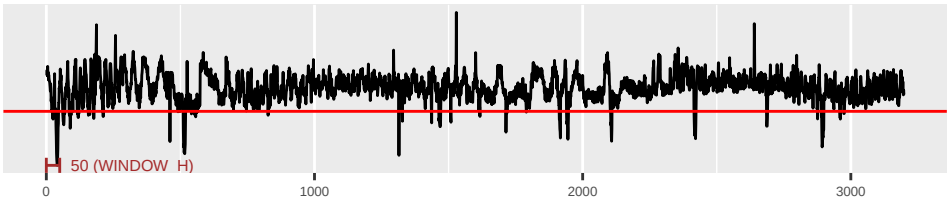
Ex # 36



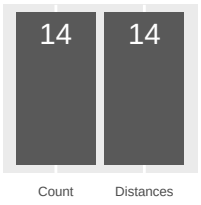
Teeths, sd: 176.52



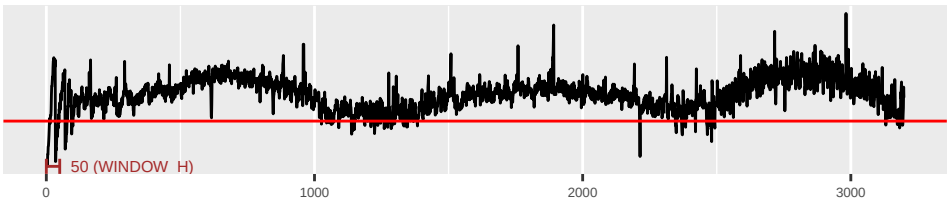
Ex # 33



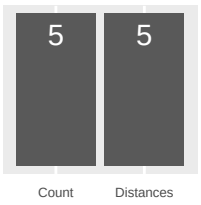
Teeths, sd: 37.27



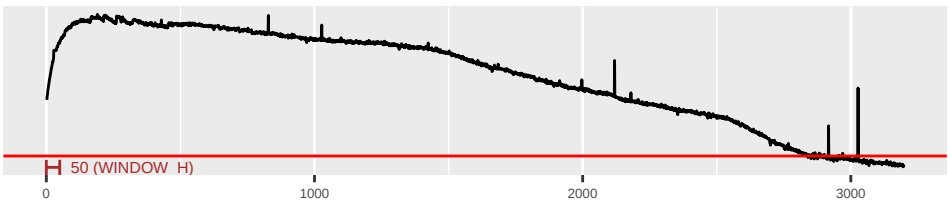
Ex # 17



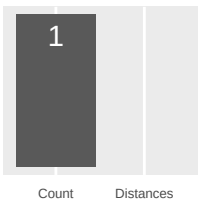
Teeths, sd: 28.94



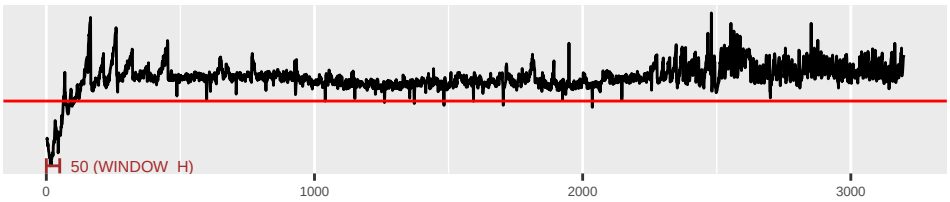
Ex # 15



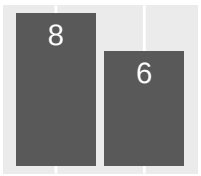
Teeths, sd: NA



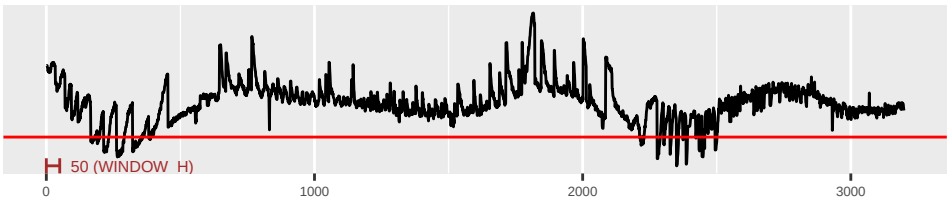
Ex # 21



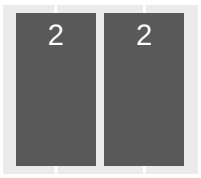
Teeths, sd: 657.29



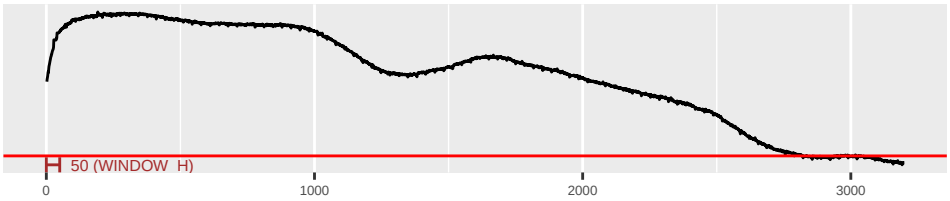
Ex # 3



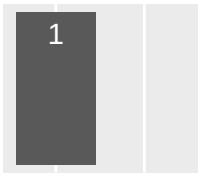
Teeths, sd: 87.88



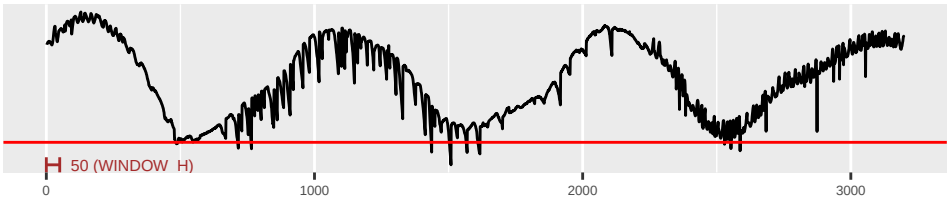
Ex # 31



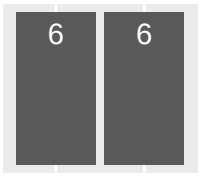
Teeths, sd: NA



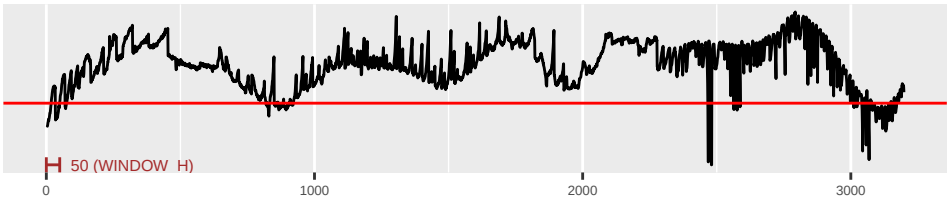
Ex # 22



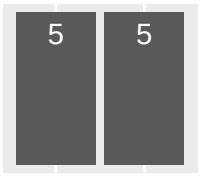
Teeths, sd: 467.90



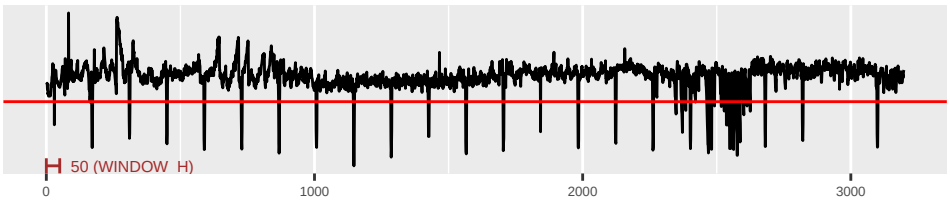
Ex # 5



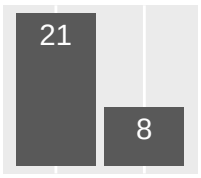
Teeths, sd: 451.18



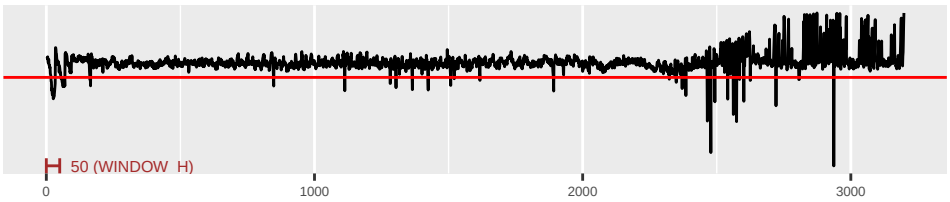
Ex # 2



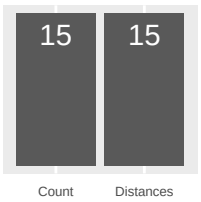
Teeths, sd: 36.77



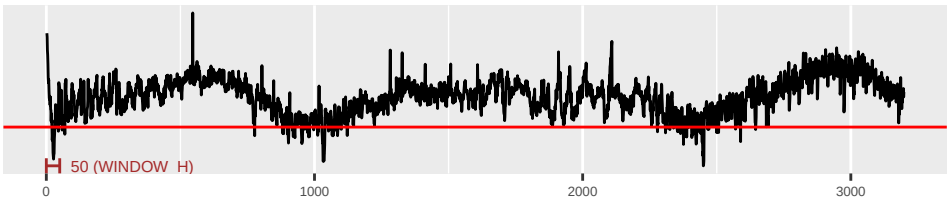
Ex # 11



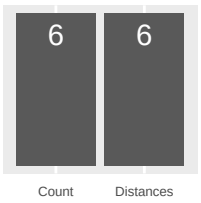
Teeths, sd: 477.83



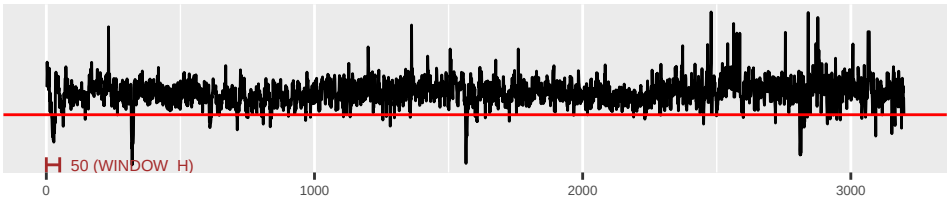
Ex # 25



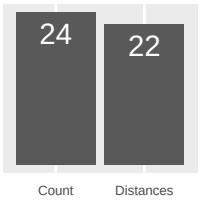
Teeths, sd: 50.32



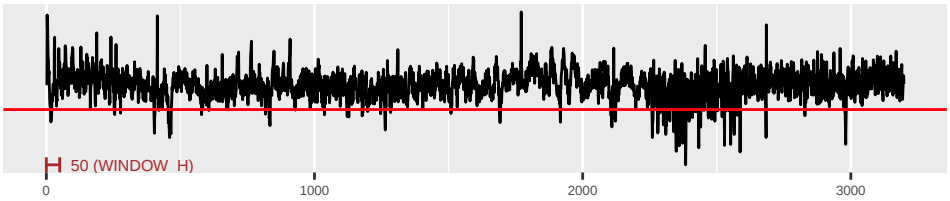
Ex # 7



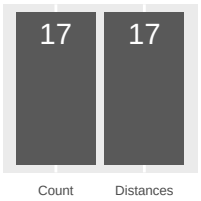
Teeths, sd: 24.36



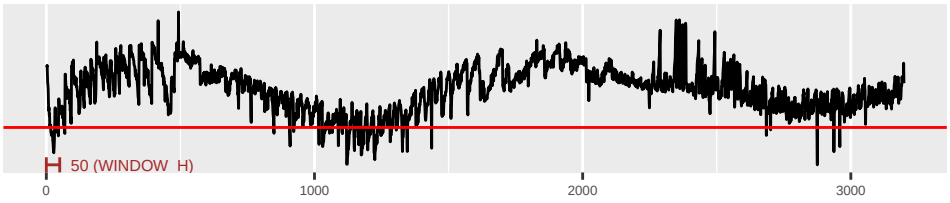
Ex # 32



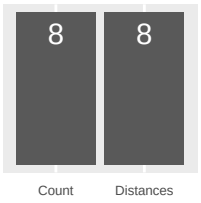
Teeths, sd: 33.37



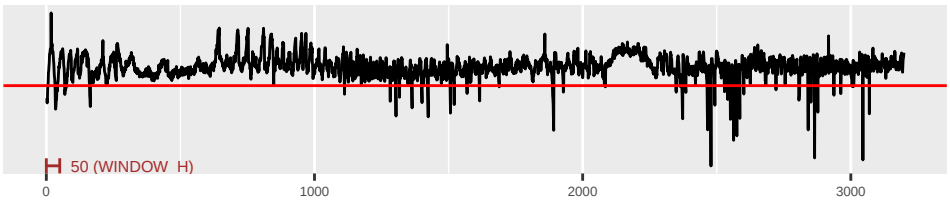
Ex # 34



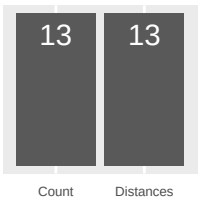
Teeths, sd: 24.07



Ex # 9



Teeths, sd: 90.90



Checking different ML algorithms

Now we will check different machine learning algorithms provided by H2O framework afainst our data set with prepared custom predictors. All models were trained using default parameters.

Generalized Linear Model

Obtained Result:

```
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                        max f1  0.813351 0.992665 205
## 2                        max f2  0.813351 0.997053 205
## 3                  max f0point5  0.895833 0.991168 203
## 4                  max accuracy  0.895833 0.985437 203
## 5                 max precision  0.988613 1.000000   0
## 6                    max recall  0.813351 1.000000 205
## 7               max specificity  0.988613 1.000000   0
## 8              max absolute_mcc  0.895833 0.401228 203
## 9   max min_per_class_accuracy  0.952032 0.666667 136
## 10 max mean_per_class_accuracy  0.929909 0.791461 186
```

Random Forest

Obtained Result:

```
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                        max f1  0.653333 0.992665 118
## 2                        max f2  0.653333 0.997053 118
## 3                  max f0point5  0.653333 0.988315 118
## 4                  max accuracy  0.653333 0.985437 118
## 5                 max precision  0.913395 0.994681 100
## 6                    max recall  0.653333 1.000000 118
## 7               max specificity  1.000000 0.666667   0
## 8              max absolute_mcc  0.913395 0.249379 100
## 9   max min_per_class_accuracy  0.968710 0.666667  57
## 10 max mean_per_class_accuracy  0.913395 0.793924 100
```

Gradient Boosting Machine

Obtained Result:

```
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                        max f1  0.814708 0.995098 184
## 2                        max f2  0.814708 0.998033 184
## 3                  max f0point5  0.814708 0.992180 184
## 4                  max accuracy  0.814708 0.990291 184
## 5                 max precision  0.997663 1.000000   0
## 6                    max recall  0.814708 1.000000 184
## 7               max specificity  0.997663 1.000000   0
## 8              max absolute_mcc  0.814708 0.574527 184
```

```
## 9   max min_per_class_accuracy  0.987919 0.497537  86
## 10 max mean_per_class_accuracy  0.814708 0.666667 184
```

Deep Learning

Obtained Result:

```
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                        max f1  0.973846 0.992665 205
## 2                        max f2  0.973846 0.997053 205
## 3                  max f0point5  0.973846 0.988315 205
## 4                  max accuracy  0.973846 0.985437 205
## 5                 max precision  1.000000 1.000000   0
## 6                    max recall  0.973846 1.000000 205
## 7              max specificity  1.000000 1.000000   0
## 8             max absolute_mcc  0.999872 0.136660  90
## 9   max min_per_class_accuracy  0.999892 0.418719  85
## 10 max mean_per_class_accuracy  0.999994 0.554187  21
```

Summary

Here is summary of results of checked models:

Model	Matthews CC	Mean Accuracy
GLM	0.401228	0.7914614
RF	0.2493786	0.7939245
GBM	0.574527	0.6666667
DL	0.1366598	0.5541872

The best perfoming algorithm seems to be **Gradient Boosting Machine** so I will use it in my final model.

Result

Final Gradient Boosting Machine Model

Parameters of used model:

```
## $model_id
## [1] "gbm_mod1"
##
## $training_frame
## [1] "exoTrainData_ext_df3_sid_8718_1"
##
## $seed
## [1] 6.811082e+18
##
## $distribution
## [1] "bernoulli"
##
```

```
## $x
##  [1] "SD"           "TEETHS1"      "TEETHS_SD1"   "TEETHS_UNIQ1"
##  [5] "TEETHS2"      "TEETHS_SD2"   "TEETHS_UNIQ2" "TEETHS3"
##  [9] "TEETHS_SD3"   "TEETHS_UNIQ3" "TEETHS4"      "TEETHS_SD4"
## [13] "TEETHS_UNIQ4"
##
## $y
## [1] "LABEL"
```

Obtained Result:

```
## H2OBinomialMetrics: gbm
##
## MSE:  0.005331968
## RMSE:  0.07302033
## LogLoss:  0.03300503
## Mean Per-Class Error:  0.3
## AUC:  0.7621239
## pr_auc:  0.9951086
## Gini:  0.5242478
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##        0   1    Error    Rate
## 0      2   3 0.600000    =3/5
## 1      0 565 0.000000  =0/565
## Totals 2 568 0.005263  =3/570
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                        max f1  0.781085 0.997352 363
## 2                        max f2  0.781085 0.998939 363
## 3                  max f0point5  0.781085 0.995770 363
## 4                  max accuracy  0.781085 0.994737 363
## 5                 max precision  0.999657 1.000000   0
## 6                    max recall  0.781085 1.000000 363
## 7                max specificity  0.999657 1.000000   0
## 8               max absolute_mcc  0.781085 0.630783 363
## 9   max min_per_class_accuracy  0.994933 0.600000 217
## 10 max mean_per_class_accuracy  0.970938 0.786726 347
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/
```

Conclusion

I managed to build model which obtained not bad MCC metrics against test data: 0.6307831. It was able to correctly recognize 2 out of 5 stars with exoplanets, but more importantly did not raise any false alarms when checking at least 600 stars without exoplanets (see confusion matrix above).

Ideas for further work

The time for course final capstone excercise is limited, and also the time that I can sacrifice on it is very limited. Because of those constraints I did not managed to realize some ideas that I initally intended. Most important are:

- I have set the model parameters (listed in "Data Exploration" section) using intuition, common sense and little experiments. Some of them should be really estimated by true trials using cross validation on parameters grid.
- Deeper investigate machine learning algorithms and possible to use parameters. Perform more experiments in this area.
- Try to use ensembling algorithm using best performing ML algorithms.
- I have a feeling that deep learning algorithm may produce good results when applied to raw data (instead of my calculated predictors). This should be verified. Flux chart might be treated as image (of rake) to recognize.
- There is a mathematical discipline that deals with functions distributing into harmonic compoments. I have a feeling that it could be somehow applied here. I mean that flux charts of stars with exoplanets are more cyclic and should be better approximated by harmonic components.
- Some Internet / literature reaserch sould be done, as our problem seems to be well known and kind of hot topic.

Final word

I would like to take opportunity and express my appreciation to all persons imvolved into preparation of this excelent "Data Scientists Proffesional" series of courses, espacially to *prof. Rafael A. Irizarry* - the main author. I am also very grateful that this course and all materials have been made publicly available .

I have learned a lot. Gained also some hands-on experiance with R programming language which I did not know before. I like very much RStudio programming environment, but honestly I'm not very big fan of R language, espacially because of its strange syntax, unclear type system, sometimes not clear help descriptions for packages, but mostly for slowness and one-thread approach. Looking forward I think I will be trying to use languages like Julia or Python for my next data science related learning efforts. I think I will continue learning of H2O framework. I may be also continuing work on this exoplanet project - check out my github repository for future updates.