Kepler

Adrien Foucart

18/09/2020

Introduction

This is a personnal work in the context of the HarvardX PH125.9x Data Science professional certificate.

The objectives are to present a study involving machine learning techniques and based on publicly available dataset.
The chosen dataset comes from observations made by the NASA Kepler space telescope. Based on the recording of light intensity, we intend to predict which star may host an exoplanet in its orbit.

Environment

The project is coded in R 4.0.2. We are going to use the following libraries:

Github

This project is available though the following **GitHub** link : https://github.com/adfouc/kepler.git

Dataset

The database is hosted on **Kaggle**: https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data

It was copied also in a shared **dropbox** folder: https://www.dropbox.com/sh/w1i36tti9g08n28/AAAKl7HwSB5MTrB7Unt3Vq4ba?dl=0

The R code automatically download and extract this database from the dropbox folder. This one is containing two CSV files, corresponding to the training dataset and to the testing dataset.

The training set contains 5087 rows. The validation set contains 569 rows.
Each row represents a star.

LABEL	1	2	3	4	5	6	7	8	9
2	93.85	83.81	20.10	-26.98	-39.56	-124.71	-135.18	-96.27	-79.89
2	-38.88	-33.83	-58.54	-40.09	-79.31	-72.81	-86.55	-85.33	-83.97
2	532.64	535.92	513.73	496.92	456.45	466.00	464.50	486.39	436.56
2	326.52	347.39	302.35	298.13	317.74	312.70	322.33	311.31	312.42
2	-1107.21	-1112.59	-1118.95	-1095.10	-1057.55	-1034.48	-998.34	-1022.71	-989.57
2	211.10	163.57	179.16	187.82	188.46	168.13	203.46	178.65	166.49

Column 1 (LABEL) tells if the star has extraplanets or not . Value “2” is an exoplanet star and value “1” is a non-exoplanet-star.

Columns 2 to 3198 represent the light intensity recorded for each star, at a different point in time.
We have no further details at this stage. We do not know in which unit this physical property is expressed exactly, but as we'll see later these values can be negative or positive and with large variations. We don't know either how much time there is between two consecutive points. We just know that the points are regularly spaced in time. The total time range is 80 days, therefore the delay between two observations is approximately 36 minutes.

Objectives and key steps

The objective is to build a classification algorithm. This algorithm would help us to recognize which stars have exoplanets, or rather which stars are good candidates for exoplanets and should be selected for further examinations.
We are going to explore the data, try to see what could be the differences between the two categories, how we might transform the data to make these differences more salient.
We'll try several classical machine learning algorithms, rather than focusing on a single algorithm and trying to optimize it.
As usual, the approach implies training the algorithm on a dedicated training dataset, then testing it on a separate validation dataset.
One particular aspect of our problem is that the dataset is largely unbalanced, with less than 1% of stars in the positive category. For this, we'll look at the confusion matrices to assess our algorithms, and not only on the level of accuracy.

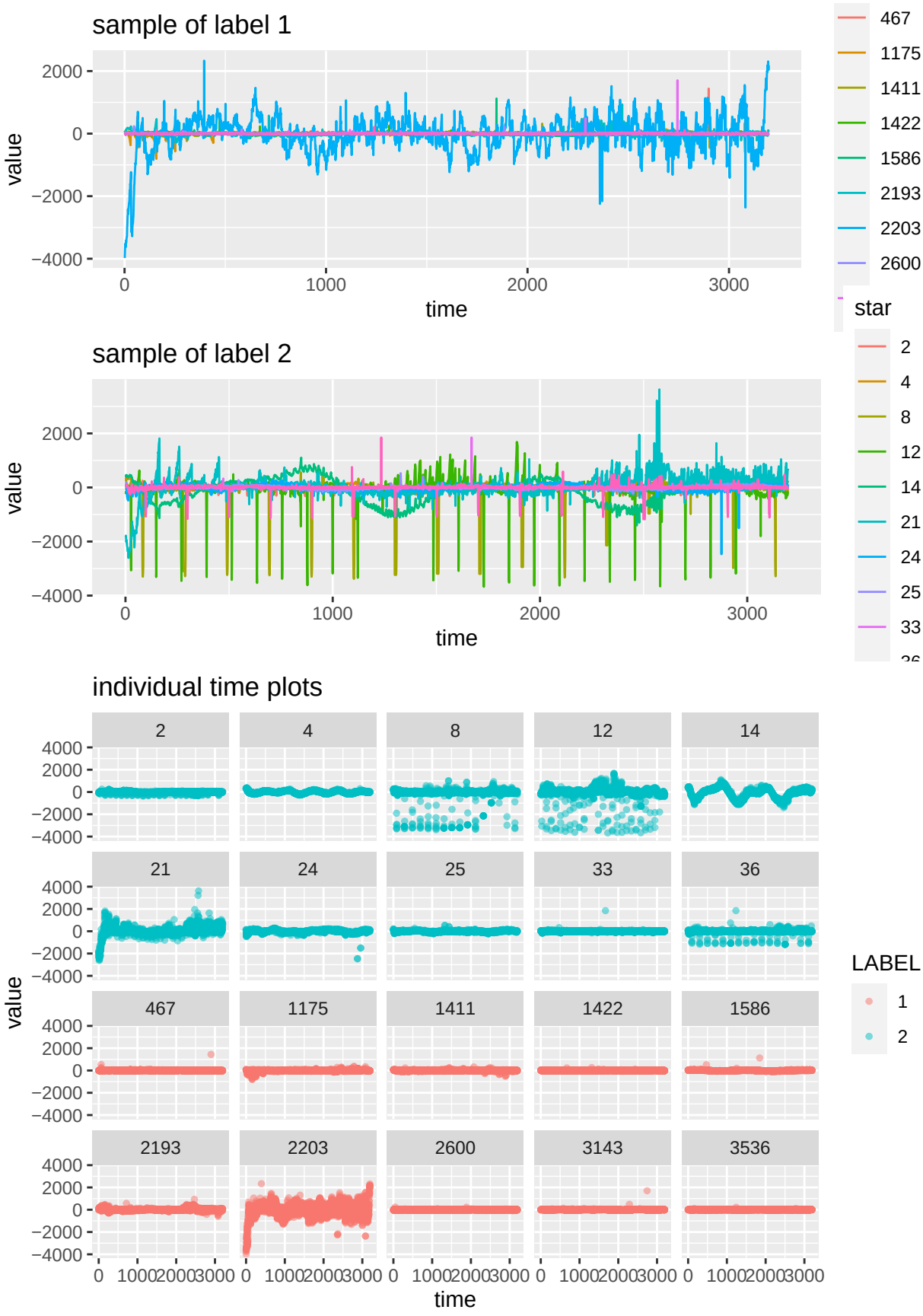
Analysis

Data exploration

As stated before, the ratio of label 2 / label 1 in the training set is very poor (0.7%):

set	label1	label2	ratio
training data	5050	37	0.0073267
validation data	564	5	0.0088652

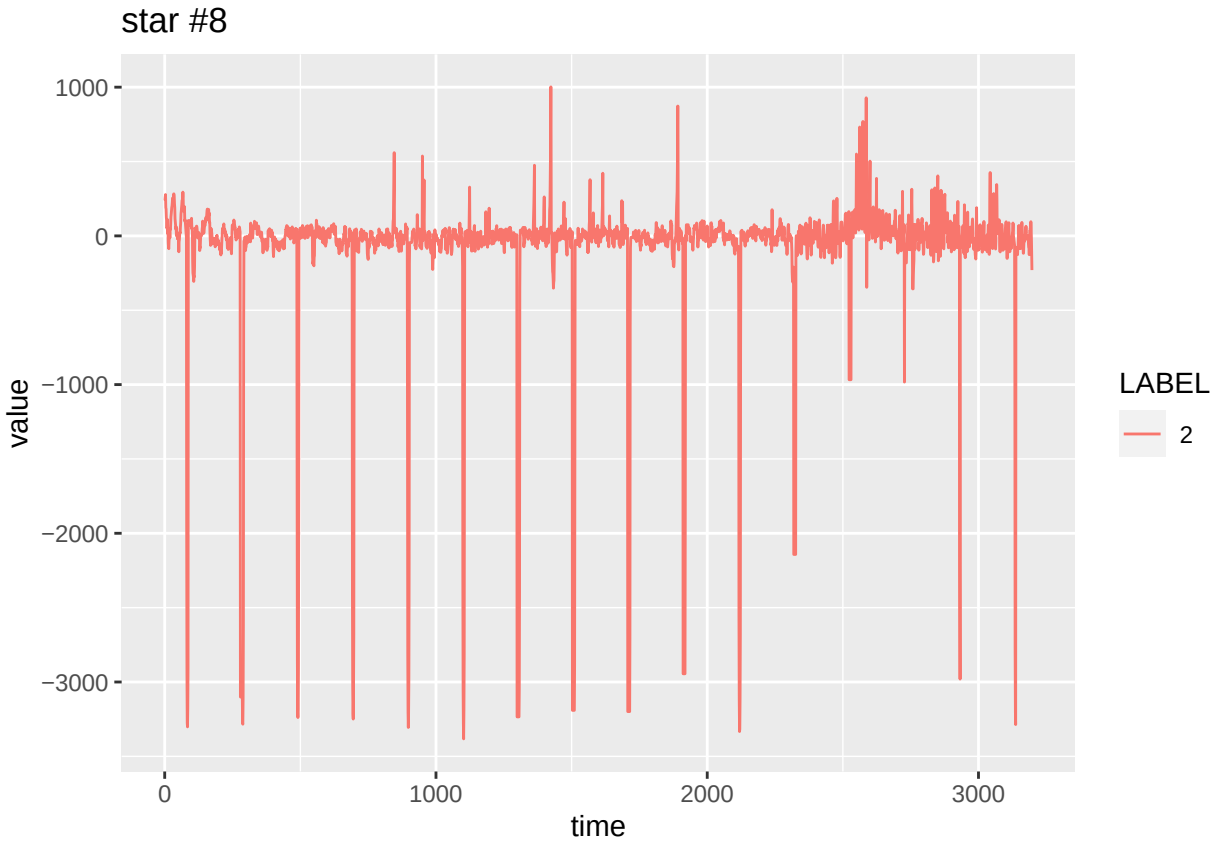
Let's plot the time evolution of a few stars in each category.


sample of label 1
value
time
467
1175
1411
1422
1586
2193
2203
2600
sample of label 2
star
2
4
8
12
14
21
24
25
33
individual time plots
LABEL
1
2

Data processing

Outliers and "dips"

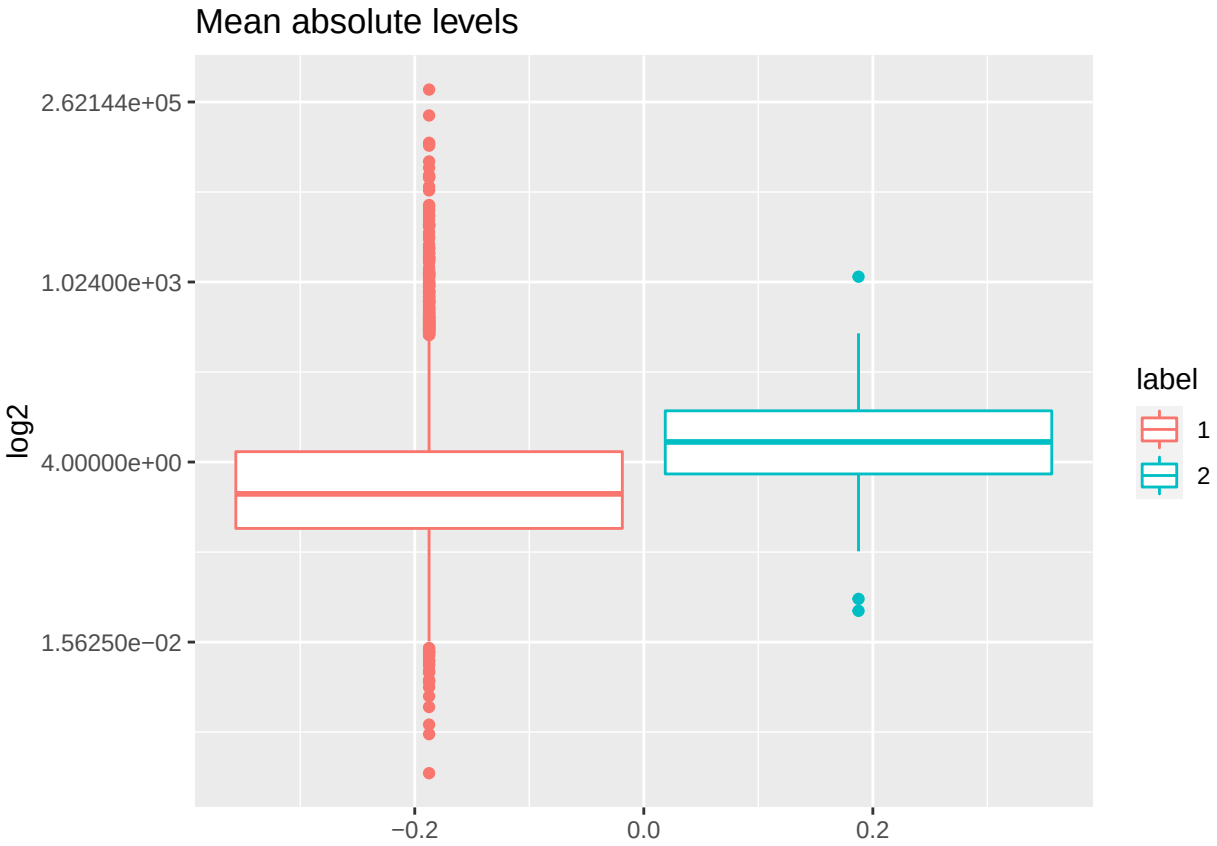
The data seems to contain outliers, i.e. very high or very low values, which could be due to errors in the dataset. Meanwhile, we should not remove all of these outliers and here's the reason.
Looking closely at the graphs, we can figure a pattern of luminosity dropping, or dimming, for a short period at regular intervals. See these "dips" for example on the trajectory of star number 8.



This makes sense if we think of a planet shading the star at every revolution. Detecting explonates based on this is known as the **photometric** technique. Unfortunately, this pattern is not visible for every exoplanet star.
Soon, we'll talk about data smoothing. We'll keep in mind to preserve these large negative variations.

Means, deviations and normalisation

We can compare the means and standard deviations of both classes. There are slight differences, but this is not meaningful due to the small number of stars in label 2 class.



Quantile tables to compare the variability of both classes:

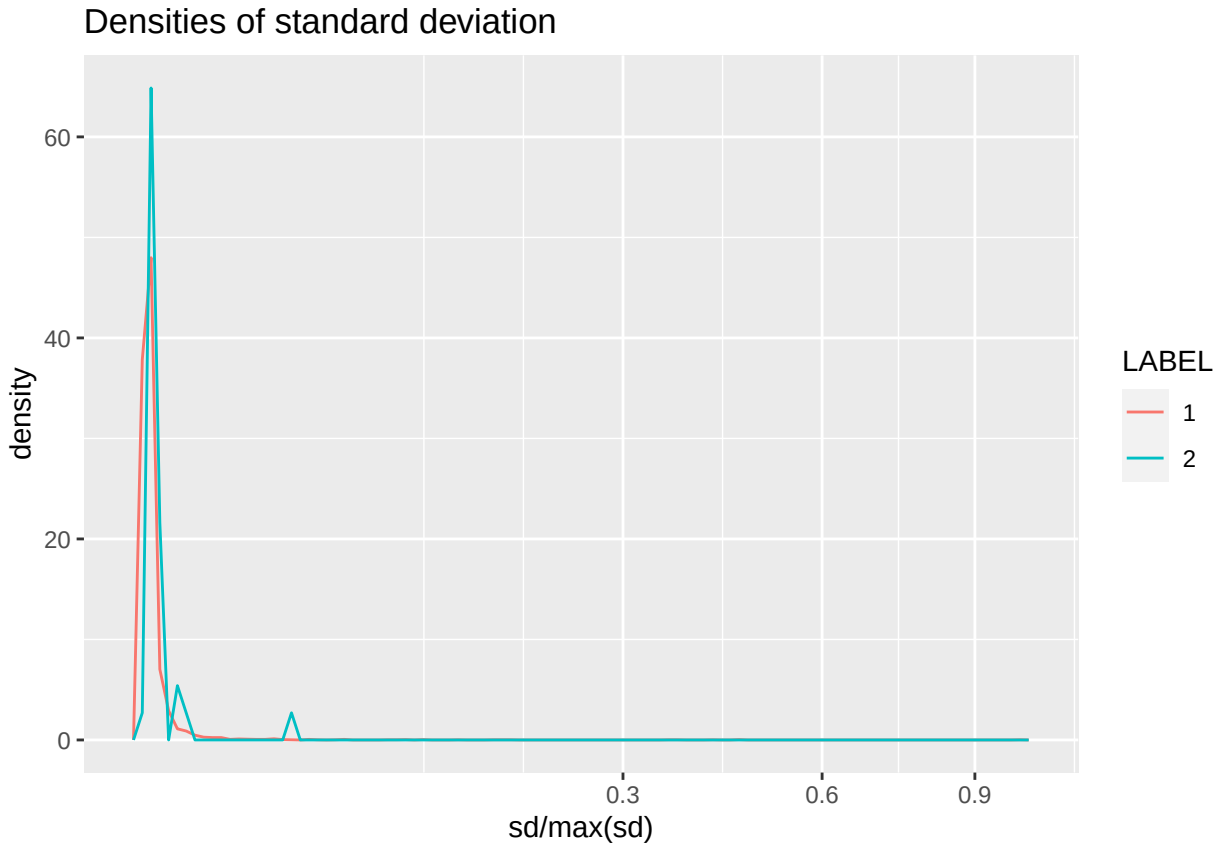
```
## [1] "Label 1 standard deviation quantiles"
```

	x
25%	16.30761
50%	39.74596
75%	110.45640
95%	1040.27042
99%	9673.01166

```
## [1] "Label 2 standard deviation quantiles"
```

	x
25%	48.72939
50%	155.68431
75%	350.41862
95%	1879.36731
99%	20097.48714

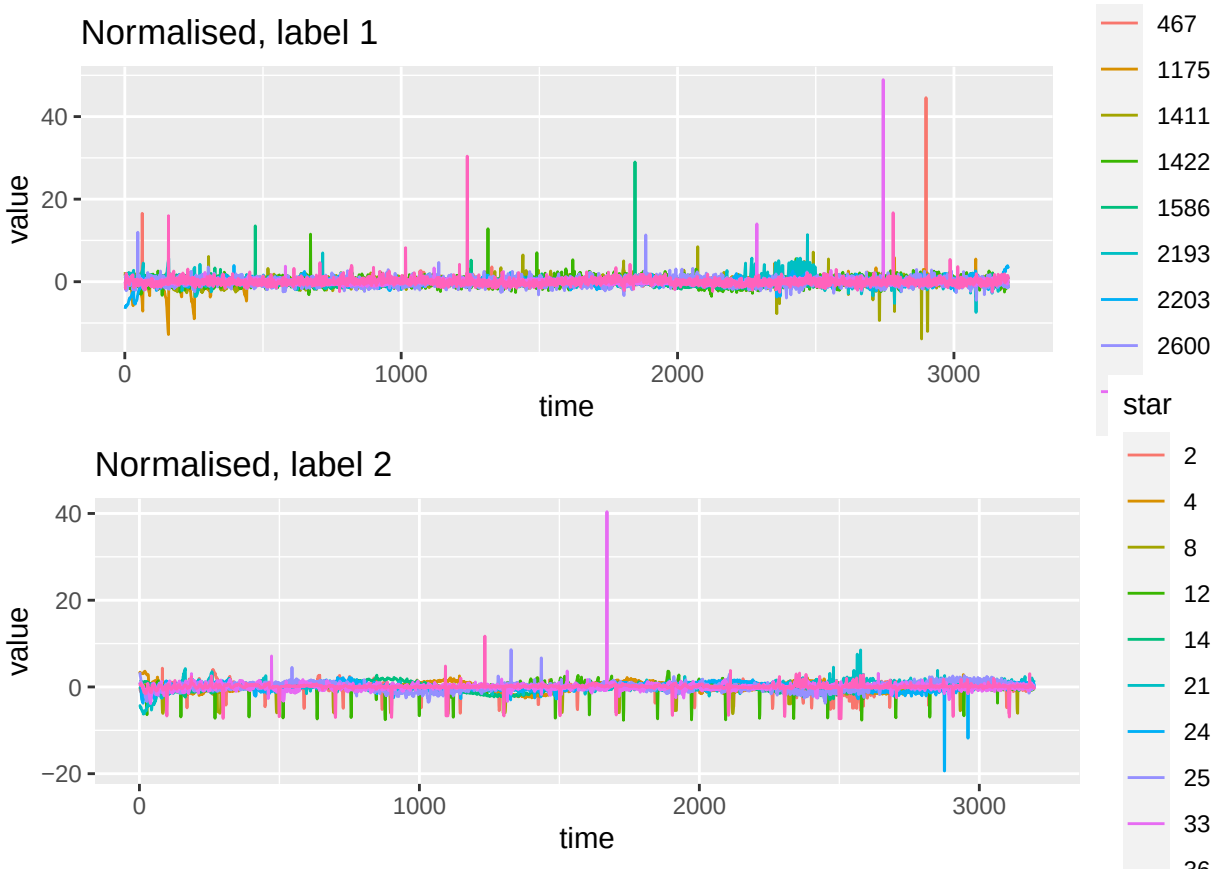
At last, this plot compares the two distributions of standard deviation (x scale is root-squared).



We cannot see obvious difference between the two classes. There are large variations in the standard deviations. We might apply a normalisation on the values of each star, i.e. center and reduce the values of each row:

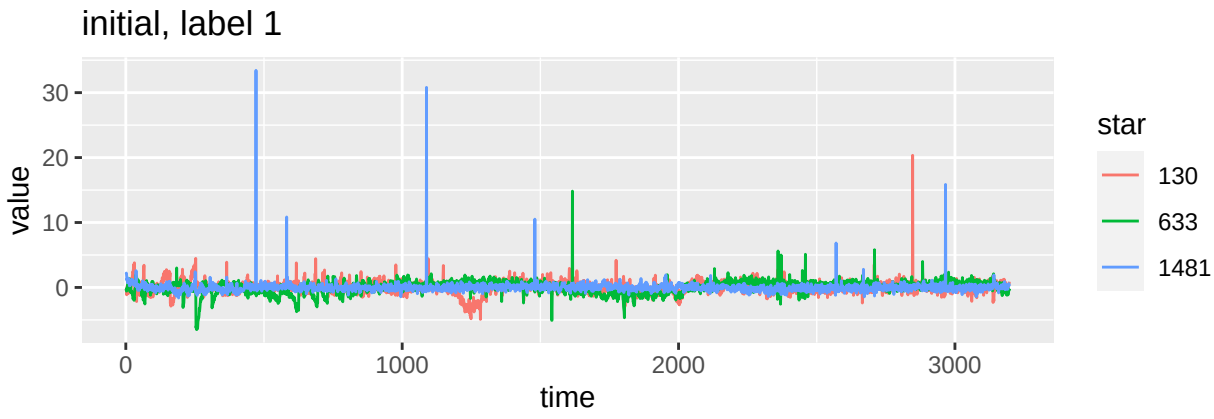
$$ReducedV_i(t) = \frac{V_i(t) - < V_i >}{SD(V_i)}$$

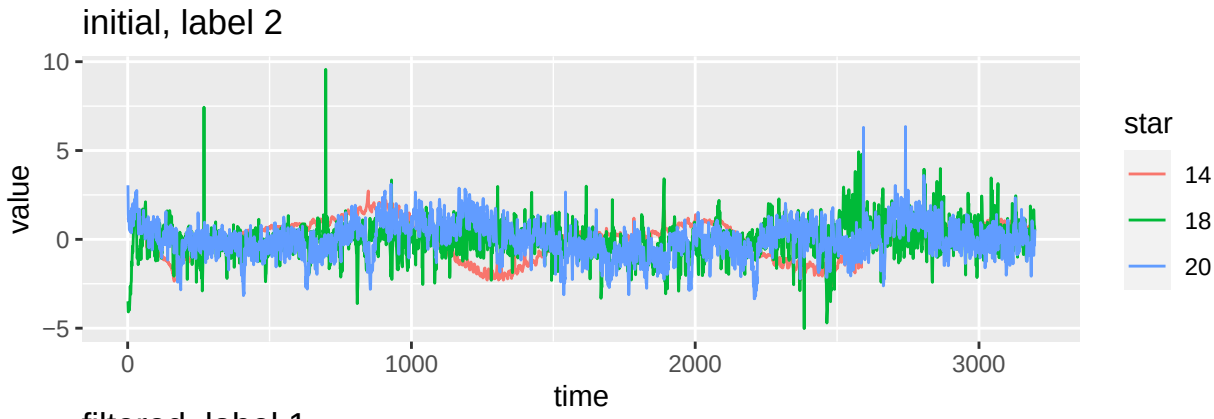
Here are a few plots after normalisation:

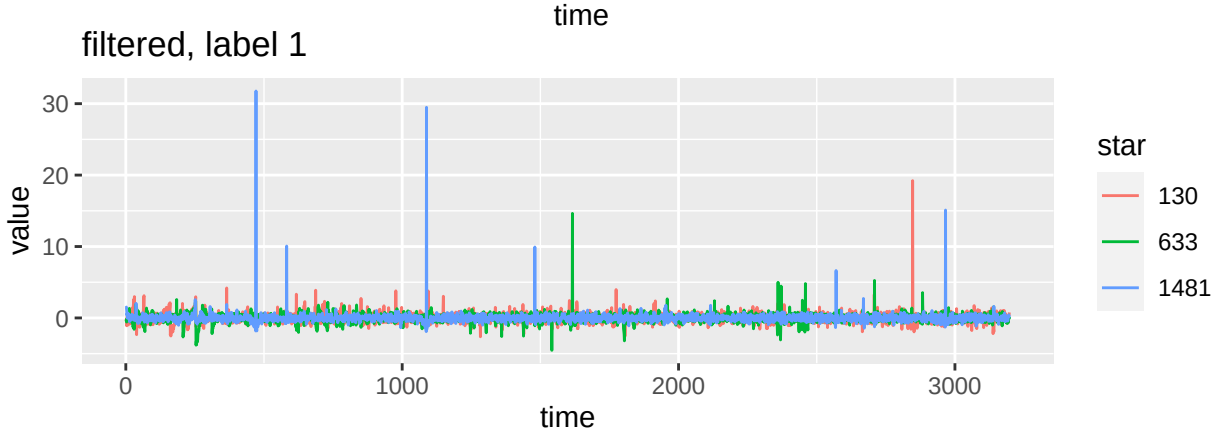


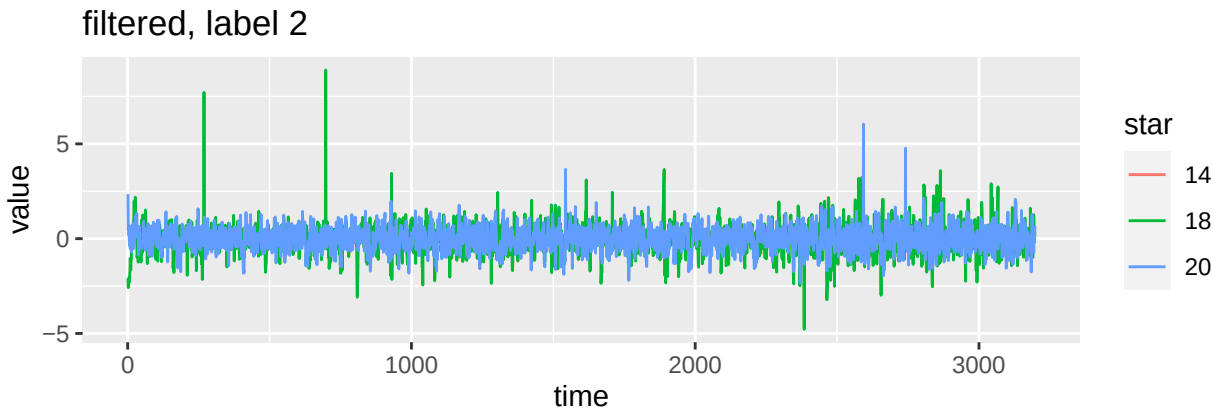
High pass filter

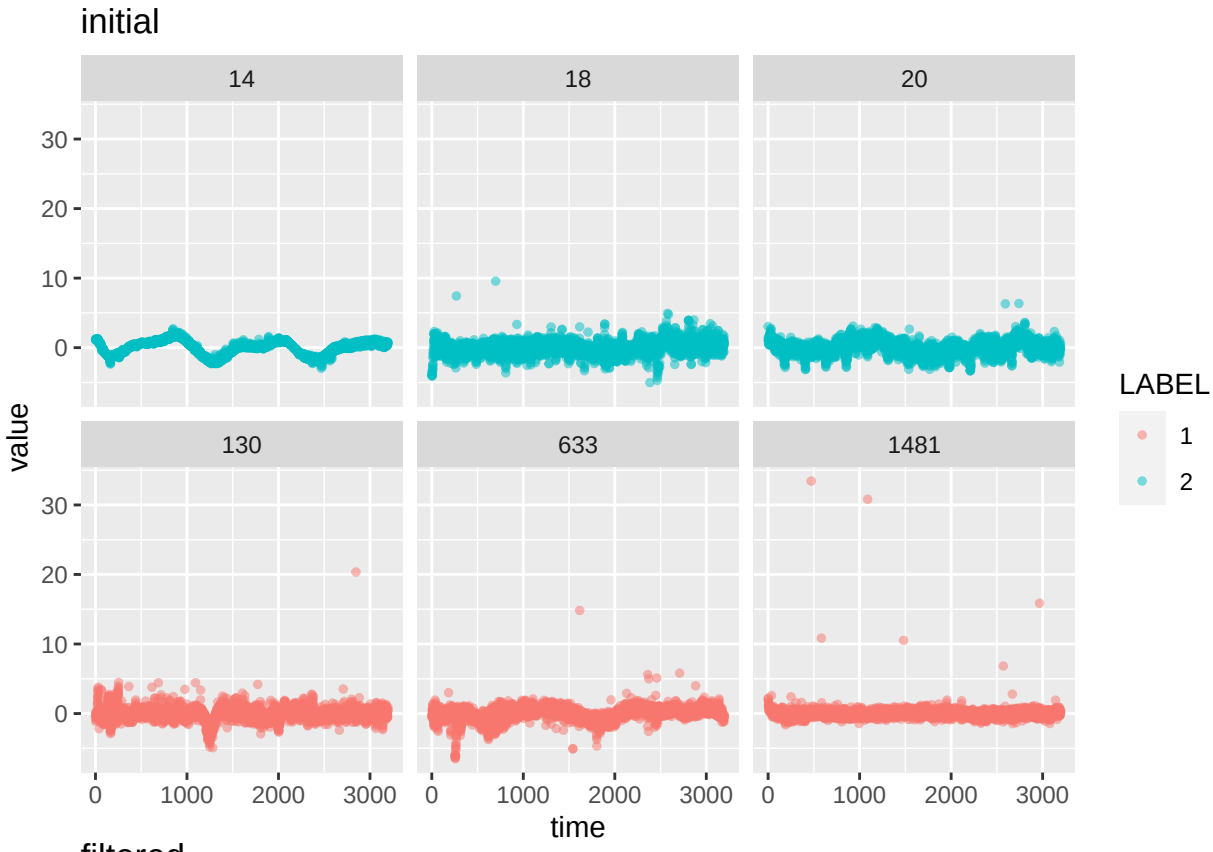
We can see various periodic components in both categories of stars. This is not in relation to the dips we want to isolate. Therefore we want to filter out these low frequency trends to clean the signal.
We use the **butter** function of the **signal** package. We apply this "high" filtering on the normalised data, with a cut frequency of 1/30th Nyqist Freq, which corresponds roughly to discarding events longer than a period of 80x24x2x30/tmax= 36 hours.

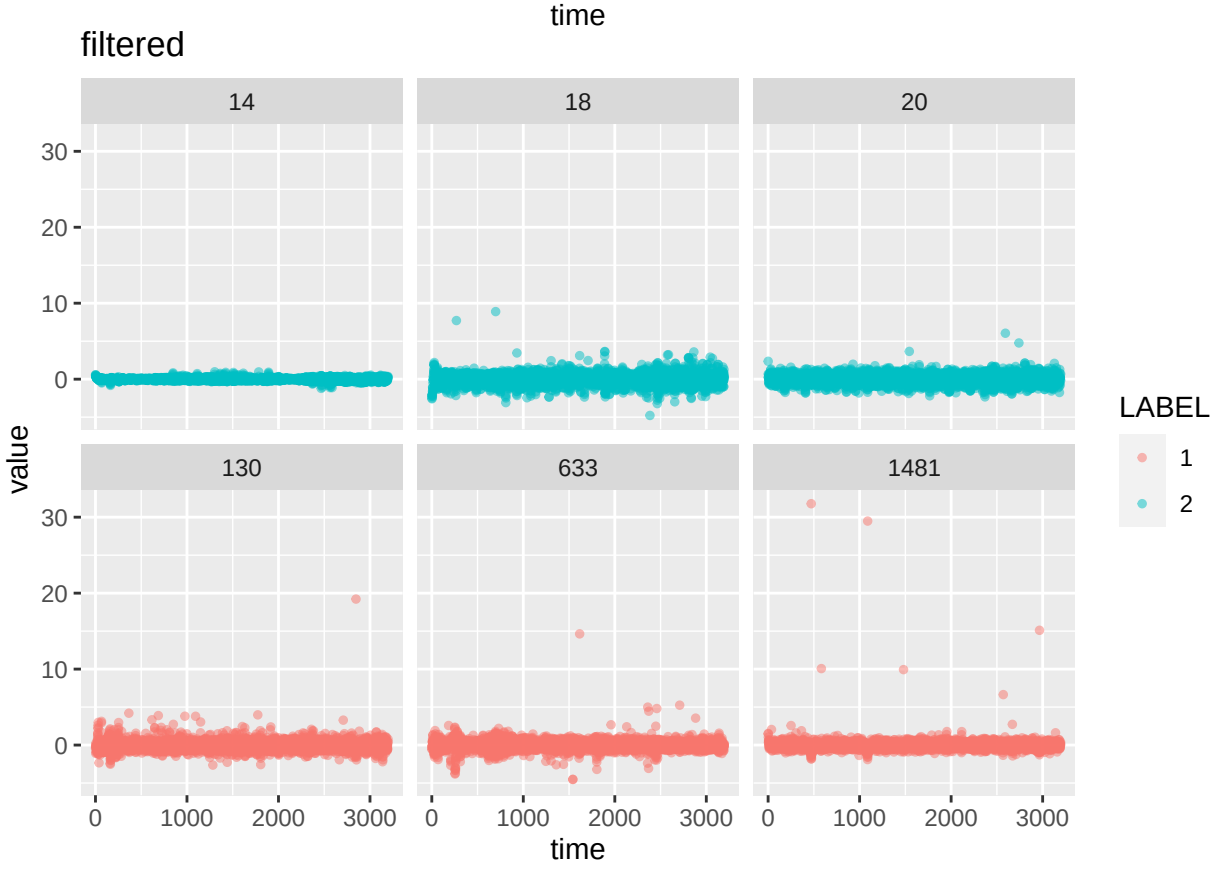

initial, label 1
value
time
star
130
633
1481


initial, label 2
value
time
star
14
18
20


filtered, label 1
value
time
star
130
633
1481


filtered, label 2
value
time
star
14
18
20

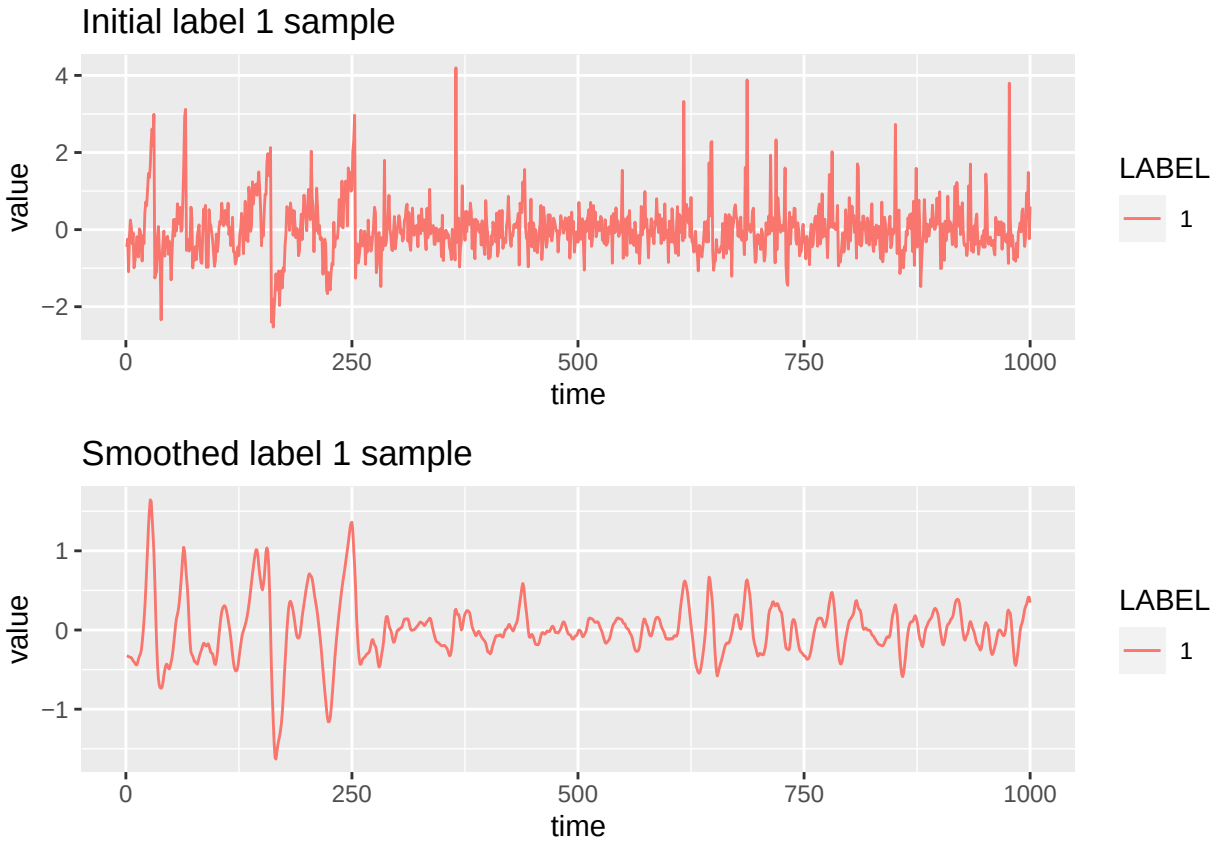

initial
14
18
20
130
633
1481
value
time
LABEL
1
2
0
10
20
30
0
1000
2000
3000

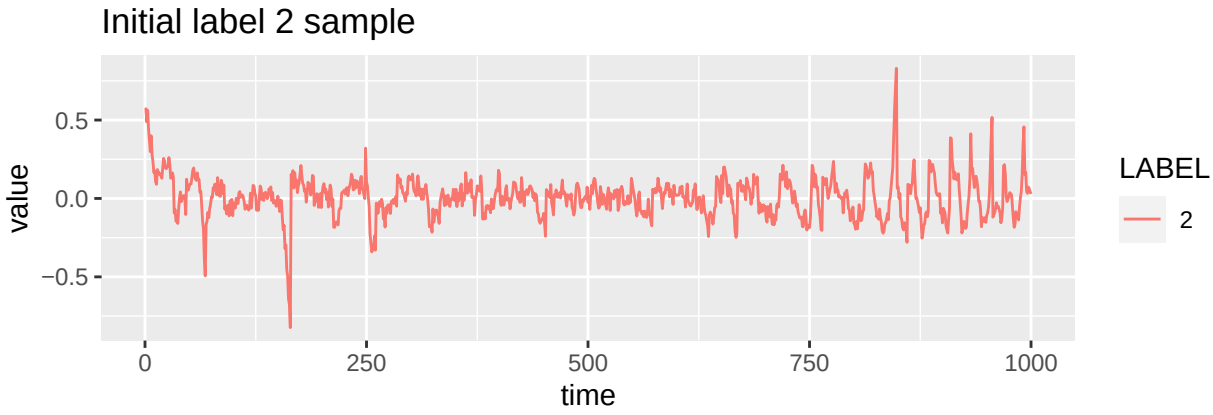

filtered
14
18
20
130
633
1481
value
time
LABEL
1
2
0
10
20
30
0
1000
2000
3000

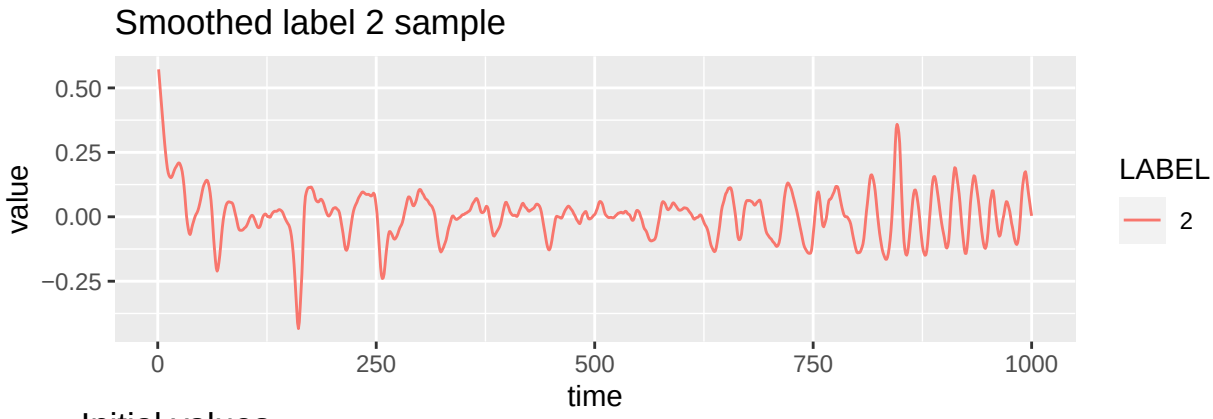
Smoothing

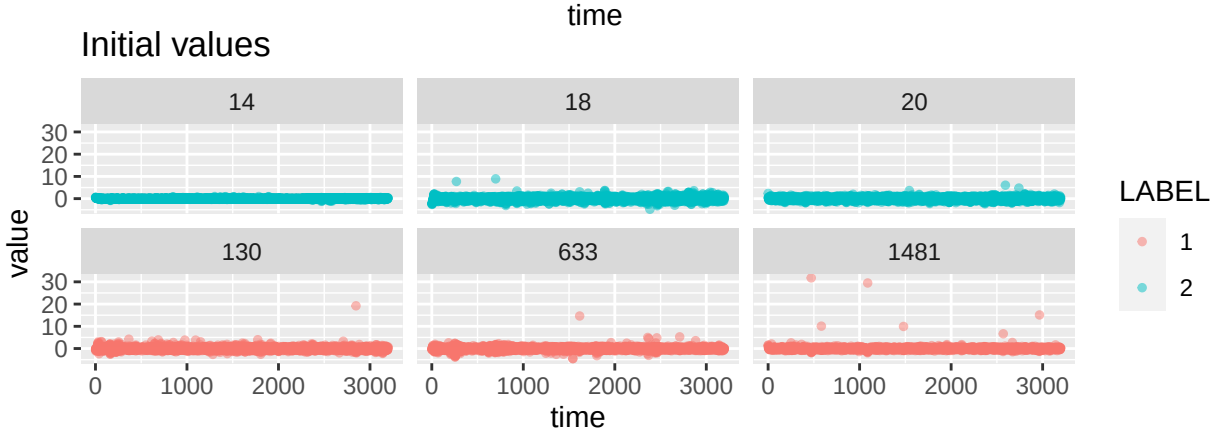
In order to improve the signals by removing high frequency noise, we might want to apply a smoothing technique to the data. We use here a **loess** method (Local weighted regression).

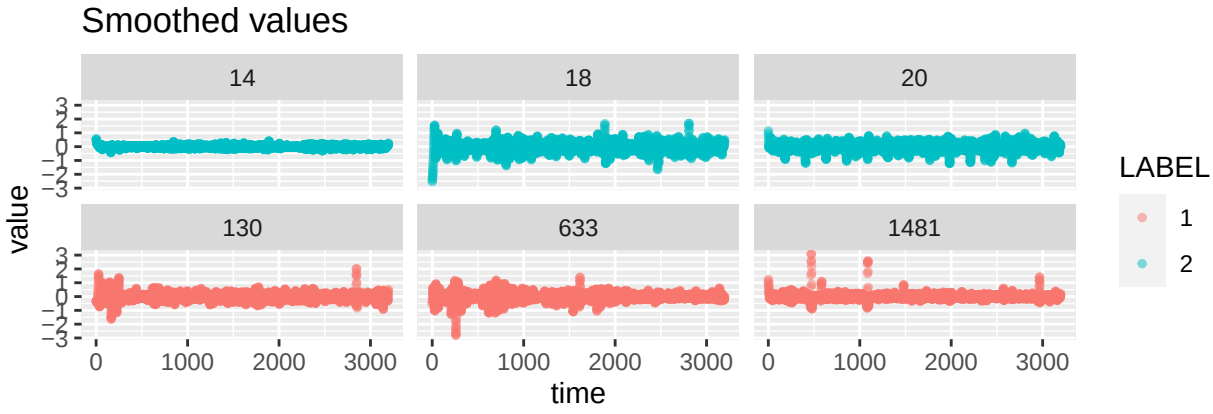
With a span of 0.005 it seems that we keep the (hopefully) usefull dips while some outliers are filtered out.




Initial label 2 sample
value
0.5
0.0
−0.5
0
250
500
750
1000
time
LABEL
2


Smoothed label 2 sample
value
0.50
0.25
0.00
−0.25
0
250
500
750
1000
time
LABEL
2


Initial values
14
18
20
130
633
1481
value
30
20
10
0
0
1000
2000
3000
time
LABEL
1
2


Smoothed values
14
18
20
130
633
1481
value
3
2
1
0
−1
−2
−3
0
1000
2000
3000
time
LABEL
1
2

Discrete fourier transform

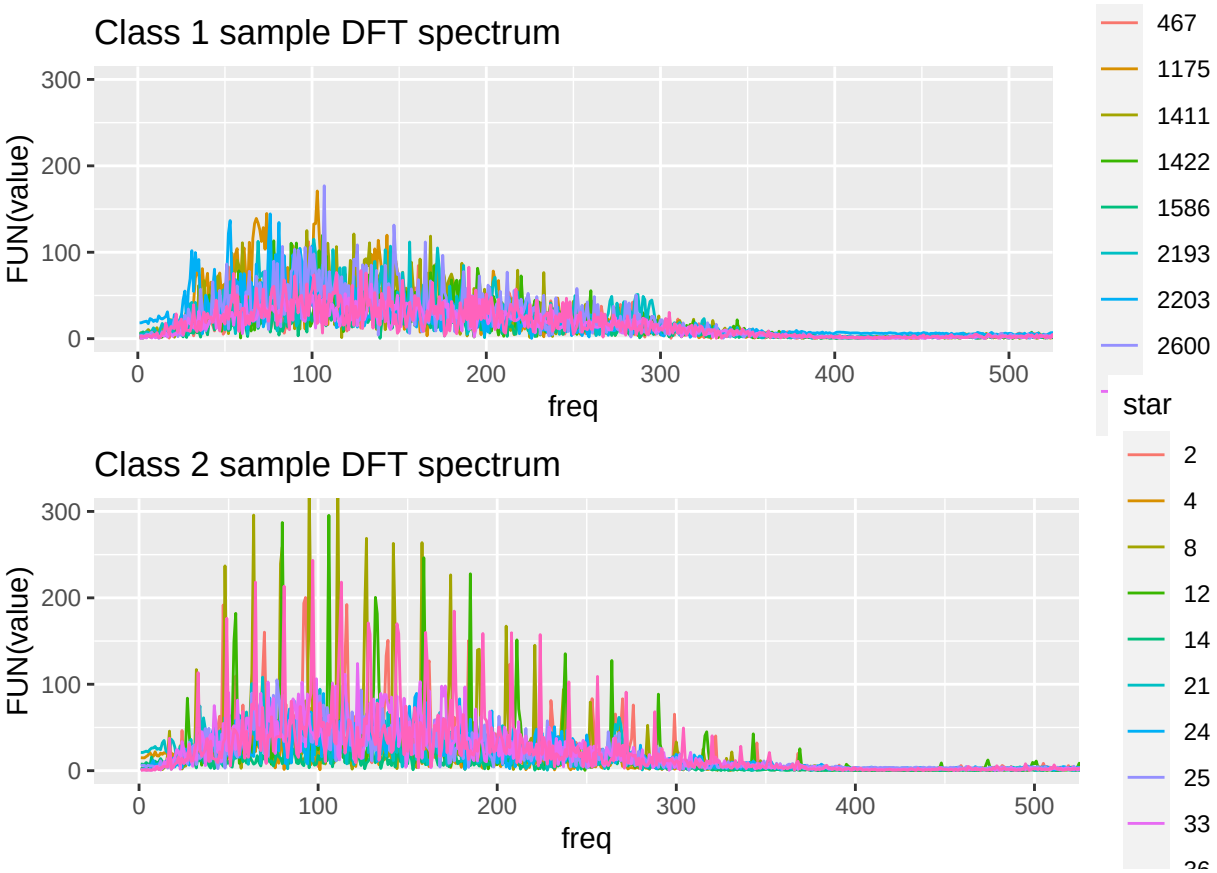
With the idea to catch the nature of the signal with time independant factors, we proceed to a **fast-fourrier-transform** (FFT) of the signal, which is a classic algorithm implementing the Discrete Fourrier Transform (DFT) mathematical tool:

$$X_k = \sum_j X_j exp(2i\pi \frac{jk}{N})$$

This step transforms data, for each star, from the time domain to the frequency domain. FFT computes complex numbers; since we are interested in the magnitude of each frequency we consider the modulus of X_k

LABEL	1	2	3	4	5	6	7	8	9
2	1.772884	1.704439	2.229502	0.8509102	9.079069	2.814572	3.726706	2.172946	2.366768
2	2.763513	2.659143	3.302138	3.5804661	3.792520	5.305434	6.514108	3.831980	5.560498
2	7.967222	7.869691	7.890368	8.8601225	8.043140	13.469317	11.192912	12.759626	8.946828
2	14.964658	14.858823	14.764838	16.3351753	17.771617	21.074811	18.003802	16.584209	21.893966
2	14.146561	14.460992	14.999239	17.3603705	19.904780	20.511534	16.789139	16.880266	16.861368
2	3.710904	3.929301	4.299473	1.5685198	4.766218	9.991135	8.875557	6.958330	5.681004

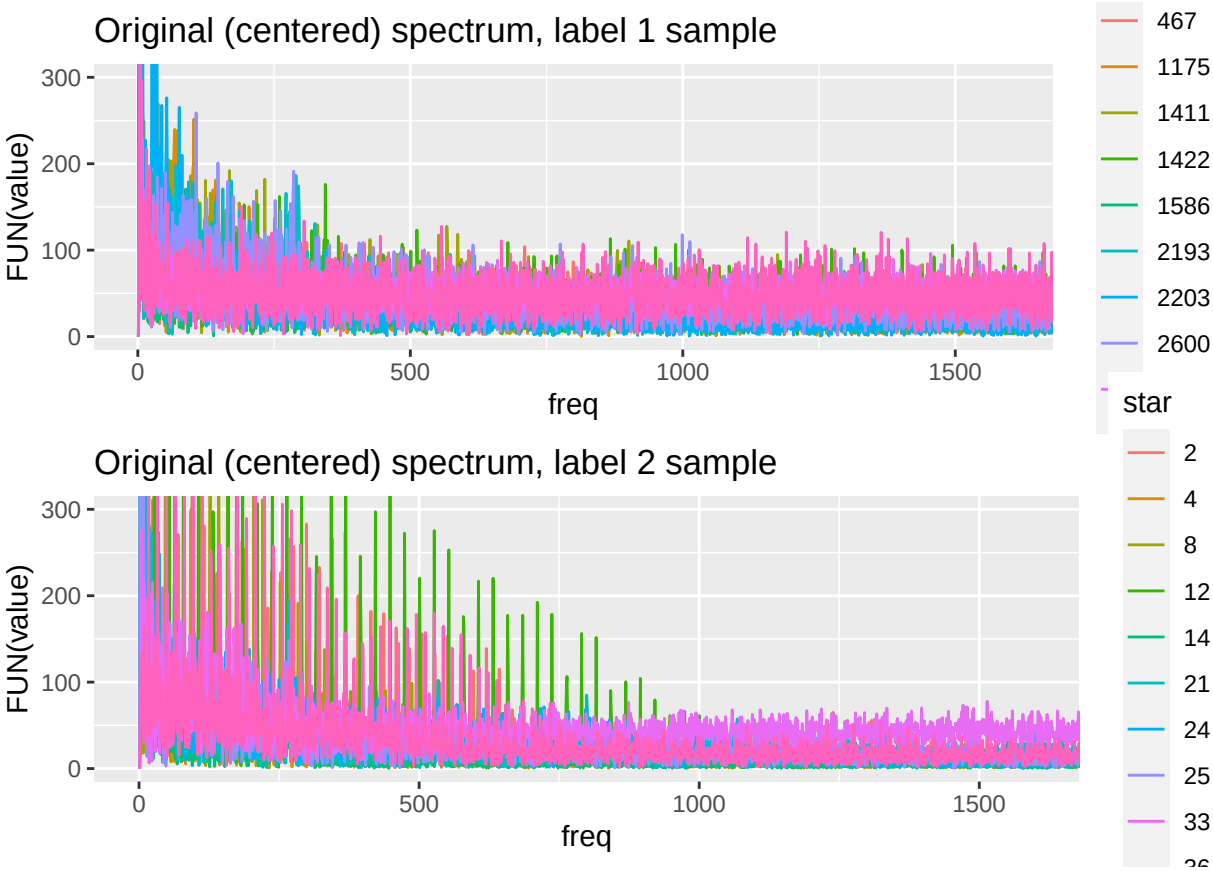
This gives a new table of 5087 rows and 3198 columns. Since the DFT is symetric with the Nyquist Frequency (N/2), we only consider the first half of the columns.
We might plot these spectrum for a sample of both classes:



On this last plot we can see the effect of filtering the low frequencies and also the smoothing effect on the

high frequencies.

We can compare it with the spectrum of the original (centered and reduced) signal :

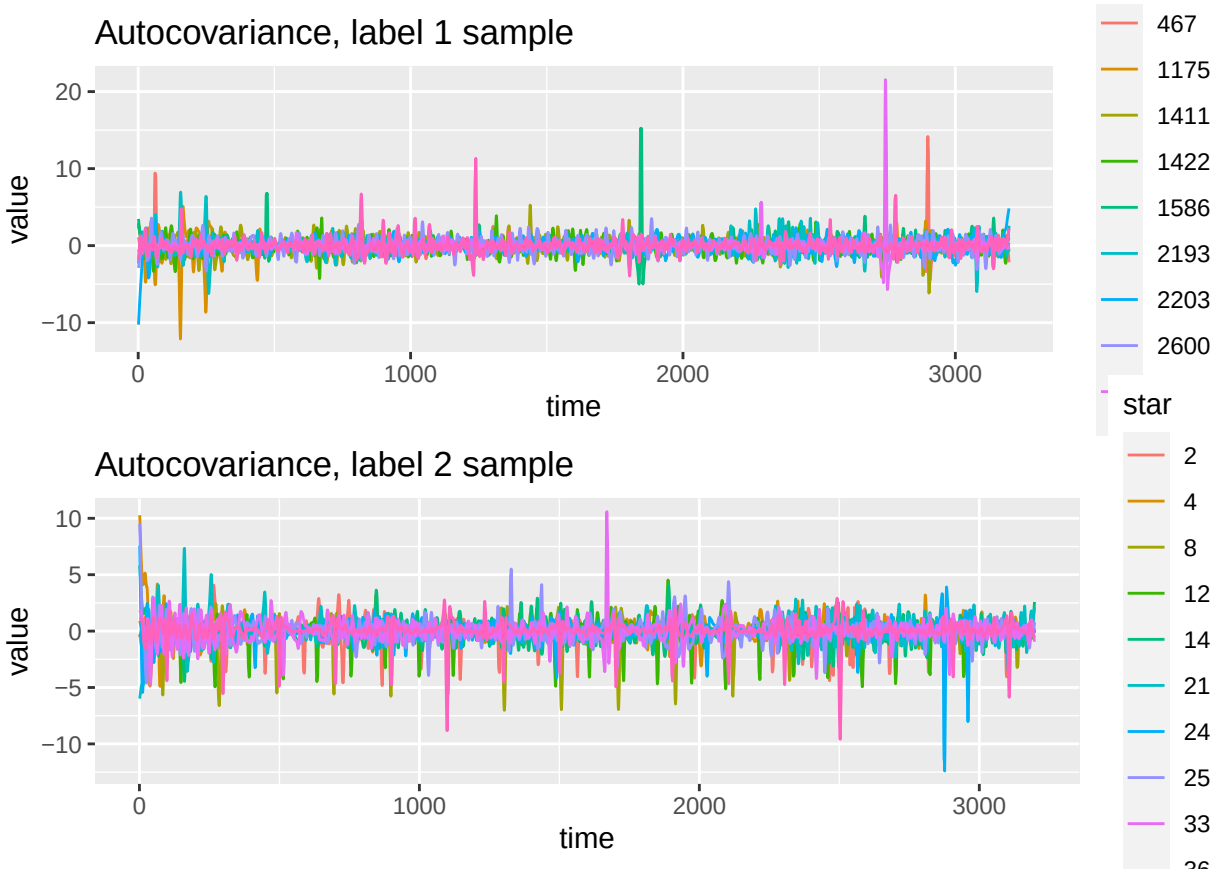


Autocorrelation

An alternate approach to the DFT would be to compute the autocorrelation of the signal for each star:

$$C(k) = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\sigma^2}$$

This is done using the **pacf** function of the stats package.



This metrics is not studied deeper, yet, in the present document.

Machine learning methods

In this section we are going to assess the predictions using several machine learning techniques and methods. The predictors X are the components given by the FFT analysis. The models will have to predict the labels Y given X.
Our first step is to divide the training set between two parts, which is known as *cross-validation*:

- one part will be used for the training of the model,
- the other part will be used to check the effectiveness of the method on a separate set of data.
  Then we are going to see how to reduce the dimensionnality of the problem using *Singular Values Decomposition* (SVD). The SVD will be built using the training part of the cross-validation set. The principal components of the SVD will feed the models to build the predictions.
  Finally we will assess our models on the test dataset which was originally loaded in the separate file testdata.csv.

Cross-validation

This step proved to be a very important one.

Initially I divided the training set in a 80/20 ratio, therefore using 4000 records to train the model, and approximately 1000 records to evaluate the predictions. With that method, all the tried models failed to detect positive records. They predicted almost only label 1 records.

Indeed, the amount of label 2 data is so rare that it is not surprising the SVD cannot take these elements into account.

The approach I finally adopted was to use a very reduced set of records to perform the SVD. I made sure this reduced training set contained a sufficient amount of labels 1 and 2. This way, the SVD takes into account the label 2 category in a more efficient way.

Here are the number of stars in each category for the validation and testing data sets:

count	training	testing
total	60	5027
label 1	40	5010
label 2	20	17

Thus, we randomly chose 20 positive stars and 40 negative stars, to build a training set of 60 records. This choice has no theorical justification, but this way we have a roughly balanced dataset, with 33% of positive stars, compared with the initial 0.7%.

SVD factorisation

The number of FFT components is 1598. Our concern is to reduce the number of components. We use the **Singular Values Decomposition** (SVD) algorithm.

SVD computing is applied on the $X_{train} = X[crossindex]$ matrix (which is the fft analysis restricted to the cross-validation training subset), after centering X columns.

$$centered(X_{train}) = U.D.V^T$$

To reduce the number of components, we keep the first K out of N values of diagonal matrix D, so as to preserve 99% of the total variance of X.

$$\sum_{i=1}^{i=K} D_i^2 = 0.99 * \sum_{i=1}^{i=N} D_i^2$$

```
## [1] "SVD, dim of X : " "60"               "1598"
## [1] "Dim of U : " "60"          "60"
## [1] "Dim of V : " "1598"        "60"
## [1] "Len of D : " "60"
## [1] "Number of factors  : " "49"
## [1] "Covered variance   : " "0.988887346784355"
## [1] "Dim of Ur : " "60"           "49"
## [1] "Dim of Vr : " "1598"         "49"
## [1] "Len of Dr : " "49"
## [1] "Dim of Z.train : " "60"                "49"

## [1] "Dim of X.test : " "5027"             "1598"
## [1] "Dim of Z.test : " "5027"             "49"
```

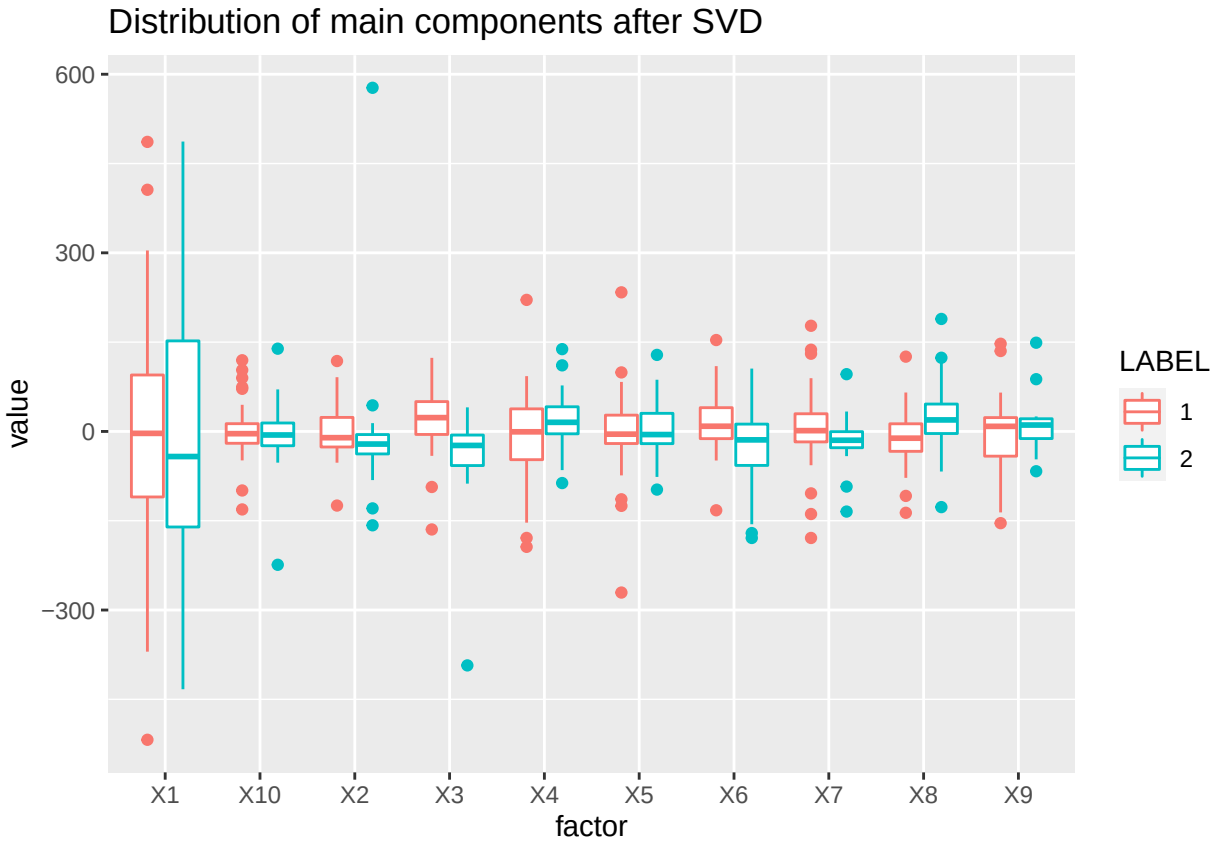
The number of components is finally 49. We computed a projection on this reduced set of components :

$$Z_{train} = X.V_r = U_r.D_r$$

We also compute the projection of the testing set $X_{test} = X[-crossindex]$:

$$Z_{test} = center(X_{test}).V_r$$

The following boxplot shows the distribution of the 10 first Z_{train} components for both categories of stars. The components number 3 and 6 seem to be particularly interesting to consider if we want to classify the points. This is shown also by the next scatterplots, which help to visualise the possibility of separating the two classes when considering particular components two by two.



Model training and ensemble approach

The ensemble approach consists in training several models, then to compare their output. The method used for one model is illustrated here for a **glm** model.
Reminder: `Z.train` was obtained from the training subset `train_fft[crossindex]` while `Z.test` was obtained from the crossvalidation subset `train_fft[-crossindex]`.

```
fitted_model <- train(y = train_fft$LABEL[crossindex] , x = Z.train,  method = "glm")
```

The model fitted on Z.train is then used to make predictions on the test dataset Z.test :

```
pred <- predict(fitted_model, Z.test )
confusionMatrix(
    data = factor(pred, levels=c("1","2")),
    reference = train_fft$LABEL[-crossindex])
```

That logic is applied for this list of classification algorithms :

- glm : Logistic Regression Model
- lda : Linear Discriminant Analysis
- naive_bayes : naive Baysian classifier
- svmLinear : linear Support Vector Machine
- knn : K Nearest Neighboors
- rf : Random Forest

Results

Cross-validation results

Here are the confusion matrices computed for each model:

```
## [1] "glm"
##            Reference
## Prediction    1    2
##          1 3108    3
##          2 1902   14
## [1] "lda"
##            Reference
## Prediction    1    2
##          1 3947    1
##          2 1063   16
## [1] "naive_bayes"
##            Reference
## Prediction    1    2
##          1 3227   10
##          2 1783    7
## [1] "svmLinear"
##            Reference
## Prediction    1    2
##          1 3721    6
##          2 1289   11
## [1] "knn"
##            Reference
## Prediction    1    2
##          1 4929   14
##          2   81    3
## [1] "rf"
##            Reference
## Prediction    1    2
##          1 4572    5
##          2  438   12
```

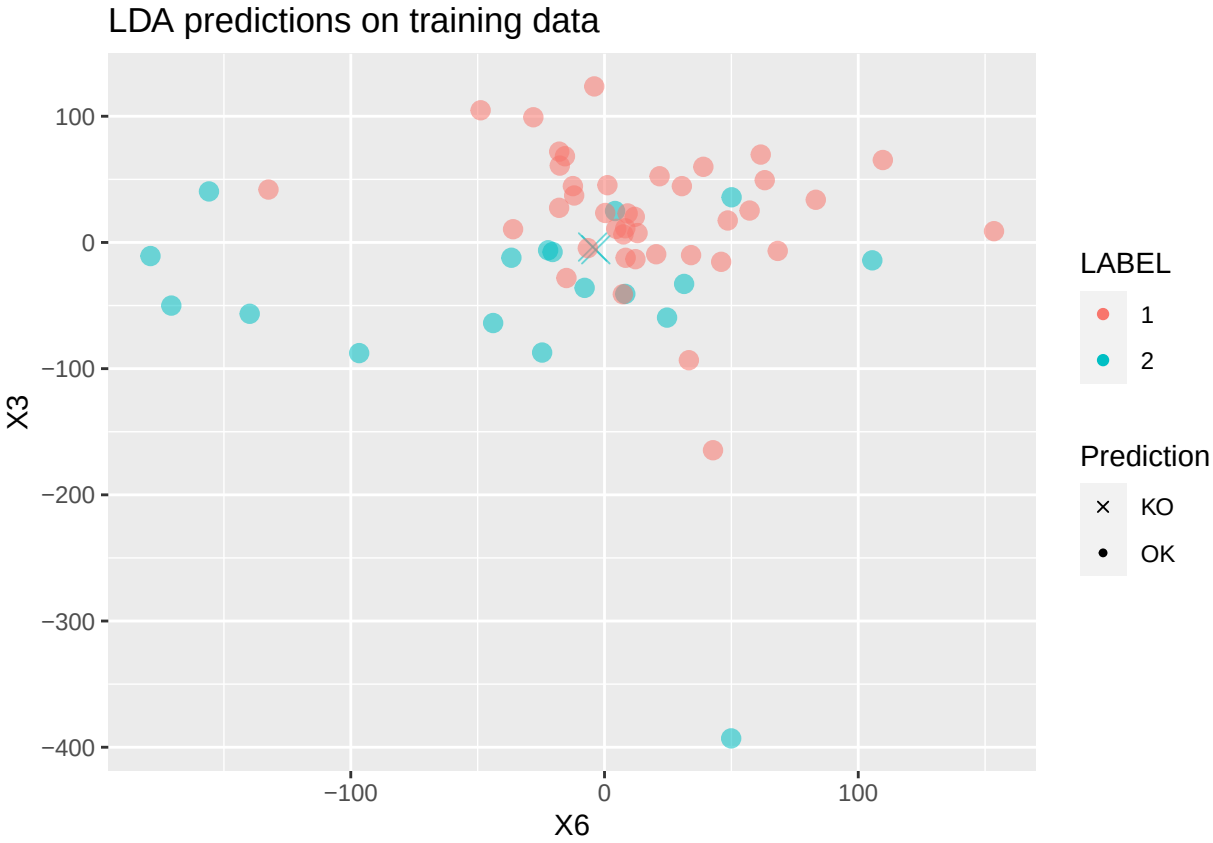
For instance we can see that the random forest (rf) algorithm correctly predicts 12 labels "2" and 5 labels "1" out of a total of 17 stars which are really labels "2".
Following are the F1 scores, sensitivity and specificity of each model, during the evaluation on the cross-validation set:

model	Sensitivity	Specificity	F1
glm	0.6204	0.8235	0.7654
lda	0.7878	0.9412	0.8812
naive_bayes	0.6441	0.4118	0.7826
svmLinear	0.7427	0.6471	0.8518
knn	0.9838	0.1765	0.9905
rf	0.9126	0.7059	0.9538

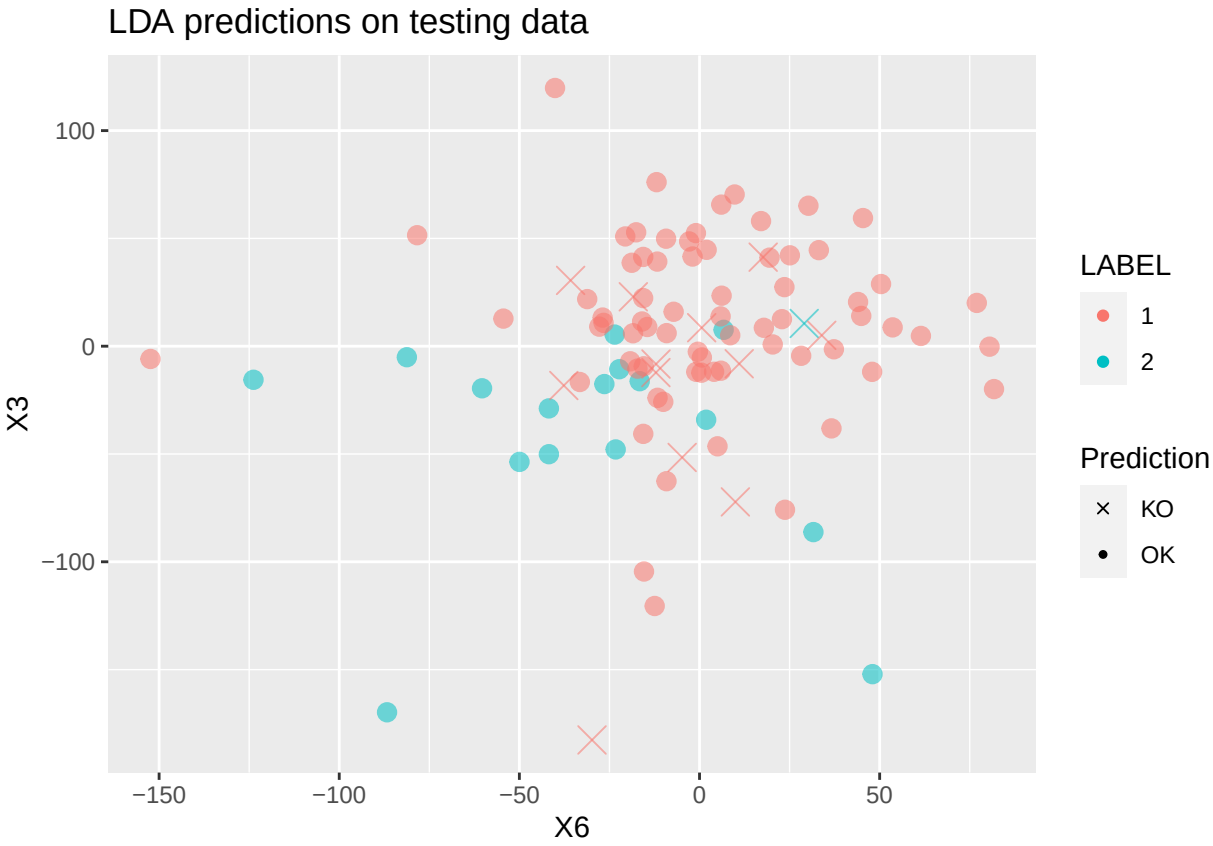
Naive Bayesian and KNN, though having a good F1 score, have poor ability to predict the labels 2, while glm and LDA are very efficient on this scope.

Focus on LDA training

Let us illustrate the result of the LDA algorithm. The first following scatterplot shows the predicted class for each points in the training and validation dataset, displayed along the SVD components 3 and 6. Classes 1 and 2 are shown in colors red or blue, while shapes tell if the prediction is correct (bullet) or wrong (cross).



Now this is the same plot for the testing data, restricted to the labels 2 and a subset of the labels 1, in order to be readable.



As can be seen, there are a few bad predictions for the labels 1, but only one bad prediction for labels 2. This is indeed what we are trying to achieve: we prefer not to miss labels 2, i.e. having false negatives, even if we have a few labels 1 predicted as labels 2, i.e. false positives.

Predictions on the testing dataset

Considering now the *testdata.csv* input, we have to apply the following transformations :

- normalize the rows (center and reduce)
- filter the low frequencies (high pass 1st order)
- smooth the rows (loess)
- spectrum analysis (FFT)
- project the centered columns onto the previously defined 49 principal components (SVD)

```
## [1] "Dim of X.test : " "569"              "1598"
## [1] "Dim of Z.test : " "569"              "49"
```

Then we are able to compute predictions, using the previously trained models, on this test dataset. We can look at the confusion matrices, specificities, sentivities and F1 scores:

```
## [1] "model :" "glm"
##            Reference
## Prediction   1   2
##          1 347   1
##          2 217   4
```

```
## [1] "model :" "lda"
##           Reference
## Prediction   1   2
##          1 441   1
##          2 123   4
## [1] "model :"     "naive_bayes"
##           Reference
## Prediction   1   2
##          1 414   5
##          2 150   0
## [1] "model :"   "svmLinear"
##           Reference
## Prediction   1   2
##          1 419   3
##          2 145   2
## [1] "model :" "knn"
##           Reference
## Prediction   1   2
##          1 559   3
##          2   5   2
## [1] "model :" "rf"
##           Reference
## Prediction   1   2
##          1 527   4
##          2  37   1
```

model	Sensitivity	Specificity	F1
glm	0.6152	0.8	0.7610
lda	0.7819	0.8	0.8767
naive_bayes	0.7340	0.0	0.8423
svmLinear	0.7429	0.4	0.8499
knn	0.9911	0.4	0.9929
rf	0.9344	0.2	0.9626

Due to the very low number of labels 2, it is not straightforward to qualify the results, because missing a single label "2" means a loss in accuracy of 20%. We can see that only GLM and LDA have good results at predicting labels 2.
Compare with the results on the cross-validation set:

model	Sensitivity	Specificity	F1
glm	0.6204	0.8235	0.7654
lda	0.7878	0.9412	0.8812
naive_bayes	0.6441	0.4118	0.7826
svmLinear	0.7427	0.6471	0.8518
knn	0.9838	0.1765	0.9905
rf	0.9126	0.7059	0.9538

Some models are relatively consistent, such as glm and lda, while other models have much poorer results, as naive bayes and random forests. We might think that for this problem some models have better generalisation capacities.

Ensemble prediction

We now try to predict the outcomes by taking into account each of the models predictions.

glm	lda	naive_bayes	svmLinear	knn	rf	star
2	2	1	2	2	2	1
2	2	1	2	2	1	2
2	2	1	1	1	1	3
1	1	1	1	1	1	4
2	2	1	1	1	1	5
1	2	1	1	1	1	6

Giving the same weigh to each model, we consider predicted label by each model, and we keep the value that is predicted by more than 50% of the models.

star	score_2	score_1	outcome
1	5	1	2
2	4	2	2
3	2	4	1
4	0	6	1
5	2	4	1
6	1	5	1

This gives the following confusion table and scores:

```
##            Reference
## Prediction   1   2
##          1 446   3
##          2 118   2
```

```
## Sensitivity Specificity          F1
##   0.7907801   0.4000000   0.8805528
```

We can see that the specificity is not so good with 40%. This is because a majority of models are bad at guessing labels 2.

During training we saw GLM and LDA models were performing better. If we retain only these two models predictions, we have this:

```
##            Reference
## Prediction   1   2
##          1 331   1
##          2 233   4
```

```
## Sensitivity Specificity          F1
##   0.5868794   0.8000000   0.7388393
```

Keeping these last two models, we correctly predict 80% of the labels 2, but with more than 40% of errors on labels 1 this is not very interesting in the end.

Conclusions

We explored the dataset and focused on some sample stars of each classes, and applied some *transformations*.
The *normalisation*, aka centering and reduction of the values across time for each star, removes the large discrepancies we can see on the original data.
A *detrending* has been done, by means of filtering the low frequency components of the individual signals. This should discard the contribution of events longer than 36 hours, and that we think are not related to the dimming of light caused by a planet.
A *smoothing* has also been applied, with the aim to discard this time the high frequency components, that we consider as noise.
We applied a *FFT* to analyse the components of the frequency spectrum, rather than the time values.
A *SVD* transformation helped to reduce the number of components.
A major difficulyt in this project is the low incidence rate of the label 2 stars (exoplanet stars). To deal with this, the SVD was applied on a reduced set of data, while making sure label 2 incidence was sufficiently important in this reduced dataset.
This way the trained algorithms gave interesting results, though far from perfect. Applying our models on the test data set, we are facing a difficulty to isolate exoplanet stars (labels 2). Algorithms that perform well in detecting labels 2 also wrongly classify many labels 1. At best these algorithms could help to discard a part of the non-exoplanet stars, and even this would not be perfect.

Among directions for *future works*, we might study more precisely the impact of the data transformations. When filtering high and low frequencies, there might be optimal cutoff values.
We should have a closer look at those labels 2 stars that could not be classified, and see if there could be an evidence of interesting new characteristics.
We should also probably work on the crossvalidation, since the selection of the training set must have a strong impact on the quality of the calibrations.
Beyond FFT analysis, we should consider other statistical measures of the signal, such as covariance, just to quote an example.
We did not tried either to optimise the algorithms, and this could be an important axis of progression: number of neighbours for KNN, depths of tress for random forests, kind of regularisation, kinbd of kernels, etc.