UNIVERSITÀ DEGLI STUDI
DI SALERNO

Sommario

1. First task

1.1. Analysis of the data

The dataset provided is formed by 80 samples and 45 predictors, while the output is scalar.

```
> dim(Dataset)
[1] 80 46
```

Fig. 1 Dataset size

The task requested to split the dataset into training and test set respectively with 80% and 20%. The problem is quite high-dimensional with a number of predictors that is similar to the number of samples in the training set. The techniques used are chosen to fit this type of problems. The high-dimensional space exposes a model to fit extremely on the training data, reducing the bias error but increasing the variance. Apart from this, an operation really complex, in a small set of data like this, is the detection of the outliers. In our case, we have a predictor range of the order of tens, while for the y it is between -30000 and 40000; consequently, with a big range like this for the response, a very small range for the predictors and only 80 values it is really difficult to detect unusual values (outliers).
Another possible problem to take care of is the correlation between the predictors; in a high dimensional problem it could be dangerous to fit a model.
We report here the results and the analysis done regarding those problems:

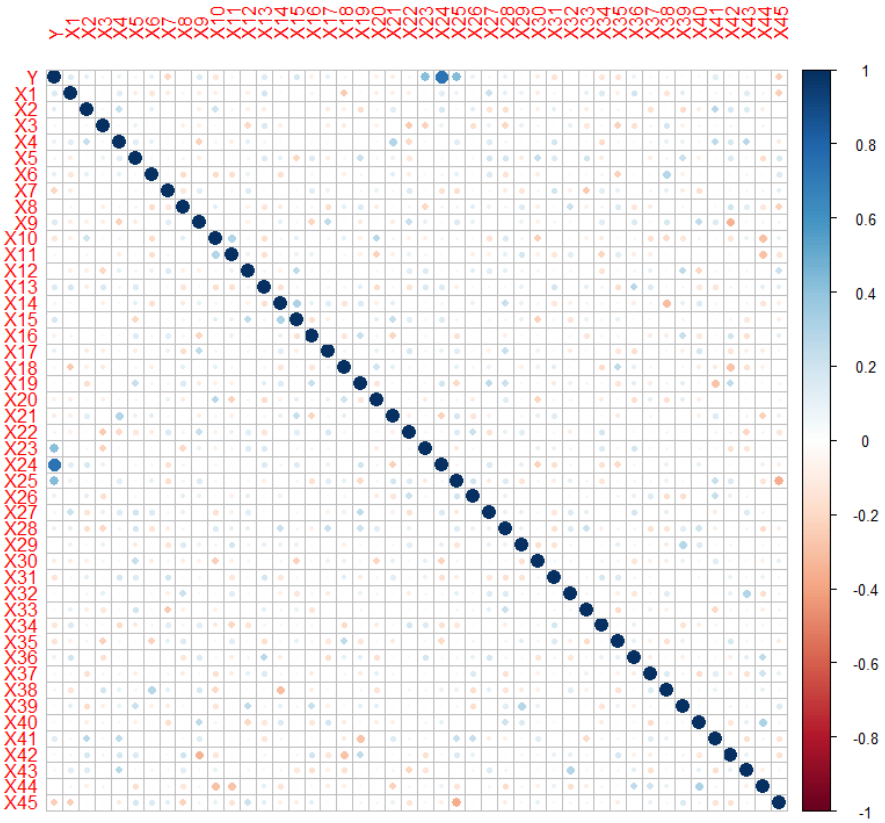


Fig.2 Correlation Matrix

We could see from the plot that there aren't variables extremely correlated, in fact the values for correlation are not so high. With that number of predictors, the probability that some variables could be written like linear combination of others is high.
But instead it is possible to see that there are some variables really correlated to the response.

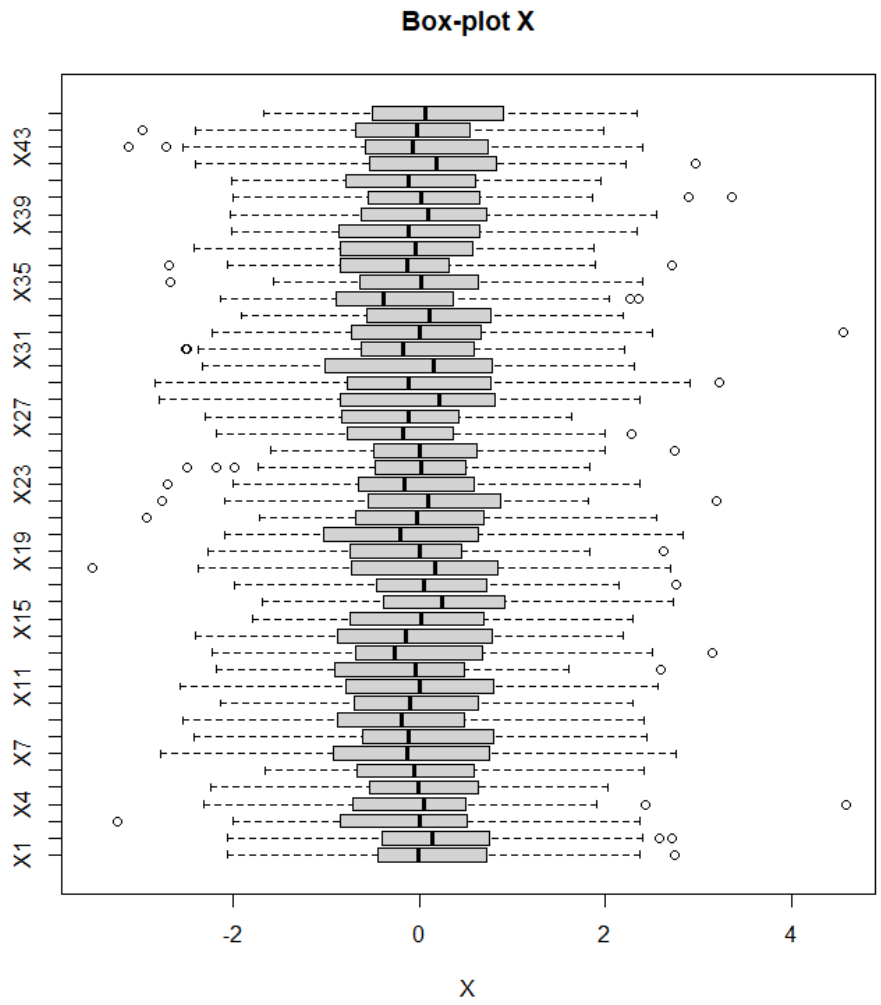


Fig. 3 Box plot on Predictors

From the box plot we could notice that albeit the predictors are not normalized, they vary in similar ranges.

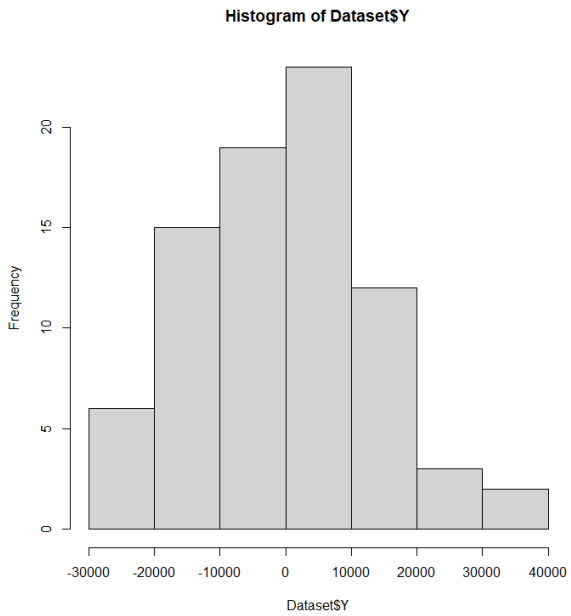


Fig. 4 Box Plot of response

For the response's histogram, we could notice that the range in which the values vary is really high and that is a problem with 80 predictors. In fact, considering that the predictors have a lower range, these points could result so distant although they are near in the feature space.

1.2. Analyzed approaches

We have analyzed data using several regression techniques. In particular we also tried to apply techniques that are commonly used in predictive modeling, especially in case of high dimension problems for their robustness through addition of bias terms. We have analyzed for each of them the characteristics in their formulation and the results that they obtain in our instance. We have conducted different studies in order to understand which approach is the most effective in modelling the relationship between the independent variable and the dependent ones.

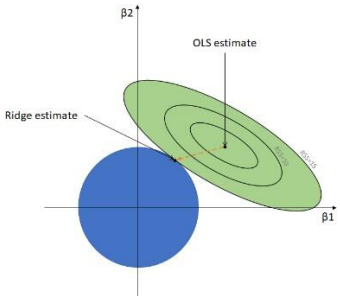
First three applied methods are regularization techniques that add a penalty term to the residual sum of errors loss function. This penalty term shrinks the coefficients towards zero. The degree of shrinkage can be controlled by a hyperparameter called the regularization strength (lambda), which determines the trade-off between model fit and simplicity.

I. **Ridge Regression**:

Ridge regression has a penalty term equal to:

$$\lambda \sum_{j=1}^{p} \beta_j^2$$

This term is proportional to the sum of squares of the coefficients. It shrinks the coefficients towards zero, but it does not set any coefficients to exactly zero; this permits it to perform theoretically better in high dimension when almost all the predictors are significant for the output. A graphical visualization of his behaviour is:



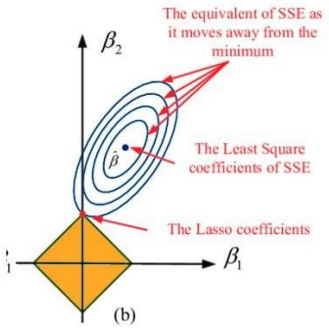
The assessment of the model is realized through a k-fold cross-validation. In the predictors we have some of them that are quite correlated; this approach will have difficulties to remove them during the shrinkage.

II. Lasso Regression:

Lasso regression has a penalty term equal to

$$\lambda \sum_{j=1}^{p} |\beta_j|$$

This term is proportional to the absolute value of the coefficients. The objective of the Lasso penalty is to include bias in the model itself, in a way to contrast situations where the variance is really strong. Lasso also can tend the beta coefficients to zero, so it performs a variable selection tending the less important beta to zero before the others. This characteristic makes the model sparse and can help to identify the most important predictor variables. A graphical visualization of the constrained problem is:



The assessment of the model is done through a k-fold cross-validation. We expect that in this

situation, the result from this constrained minimization will be optimal.

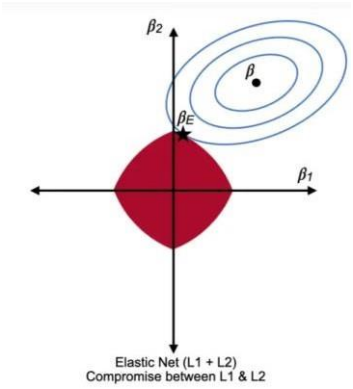
III. ElasticNet Regression:

ElasticNet regression is a combination of Ridge and Lasso regression, which adds a penalty term to

$$\lambda \sum_{j=1}^{p} (\alpha \beta_j^2 + (1-\alpha)|\beta_j|)$$

the residual sum of squares loss function that is a weighted sum of the sum of squares of the coefficient and the absolute value of the coefficients. This penalty term combines the strengths of Ridge and Lasso regression and can shrink the coefficients and set some coefficients to zero simultaneously. The degree of shrinkage and variable selection can be so controlled by two hyperparameters (alpha and lambda):

A graphical visualization of the constrained problem is:



The assessment of the model is realized through a k-fold cross-validation. We expect that with this approach we will have good responses, getting the advantages of the two above mentioned approaches.

IV. Subset selection with AIC and BIC:

This approach has been realized doing variable selection with an estimate of the test error, so with mathematic adjustments techniques on the training error, not using effectively the test samples. The subset selection has been conducted with a hybrid approach (possibility of adding variables and remove them during the different steps) estimating the test error before through the AIC and consequently also through BIC to evaluate both an estimate that penalizes less the number of predictors like AIC, either one that results more stringent like BIC.

We have also evaluated performance on linear regression through OLS, knowing its limits in this environment, but only to have a benchmark for evaluations:

V. Linear Regression:

Linear regression is a simple and straightforward approach to modeling the relationship between a dependent variable and one or more independent variables. The method estimates the coefficients of the independent variables by minimizing the sum of the squared differences between the observed and predicted values of the dependent variable. In our analysis, we fit a linear regression model to the data and evaluate its performance using measures such as mean squared error and R-squared.

To evaluate the performance of all these described techniques, we used a dataset with multiple predictor variables, and we compared the variable selection produced, and generalization performance of each approach. The variable selection can be evaluated by comparing the magnitude of the coefficients and the number of non-zero coefficients for each approach. Predictors with low p-values and high regression coefficients are likely to be the most significant contributors to the model.

The generalization performance can be assessed by splitting the data into training and test sets, and by

comparing the prediction performance on the test set for each approach.
In conclusion, this section provides an overview of the analyzed approaches, Ridge, Lasso, ElasticNet and linear regression with subset selection, and illustrates how they can be applied to linear regression to reduce the complexity of the model, improving the model fit, and enhancing the generalization performance. By comparing the performance of these techniques, we can gain insight into the strengths and weaknesses of each approach and choose the best technique for the specific data analysis problem.

1.2.1. Evaluation of results of different approaches

The first performed step was to gather and pre-process the data. This includes cleaning and transforming the data, handling missing values and outliers, and transforming the variables if necessary.
Once the data is prepared, the next step was to choose the appropriate model.
After choosing the model, the next step was to fit the model using the R Studio tools and perform statistical tests to check the goodness of fit of the model. The goal is to determine if the model provides a good explanation of the relationship between the variables and if it is a useful tool for predicting future values of the dependent variable. Finally, the results are analyzed and interpreted.

Linear Regression

The main objective of the linear regression approach is to fit a line through the data points that best describes the relationship between dependent and independent variables.
In this case, a multiple linear regression model is used to model the relationship between the independent variables and one dependent variable.
The regression coefficients and other statistical measures such as R-squared are used to evaluate the quality of the model and to evaluate the capability in making predictions about future values of the dependent variable.
In this specific case, after fitting the linear regression model using the appropriate RStudio command, it was possible to assess the significance of each regressor by calculating the associated p-values. From this analysis, it was apparent that most of the predictors used in the model have a very high p-value, indicating that they are not significant contributors to the model, indicating that probably an approach of variable selection is required in order to extract only the more significant predictors.
In this analysis we report a R-squared that is very high, it means that the variance of the model has been fully explained (but it is evident because we utilize all the predictor). Also, the value of F-statistic is very high (9.004e+04).
VIF (Variance Inflation Factor) is a measure of the amount of multicollinearity in a multiple regression model. It is calculated for each predictor variable in the model and a high value of it, means that there is collinearity in the model, and this can lead to unstable and unreliable coefficient estimates. In such a case. By using the VIF metric, we can assess the impact of multicollinearity on the model fit and make informed decisions on predictor selection.
So, we utilized the command “VIF” to check for multicollinearity, as it is possible to see the values of Variance Inflation Factor, there are no values that exceed 5 or 10 and that indicates a problematic amount of collinearity (showed also in the correlation matrix reported).

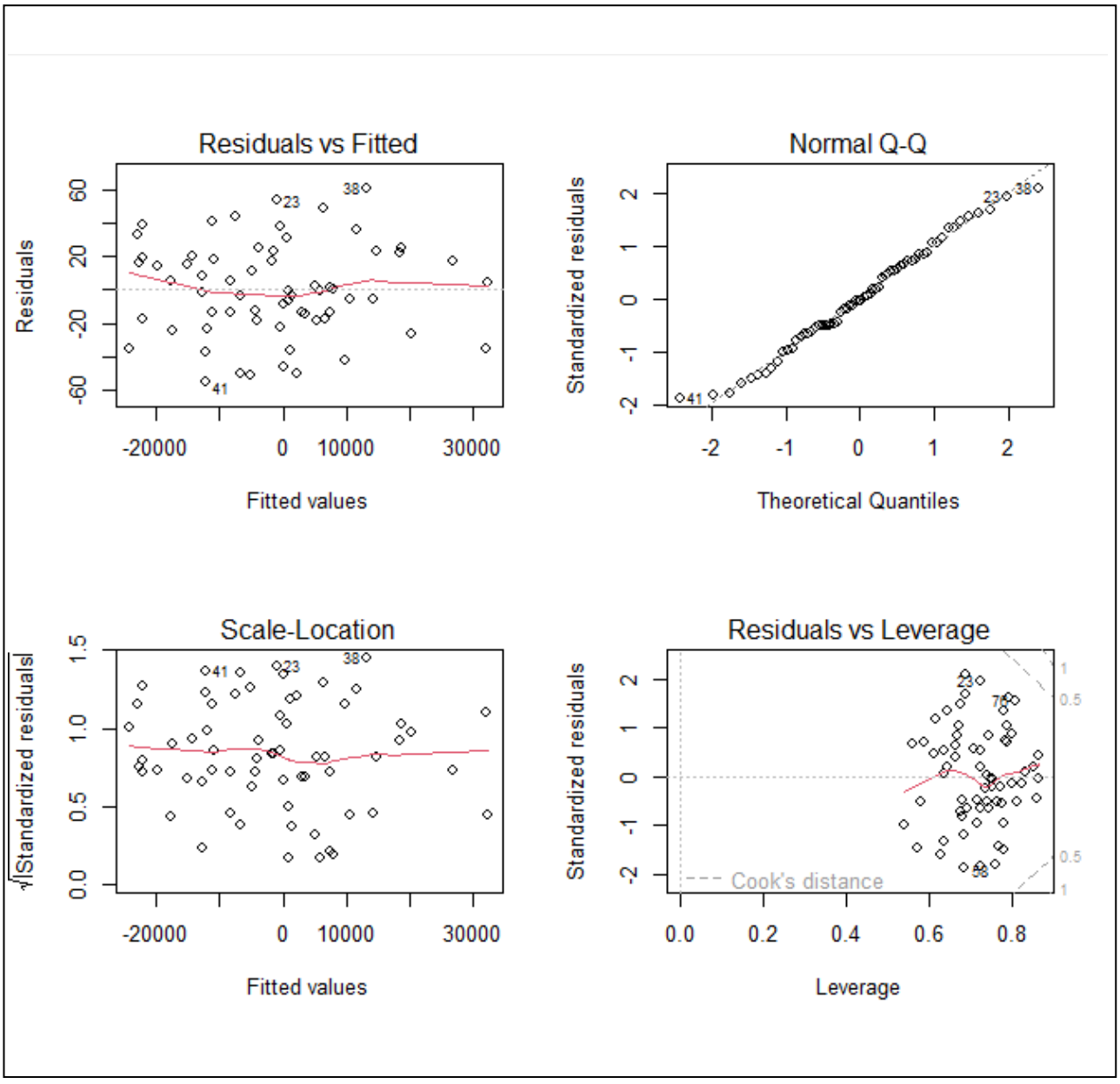
VIF

```
> vif(fit)#check for multicollinearity, as it is possible to see the values of Variance Inflation Factor, there are no values that exceed
 or 10
      X1       X2       X3       X4       X5       X6       X7       X8       X9      X10      X11      X12      X13      X14      X15
4.977506 2.932073 5.044866 3.524720 2.863515 2.997604 2.226957 3.170692 3.265104 4.249905 2.907709 2.972854 2.367329 5.673955 4.700973
     X16      X17      X18      X19      X20      X21      X22      X23      X24      X25      X26      X27      X28      X29      X30
3.414875 3.148252 5.138275 3.981436 4.106939 2.672978 4.195215 2.642777 2.977307 3.766876 2.954246 2.526597 2.775771 2.140874 2.903458
     X31      X32      X33      X34      X35      X36      X37      X38      X39      X40      X41      X42      X43      X44      X45
2.727851 1.831246 2.453006 2.933579 3.354334 3.774598 2.242355 2.950048 2.372696 3.398658 4.511663 3.723028 3.371018 4.162761 3.762292
```

Then we evaluated the error on the test set, this allowed us to evaluate the model's performance. As it is possible to see the MSE is very large, but it is reasonable according to the discussion about the significance of the predictors and the fact that this problem is quite high dimension.

MSE
260427747

The analysis of the linear regression approach has been used only as a benchmark because we were sure that this approach will not perform well in a situation like this.
Then we also evaluated different plots of the residuals, as it is possible to see in the image below.



The Q-Q Plot compares the quantile of the data to that of a normal distribution (default options) and how it is possible to see the plot is fairly aligned along the bisector. In particular it is a test of gaussianity because we are analyzing the residuals, and for different evaluations we assume that gaussianity is satisfied.
If this does not happens it means or that the initial error is not gaussian(another assumption we generally do), or that our model introduced an error so large that the quantiles were wrong.
From the bottom-left image it is possible to see that there are no high-leverage points because there are no points that exceeds Cook's regions. From the bottom-right image it is possible to see that there are some possible outliers. From the first image it is possible to see that there are not worrying situations about collinearity. Because of the fact that the problem is quite high dimension, probably the base regression problem is not a good approach. Because of this reason also other types of approach have been

tested.

Ridge Regression

To perform cross validation different values of k have been tried, but in the script is left only the best one.
The obtained results:
k=3 -> 9712.763
k=5 -> 9712.763
k=7 -> 9712.763
k=10 -> 9712.763

Starting from this value it is possible to see that for values of k that are 3,5,7,10 there are no differences between the test MSE.
For all values of K there are no differences, but anyway the test MSE is quite large, this because it is obviously a problem that requires variable selection.
Between these values we chose k=5. This because probably it is the right trade-off in terms of not having very many data to fit the model at each iteration, even if having larger validation set at each iteration, and of having a very small validation set at each iteration, that means having less qualitative estimates.
Following is represented a graph showing coefficients trend for different shrinkage values:

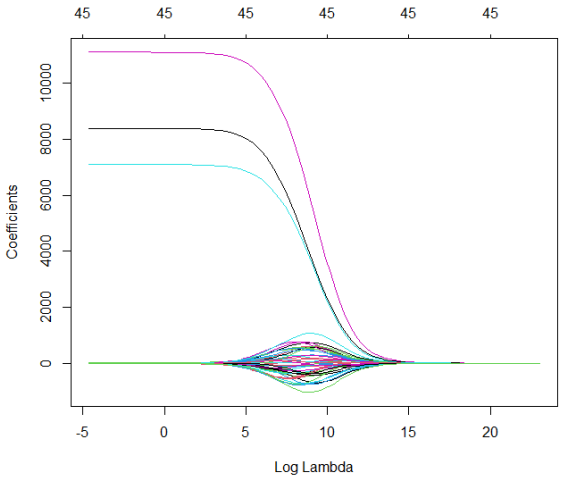


Fig. 5 Coefficients values on Lambda changes

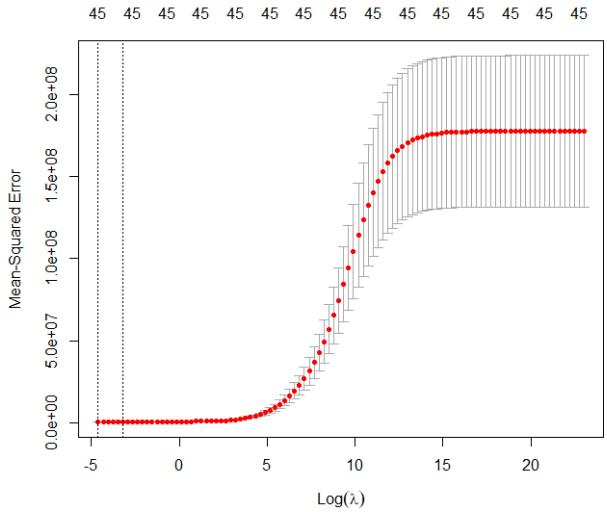


Fig.6 MSE changes

The reported results show that Ridge has a remarkable improvement compared to OLS. This is because the problem we faced with is very particular and therefore it would be more appropriate to treat it with shrinkage techniques, even if Ridge will prove to be the worst among the techniques tried. It is important to notice in the fig.6 that, even increasing the value of lambda, the number of predictors remain the same, because Ridge do not perform variable selection, but only shrinkages values of coefficients towards zero, this is the limit of this approach in this case. In this figure it is possible to see both the best value of lambda and the value according to the one-standard-error rule.

Lasso Regression

To perform cross validation different values of k have been tried, but in the script is left only the best one.
The obtained results:
k=3 -> 4478.796
k=5 -> 5992.016
k=7 -> 5551.427
k=10 -> 5551.427

Starting from these results it is possible to see that the best value is k=3. This because probably it is the right trade-off in terms of not having very many data to fit the model at each iteration, even if having larger

validation set at each iteration, and of having a very small validation set at each iteration, that means having less qualitative estimates.
Following is represented a graph showing coefficients trend for different shrinkage values:

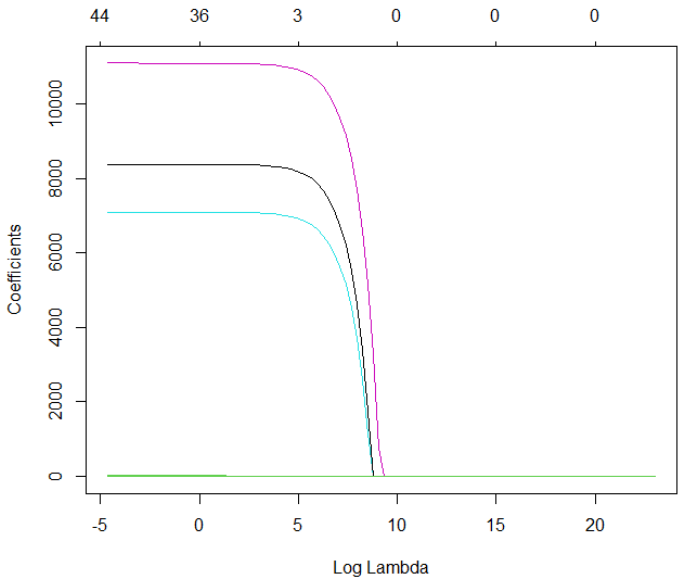


Fig. 7 Coefficients values on Lambda changes

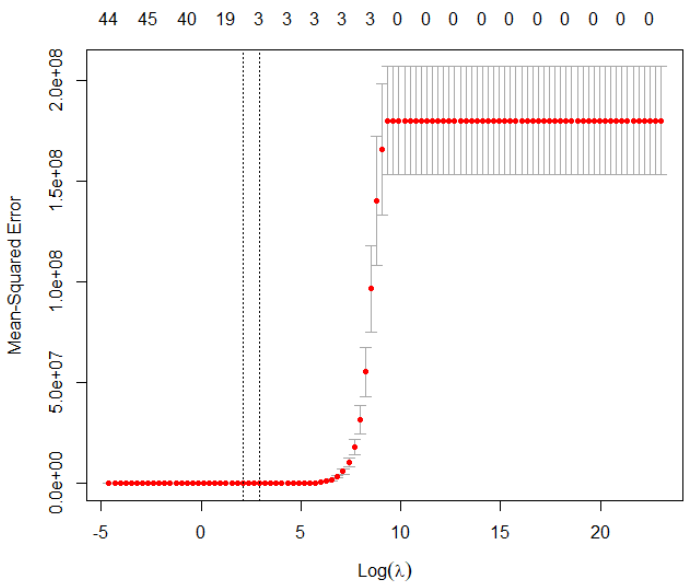


Fig. 8 MSE changes

It is important to notice that with this approach the values obtained on the test MSE are really smaller of both OLS and Ridge approach. So, as we expected, Lasso better fits this problem. In the fig.8 it is possible to see that the problem encountered for Ridge does not persist also for Lasso. In fact, starting from a certain value of alpha different coefficients became zero. In this figure it is possible to see both the best value of lambda and the value according to the one-standard-error rule, in particular this will lead to a number of regressor equal to three, that we noted to be the most significant.
Lasso performs better because we are quite in high dimension context, and so a selection of the predictors is necessary to obtain good values for bias-variance trade off, and consequently obtain a better generalization error.

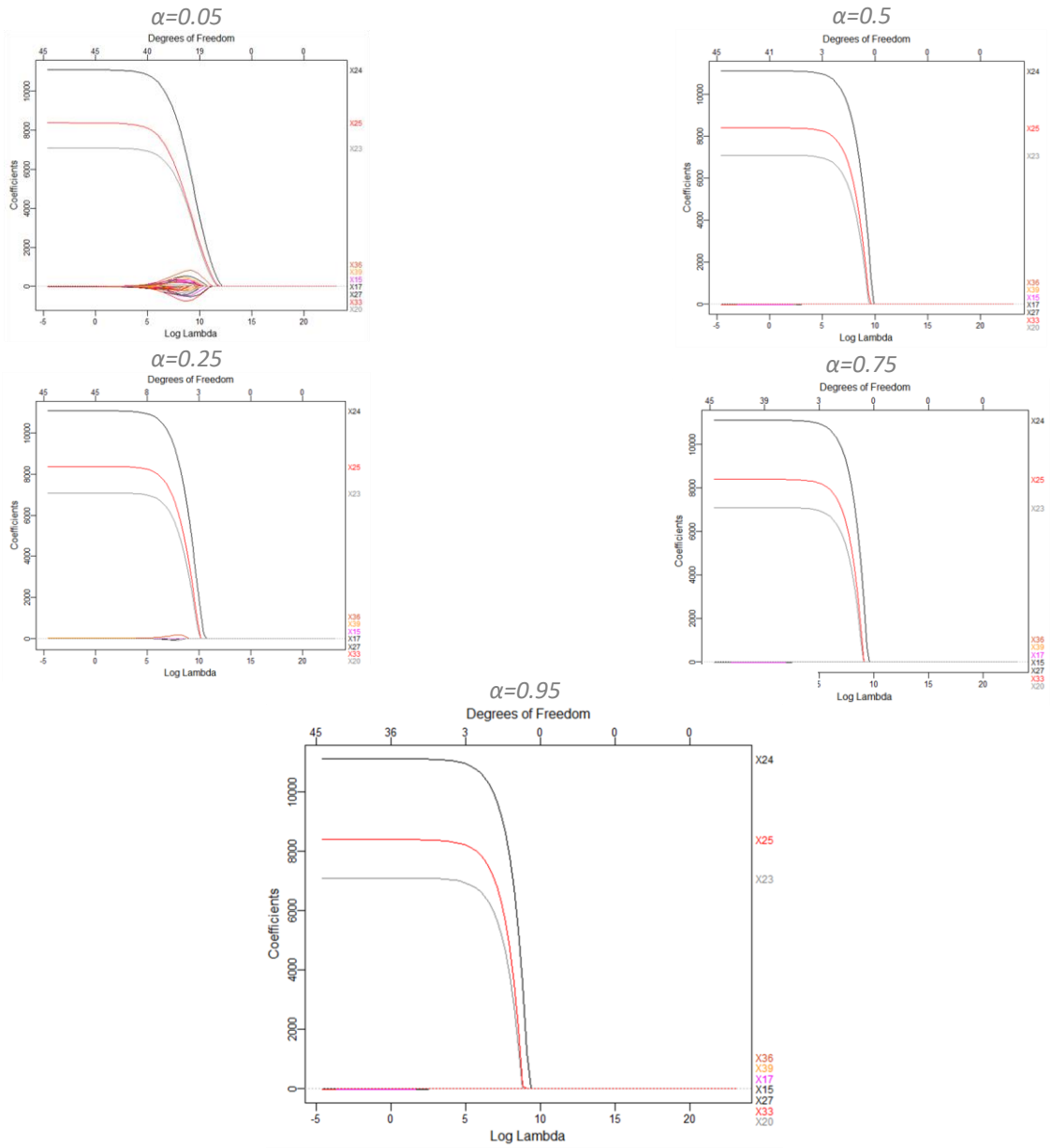
ElasticNet Regression

The optimal values of lambda and alpha are determined by comparing the performance of the model with different values of these two parameters.
The combination of Ridge and Lasso Regression in ElasticNet Regression provides a powerful tool for analyzing complex datasets and building robust and interpretable models.
To perform cross validation different values of k have been tried, but in the script is left only the best one.

The obtained results for different values of alpha:
k=3 -> 9755.636 9782.046 7659.451 5524.495 5132.915 4595.025 4478.796 , best_alpha = 1, num pred=10
k=5 -> 9755.636 9782.046 7231.490 6348.191 6052.819 5690.369 5992.016 , best_alpha = 0.95, num pred=16
k=7 -> 9755.636 9782.046 7231.490 5942.455 5637.127 5690.369 5551.427 , best_alpha = 1, num pred=16
k=10 -> 9755.636 9782.046 7231.490 5942.455 5637.127 5690.369 5551.427 , best_alpha = 1, num pred=16

Starting from these results it is possible to see that the best value is k=3. This because probably it is the right trade-off in terms of not having very many data to fit the model at each iteration, even if having larger validation set at each iteration, and of having a very small validation set at each iteration, that means having less qualitative estimates.
Starting from the obtained results it is possible to see that in general Lasso works better than ElasticNet, in fact Lasso corresponds to set the value of alpha equal to 1. This highlights that in a situation like this, in which visibly only a small subset of predictors has an higher value of significance, Lasso approach performs better, because do variable selection. So, Lasso performs better than Ridge because it is a situation in

which variable selection is required, and in fact an higher value of alpha means to tend more towards Lasso than Ridge.
The only exception is for k=5 for which, probably for the configurations of training and validation split, results that ElasticNet has better performance
respect to Lasso.
In particular analyzing the values of test MSE varying alpha, for k=5, it is possible to see a U-shape, that also allows to obtain for alpha=0.95 a more stringent selection.
Following is represented a graph showing coefficients trend for different shrinkage values:



Starting from these plots it is possible to see the trend of the shrinkage, and in particular how, increasing the value of alpha, the approach tends towards Lasso rather than towards Ridge, weighing more the L1 penalty than the L2. It is important to notice that increasing the value of alpha leads to a better performance of the shrinkage, and so this is in agreement with the analysis made for Lasso and Ridge, and so even with a weighing mechanism Ridge does not improve the final performances.

Subset selection with AIC and BIC

Among the different analyzed approaches, we also try an approach of selection using a hybrid approach, in particular the mixed selection that is hybrid because combine backward and forward steps. Among the subset selection methods, we chose this because better suits our problem. Specifically, the number of data is not very large, but however best subset selection is a very expansive technique. We have preferred not to use techniques such as forward or backward selection as they are less flexible respect to mixed selection, in fact backward cannot be used when the problem is an high dimension one, but both have the problem that during the different steps there is not the possibility to came back and a common situation is that for example some predictors became useless when some others are added and vice-versa.
During the different steps of selection, to evaluate the best model that is those with the best test error, we use two techniques that make an estimation of the test error making and adjustment on the training error: BIC and AIC.
In particular the obtained results suggest that accordingly to AIC the number of predictors to be used is 11, instead accordingly to BIC a small number is required, and in particular 9. This because depending on the equations used to compute these two parameters, when the number of samples is greater then 7, BIC generally places a heavier penalty on models with many variables.
The two obtained values of test MSE are quite good, especially that of BIC, which using a lower number of predictors, does not introduce large Bias in the model. In particular, accordingly to the model proposed by AIC, the test MSE is equal to 7281.48, for that proposed by BIC the test MSE is 7154.536, so a little higher. Also, only for test purposes we have also tried to increase BIC's penalty terms, and we noticed the performance increases a lot, selecting only three variables.

1.3. Chosen approach

Comparison between MSEs				
	Ridge	Lasso	ElasticNet	Subset selection AIC and BIC
Test MSE	9712.763	4478.796	4595.025 with alpha=0.95	AIC: 7281.48 BIC: 7154.536
N. of Regressors	46	10	10	AIC: 11 BIC: 9

From this table is possible to see how Lasso is the ones that performs better in terms of MSE in this environment, even if we saw the variable selection conducted from all this approach:

AIC variables

```
[1] 26
> > newvarAIC$coefficients
 (Intercept)          X13          X21          X22          X16          X14          X19          X27
-0.169199 10506.243148  9913.012536 10498.779934 12096.002688 11394.230027  9695.709355   278164
         X20          X15          X23          X24          X18          X12          X17          X33
11001.087141 11596.116834 10998.854765 10292.889587  6798.280799  6805.302993  3193.720463   158541
         X26          X42           X5          X40          X39          X37           X2
 -5.081536   -11.272921     8.102831   -15.738420     7.519648    -8.937148    -7.334638
         X31          X46          X44          X49          X10          X35
 -5.097805    -6.427071    -4.550874    -4.924640     5.213031    -4.492366
   9.037597    -6.851258
```

BIC variables

```
> print(newvarBIC$coefficients)#coefficient estimates
 (Intercept)          X24          X25          X23          X27
  -6.979347 11098.432691  8392.003655  7093.926770   -16.719628
         X29          X20          X39          X42          X33
  10.362818   -13.297284    13.630425   -11.320983   -10.686947
```

Starting from these results it is possible to see that the best value of test MSE has been obtaining with Lasso. This is reasonable because observing correlation matrix, it is clear that only a small subset of the regressors is really significant for the response, and so an approach that performs variable selection is required.

Ridge variables

	s1
(Intercept)	9.6495301
X1	-13.1137751
X2	2.8146869
X3	-8.6764674
X4	6.2798533
X5	7.4157832
X6	-9.9017262
X7	-0.8686902
X8	-5.0723596
X9	4.5059927
X10	5.4300107
X11	-0.6020291
X12	-11.6356541
X13	-7.5248003
X14	-17.5928921
X15	9.3941173
X16	-12.3043193
X17	2.5090459
X18	-12.6271856
X19	0.1856019
X20	-12.1074531
X21	3.1094735
X22	-16.2521605
X23	7095.1467584
X24	11098.4522133
X25	8398.1454511
X26	11.0339405
X27	-4.1741647
X28	-2.3048623
X29	4.5920555
X30	-3.6012688
X31	0.7218751
X32	-8.0756326
X33	-11.8224699
X34	7.0206927
X35	-7.5603455
X36	13.2528327
X37	2.7375202
X38	-7.8611630
X39	21.1009757
X40	-9.1317565
X41	-18.4489955
X42	-8.0807900
X43	-8.0176059
X44	-2.7922599
X45	-15.6731654

Lasso variables

	s1
(Intercept)	2.3170571
X1	.
X2	.
X3	.
X4	.
X5	.
X6	.
X7	-0.4191010
X8	.
X9	.
X10	.
X11	.
X12	.
X13	.
X14	.
X15	.
X16	-0.5453198
X17	.
X18	.
X19	.
X20	.
X21	.
X22	.
X23	7090.3154210
X24	11086.2901030
X25	8383.4087902
X26	.
X27	-0.1611993
X28	.
X29	.
X30	.
X31	.
X32	-2.3116137
X33	.
X34	.
X35	.
X36	1.1417781
X37	.
X38	.
X39	.
X40	.
X41	.
X42	.
X43	.
X44	.
X45	-0.7731021

ElasticNet variables

	s1
(Intercept)	2.4683104
X1	.
X2	.
X3	.
X4	.
X5	.
X6	.
X7	-0.9810662
X8	.
X9	.
X10	.
X11	.
X12	.
X13	.
X14	.
X15	.
X16	-1.0163436
X17	.
X18	.
X19	.
X20	.
X21	.
X22	.
X23	7090.4073129
X24	11086.1033929
X25	8383.7905413
X26	.
X27	-0.1181071
X28	.
X29	.
X30	.
X31	.
X32	-2.8182433
X33	.
X34	.
X35	.
X36	1.4270813
X37	.
X38	.
X39	.
X40	.
X41	.
X42	.
X43	.
X44	.
X45	-0.9941986

This is also confirmed by the observation of ElasticNet behavior varying the alpha parameter. In fact, alpha is a parameter that varies in the range 0-1 and weighs more Lasso penalty term or Ridge's one.
It is possible to see as much as alpha tends to 1 and so to Lasso, the performance improves.
This is because Ridge only shrinkage coefficients towards zero but does not set one of this to exactly zero.
In particular it is possible to notice from the script that also with stepwise hybrid selection approach, good performances have been obtained, especially with BIC that has a more stringent penalty respect to AIC.
Only for test purposes we have also tried to increase BIC's penalty terms, and we note the performance increases a lot, selecting only three variables.

1.4. Final results

Once the model and its parameters were selected using cross-validation and minimizing the test error, the coefficients were transformed to determine the suggestion.
The value of the selected coefficients was divided by 100 and rounded to the nearest integer, and then converted into ASCII code using each coefficient. The final result was used as a clue for solving the second part of the exercise. The suggestion turns out to be:

"GoT"

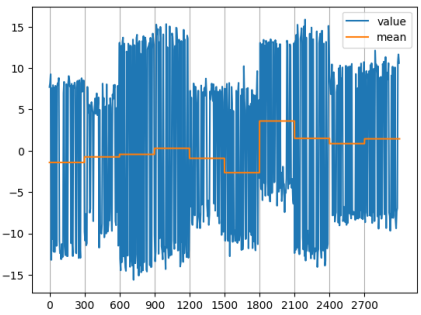
2. Second Task

For the classification task, we have been provided of a training set made up of pairs (Xi,Yi), Xi ∈ R^d and Yi ∈ {-1,1} and a test set; the dimension of the training is of 24 thousand of samples, with each X of dimension d=20 and the relative one-dimensional label Y, while the test set is made of 80 samples, with the same dimensions. The task was to firstly apply a dimensionality reduction of the training set to obtain data in a new lower dimensionality space; this has been done through the PCA. Then we had to apply the stochastic gradient descent algorithm analyzing different choices for the step-size. At the end, the obtained system had to make predictions on a new group of data, as a sort of test set, with a specified decision rule. Converting in ASCII the binary string obtained from the test observations classification, we acquired the result.

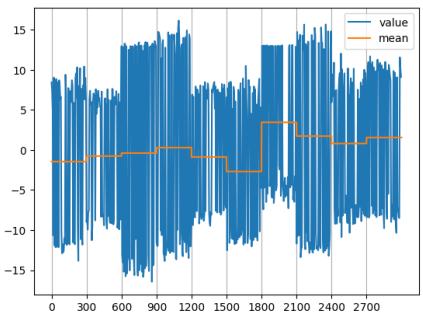
2.1. Analysis of the data

The training set is made up of 24 thousand of samples with a dimensionality of 20. So, our problem is not in a high-dimensional space that could have been very critical to work on. So, before the development of the system an exploratory analysis on the dataset has been conducted, considering the fact that it is made by different acquisitions over time. This preliminary step is fundamental to start the parameters estimation and also to understand the obtained results during each experiment.

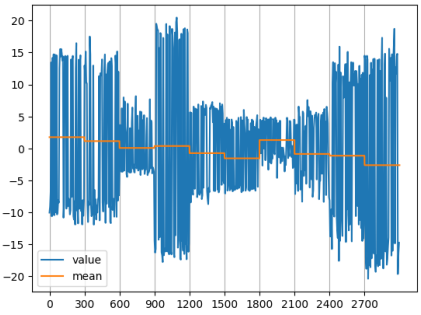
The first analysis was conducted in order to evaluate the trend of the features over time. Consequently, what has been done is to plot the different values of all the predictors during the different acquisitions. Because of the fact that the obtained results show an equivalent trend for each of these, following are reported only 4 of the predictors as a manifestation of what happens. It's possible to see from the graphs that the data seems to be not stationary as a whole, but rather stationary within a specific range of time intervals. In a further analysis, it seems clear that the range above mentioned is about 300 time intervals; these considerations are supported by the constant average value (plotted in the graphs) in the range and by the evident fluctuations of it among different ranges. The intensity of the fluctuations is different for each feature.



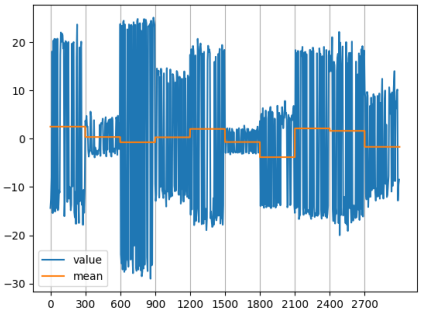
feature 0



feature 1



feature 2



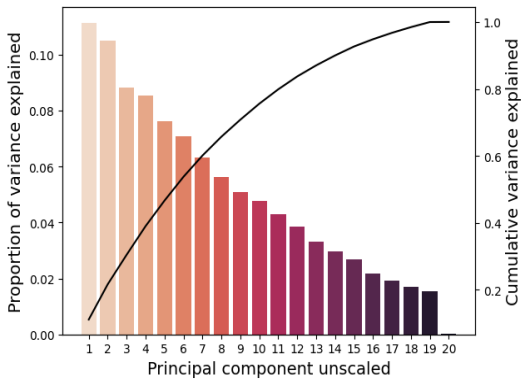
feature 3

2.2. PCA evaluation

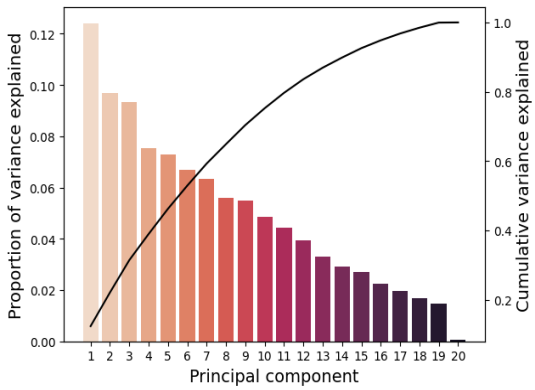
After analyzing the data we were dealing with, we try to perform a dimensionality reduction in order to exploit its benefits in terms of denoising and computational power savings keeping an equal, if not better, accuracy. To perform PCA there are several approaches that can be taken into account. The method we used is a manual implementation of the eigen decomposition. The dataset was centered subtracting the mean (it is necessary when a PCA reduction is applied).

The first choice that we have to perform is if scale the data or not, in fact calculating PCA on scaled and unscaled data can provide different results. When the data is unscaled, the PCA is sensitive to the scale of the variables, meaning that variables with larger scale will have a larger impact on the calculated principal components. Scaling the data, on the other hand, ensures that all variables are on the same scale and prevents variables with larger scale from dominating the calculation of the principal components.

For this consideration we choose to conduct the PCA on **scaled data** because often more robust and provides a more meaningful representation of the underlying structure of the data.



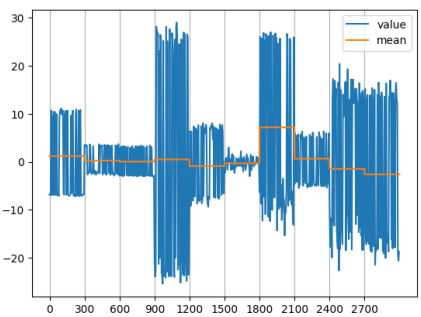
PCA on no scaled data



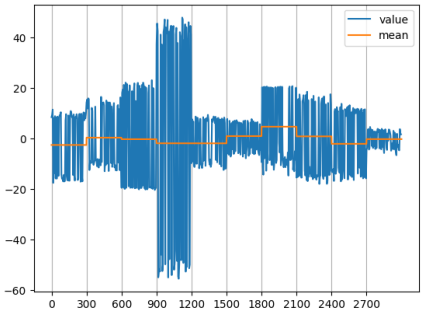
PCA on scaled data

These two plots illustrate the proportion of variance explained by each principal component, as well as the cumulative proportion of variance explained by all components. As we can see, the first few components explain a quite significant portion of the variance in the data (not in a really strong way), so we decided to take for our algorithm **5 principal components**, that we think could be a good tradeoff between the reduction of the dimensionality (20 -> 5) and the variance included (more than 50%).

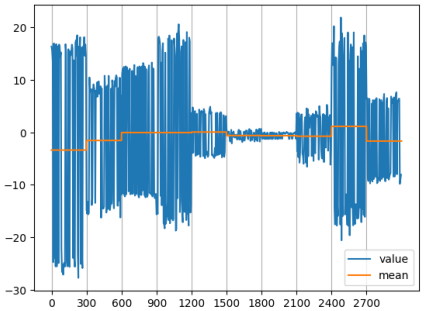
Obviously, the PCA projects the data in a new feature space in which each component is a linear combination of starting ones. So, to ensure that the hypothesis made on the starting features are preserved in this new space we repeated the analysis also on the obtained principal components. Surely, we expected that the trend would be preserved and also that the values fluctuations over time should be emphasized on the firsts principal components because they enclose the most of data variations.



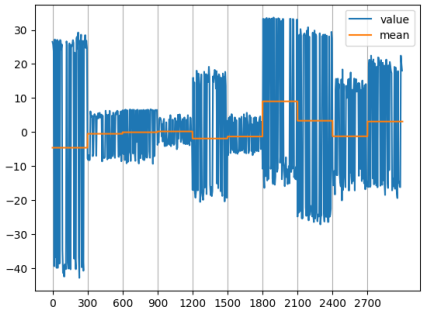
first principal component



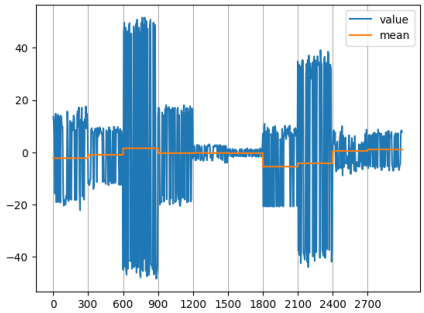
second principal component



third principal component



fourth principal component



Fifth principal component

2.3. SGD Analysis

To obtain a model capable of make predictions, a training procedure using the Stochastic Gradient Descent algorithm was implemented, considering as model the logistic regression. SGD is an optimization algorithm that is used to find the minimum of a cost function by iteratively updating the parameters of a model. The logistic regression model was used to perform binary classification on the data. Here are reported the used functions.

$$Loss\ Function = \log(1 + e^{-yx^T\beta})$$

$$Cost\ Function = E[\log(1 + e^{-Y\ X^T\beta})]$$

In the SGD the J(B) or Cost function is estimated online so only on each new sample (without the need of compute expected value). It changes a lot with respect to the classic Gradient Descent algorithms.
So, the cost function became:

$$eCost\ Function = \log(1 + e^{-ynewxnew^T\beta})$$

As said before it is only an estimate of the true Cost Function.
In order to update the parameters, at each iteration of the algorithm we compute the gradients of the "$eCost\ Function$":

$$Gradient = \frac{-y * x}{\log(1 + e^{-yx^T\beta})}$$

so, the updating rule is:

$$Bs = Bs - StepSize * grad(eCost\ Function(xnew, ynew))$$

The $StepSize$, also known as the learning rate, is a crucial hyperparameter in the stochastic gradient descent (SGD) algorithm. It determines the size of the updates made to the model parameters at each iteration of the algorithm. It affects the speed at which the algorithm converges and the stability of the convergence. We know that for SGD, keeping the step size parameter constant, do not achieve the optimum minima of the gradient but the parameters update jump into one of its surroundings (until a constant state error).

The strength, however, of this approach lies in the fact that changing the truth, the algorithm is able to fit the model on this new data, so we decide to walk this road, because though the previous analysis it is evident the non-stationarity of the data.
The other solution would be to set a dynamic step-size that we know achieve the local minima but at the same time has an elephant's memory, that is not able to change with a truth variation.
So we decided to implement the algorithm with a **constant step size**.

2.3.1. Step size selection

Based on what has been said above, we have established the non-stationarity of the data. Furthermore, we have seen that we could assume a “local stationarity” in a window of more or less 300 samples.
An analysis like that could be not feasible in a real online case. But in real situation knowing the physics of the studied problem, it could be at least estimable whether the data are stationary or not. So, what we verified numerically from the dataset, in real situations can be at least approximated, in order to pose it as an assumption to find future parameters. In addition, often is also possible to evaluates how often "a prediction" arrives and thus how many data we have for the model assessment before it.
In order to choose the step size of the SGD algorithm, we decided to exploit a relation between it and the required time of convergence (that is a known constraint for us); in particular we know that there is an inverse proportion, so bigger is the step size, smaller is the time of convergence of the algorithm. Now, thinking in a precise manner, this relation is also influenced by other parameters such as the one of the strongly convexity and the one relative to Lipschitz continuity of the gradients (both difficult to evaluate).
In a more qualitative approach, we have approximated this relation simply as: **time to converge inverse of the step size**.
So, we evaluated that the step size should be **0.01** in such a way that the time of convergence result about 100 step, a measure quite robust with respect the time of each evaluation (300 step) and the time where the truth probably remain the same (about 300 step).

2.4. Prediction on test set

The evaluation of the obtained model was also conducted by evaluating the predictions on test set. The classification value on test samples was calculated by multiplying the feature of each sample with the parameters estimated by the stochastic gradient algorithm at a specific time written in the relative row. So, the classification was performed online, at prefixed instant, during the execution of the stochastic algorithm and, as described in the requirements, assigning the value of 1 if the result value was greater than zero, otherwise -1. After obtaining the classification for each sample this is converted into coded ASCII characters with 8 bits inlength, converting the -1 label in 0.
The result of this operation returns the phrase:
“***WntrComing***”. We can interpret it as: "**WinterComing**"
Once the suggestion was obtained, we recognized the original sentence using the new suggestion and the one collected in the previous exercise.

2.4.1. Obtained result

So, the final quote turns out to be:
Winter is coming, we know what’s coming with it. We can learn to live with the wildlings, or we can add them to the army of the dead.
From the film:
Game of Thrones

2.5. Further analysis on the chosen parameters

Following we conduct two types of analysis one on the data obtained for the training of the model and one conducted on the data coming at given instants to make predictions.

2.5.1. Analysis on training data prediction:

To evaluate the evolution of model's accuracy, we also involved training samples. In particular what we have done was, as each new sample arrives during the online procedure, evaluate the current model prediction on it and the corresponding loss, before that this sample is used in the fitting procedure. In this way we exploited the most the training set, using it as a sort of test set, to estimate the goodness of the model at each instant.

MIE analysis

The first parameter that we have seen for analyzing the performance of the model with the step size selected is the MIE. It's a very basic information to evaluate at all how our model performed.
We decide to include in this analysis also the MIEs relative to other solutions, about the value of step size or that of principal components taken, in order to make comparisons.
We know that this term is probably the less informative in this case (where change the truth), in fact, a better analysis could be done seeing how change the instant error over time and not only watching a mean value of the error. In fact, observing the trend of the instant error we expect to see some peaks where the underlined truth changes and flatten zeros values as soon as the model settles down. Instead observing the mean value of the error this peaks might be hidden, so we think it is important to see all both this information, that could give a complete view of what's going on.
Below we first report an illustrative table of MIE's obtained in different settings:

MIE	Learning Rate	Principal components
0.1519	0.01	1 (best)
0.0369	0.01	5 (best)
0.0954	0.01	10 (best)
0.1181	0.01	X (best)
0.2949	0.001	1
0.0782	0.001	5
0.4081	0.001	10
0.3776	0.001	X
0.6001	0.0001	1
0.4078	0.0001	5
3.7107	0.0001	10
3.7657	0.0001	X
1.5679	0.00001	1
3.2407	0.00001	5
25.2720	0.00001	10
27.1735	0.00001	X

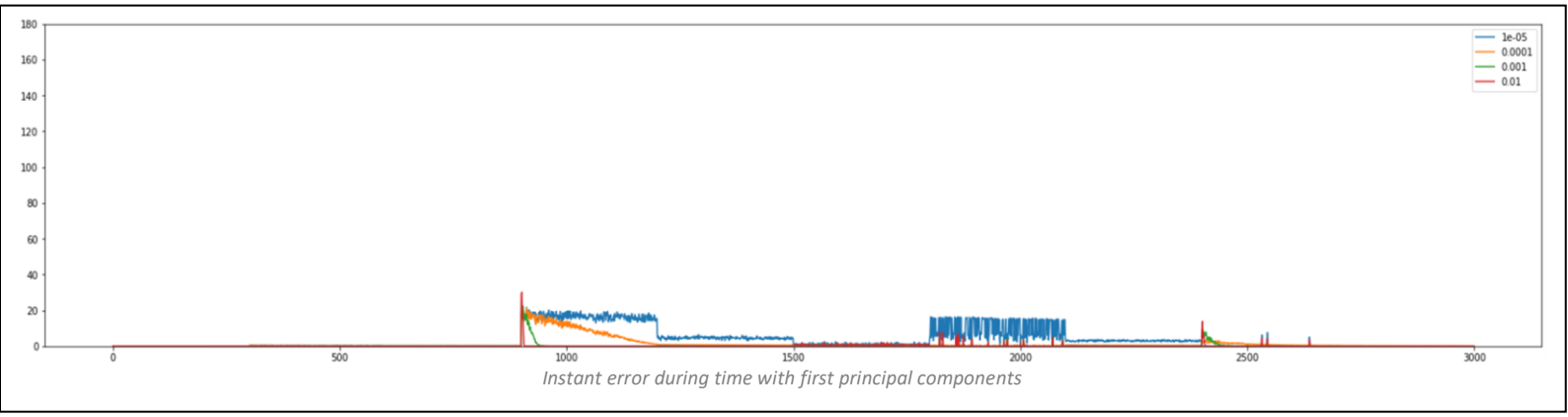
It is possible to see how, when the learning rate become too small, the MIE param becomes worryingly high. Probably when this rate becomes too small, to follow the changes in the truth, never being able to settle the pattern. Is possible to see that in this situation, including more principal components in the model is dangerous for the performance, as the updating rate become not sufficient to correct a large number of parameters, leading to accomplishing bigger errors.

Instant error analysis

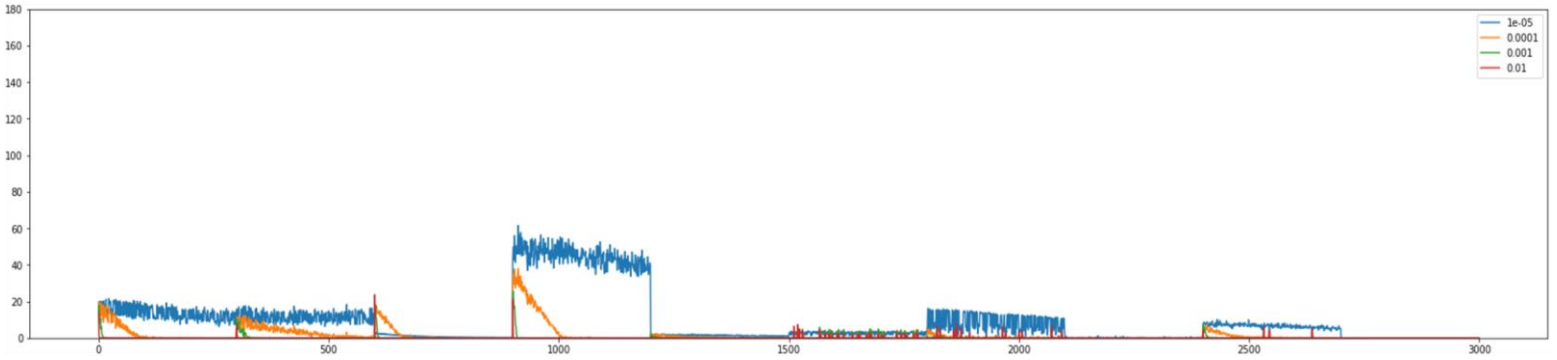
Continuing the considerations of the previous paragraph, we decide to see how the instant error explained above, changes during the training procedure. In particular also in this case we decide to make a relative comparison with other model configurations.
Below we report some comparison graphs, they are explorative as for the step size as for the principal components selected for the comparison.

So for each graph, we fixed the number of principal components selected and we represent, with different colored lines, the different step size analyzed.

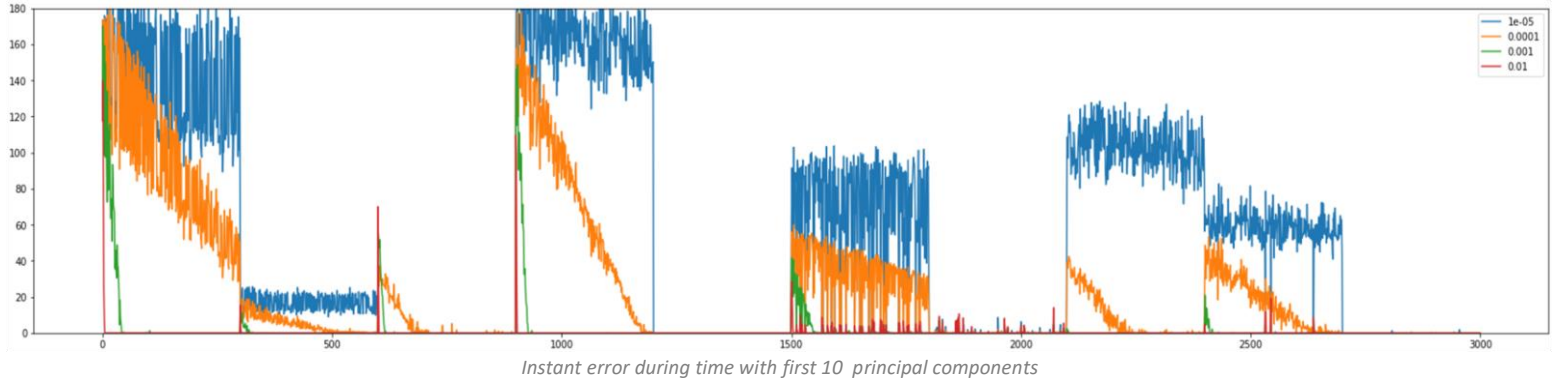
For graphic quality only the first 3000 time instants are initially shown, with y in a range [0,180]:



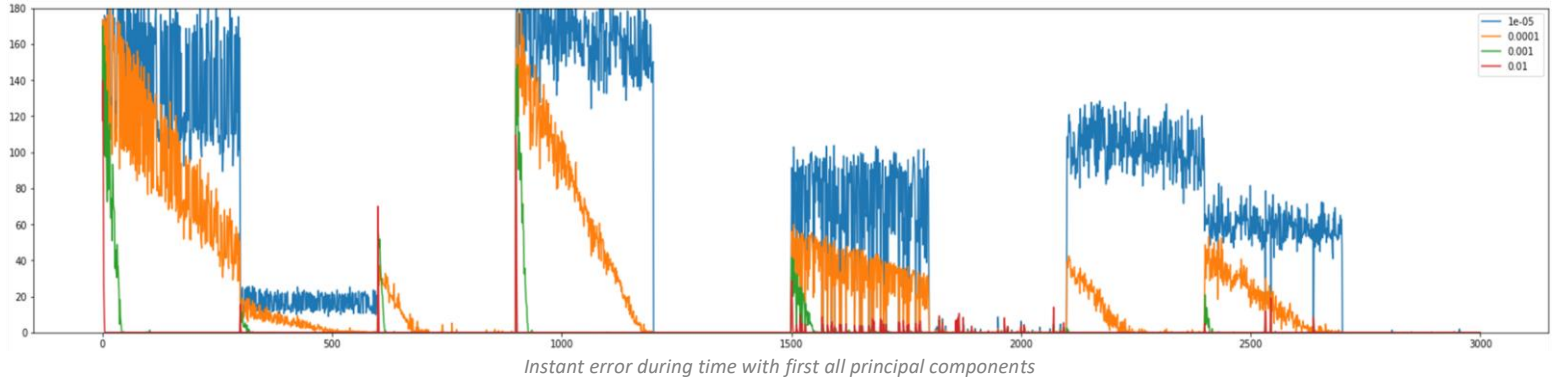
Instant error during time with first principal components



Instant error during time with first 5 principal components

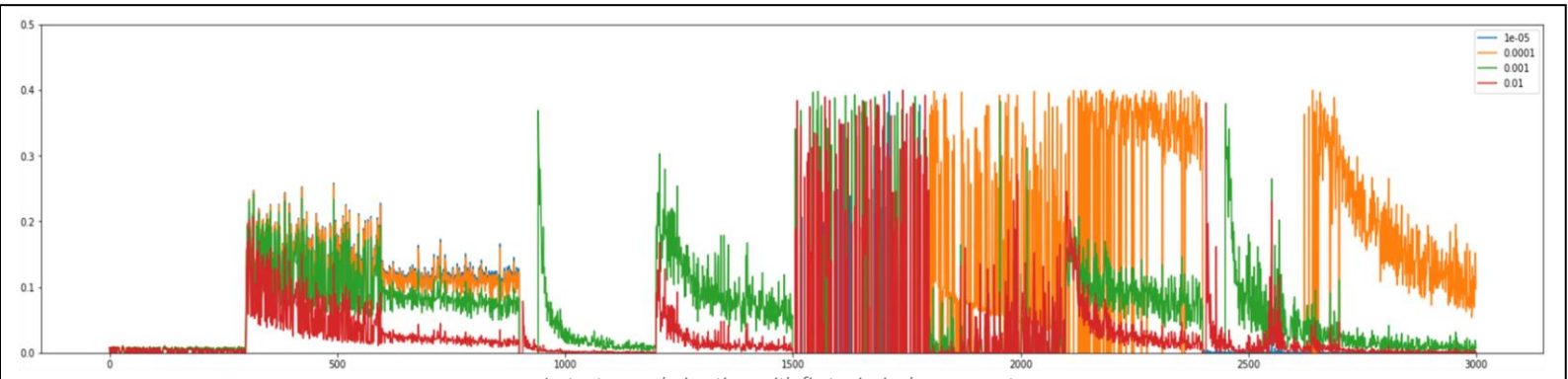


Instant error during time with first 10 principal components

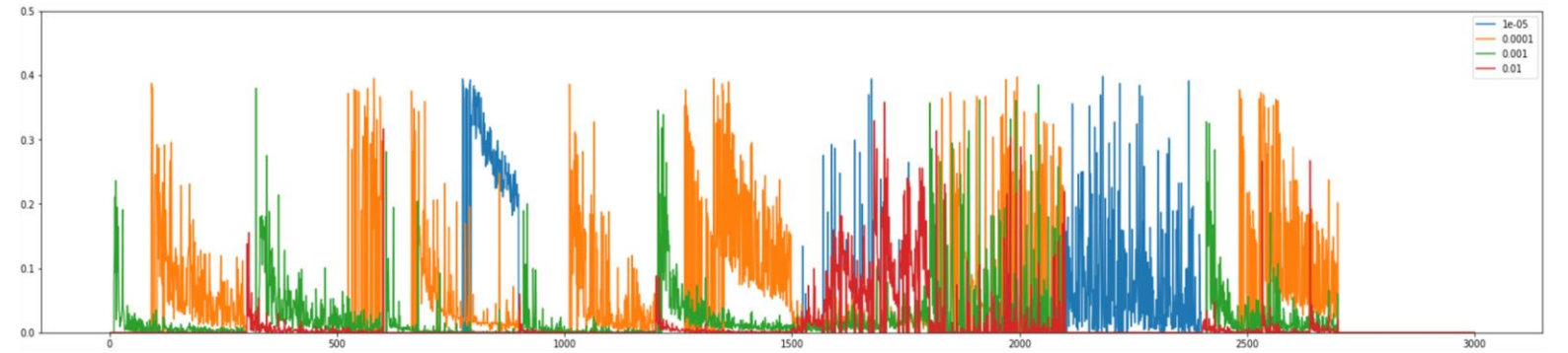


Instant error during time with first all principal components

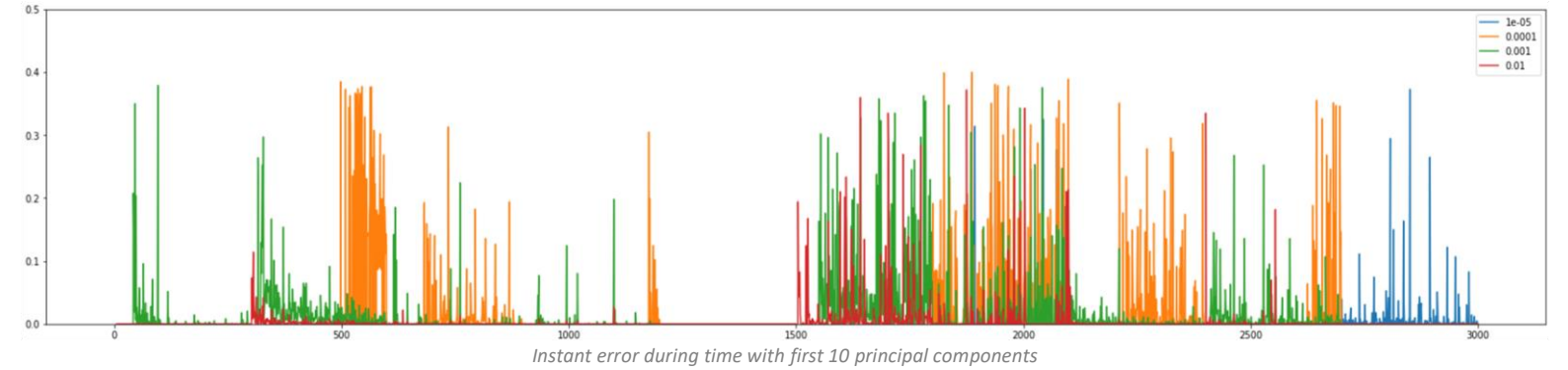
Same configuration's Graphs but in a y range [0,0.5], removing instant errors bigger than 0.5:



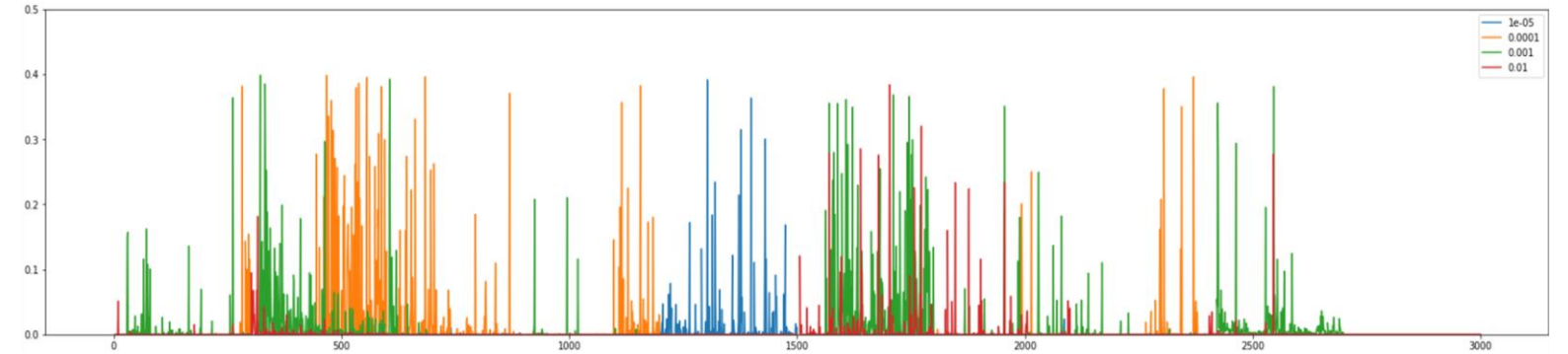
Instant error during time with first principal components



Instant error during time with first 5 principal components

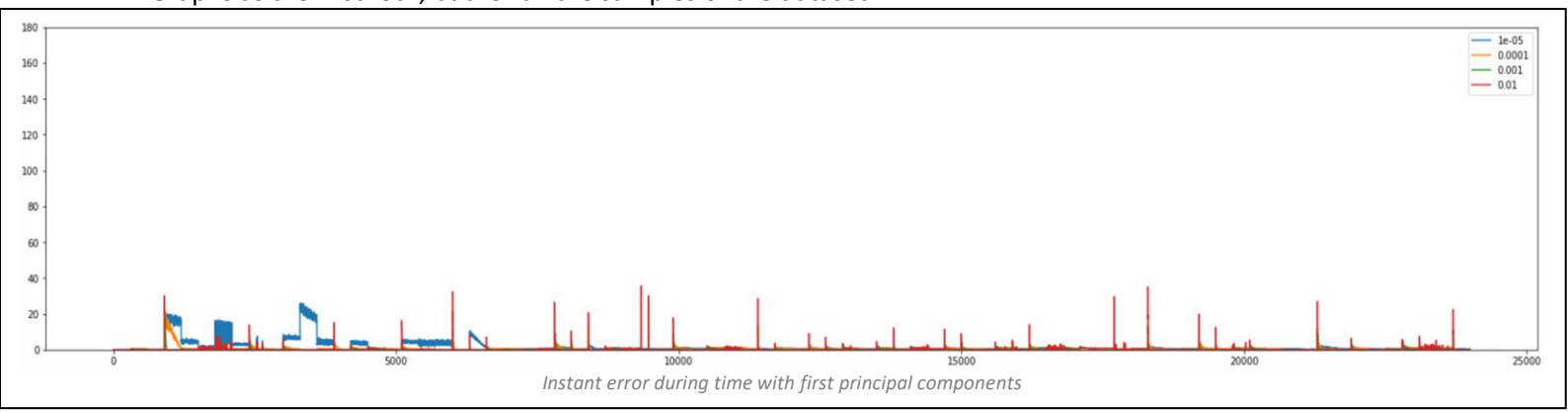


Instant error during time with first 10 principal components

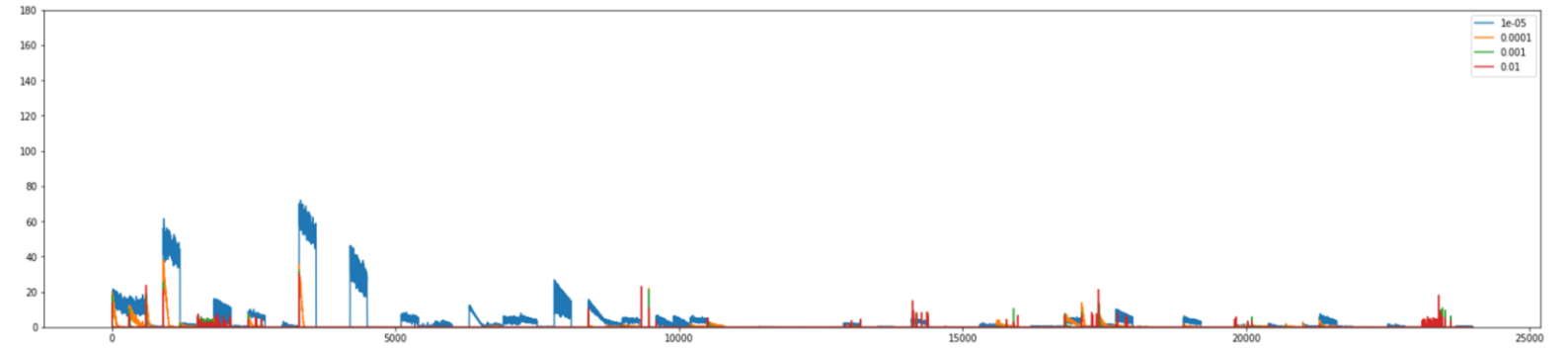


Instant error during time with all principal components

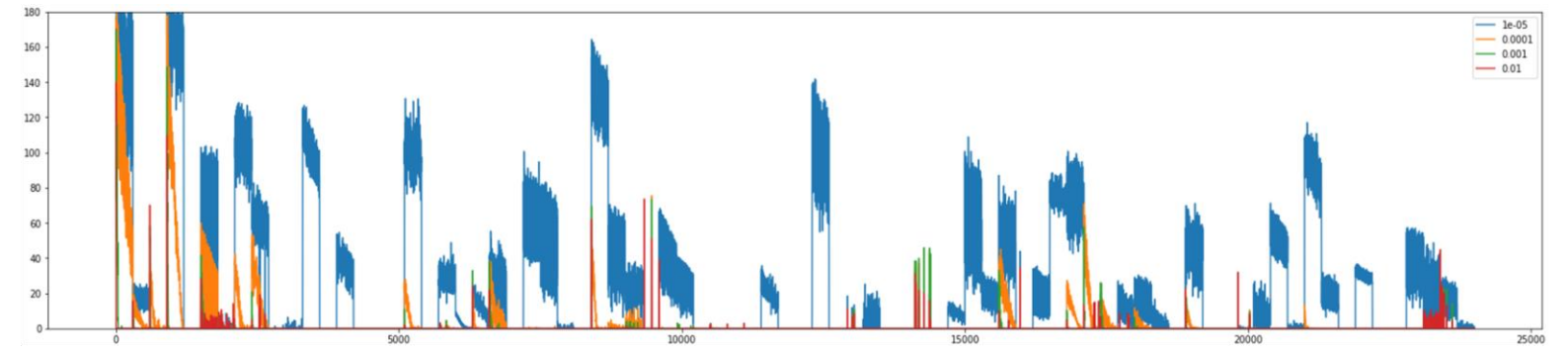
Graphs as the first four, but for all the samples of the dataset:



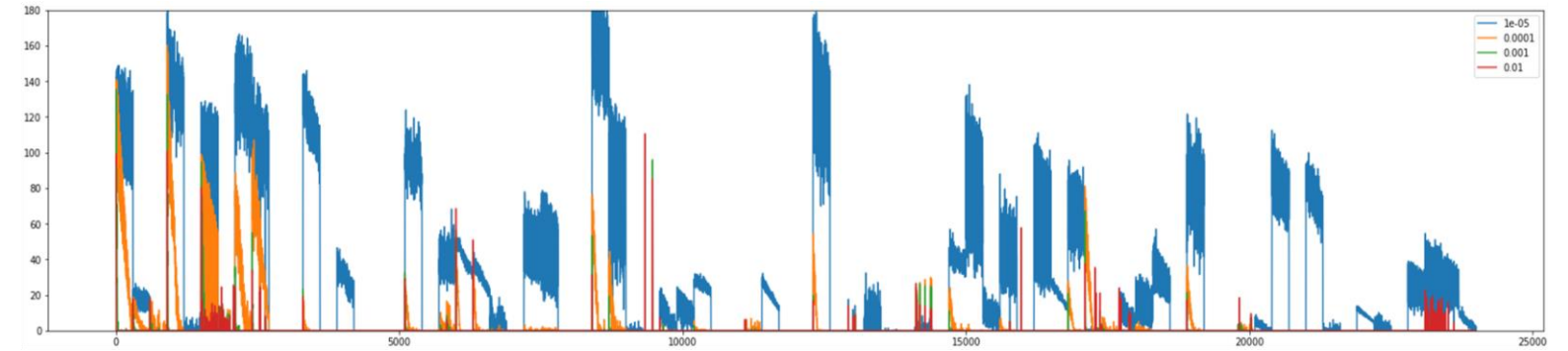
Instant error during time with first principal components



Instant error during time with first 5 principal components



Instant error during time with first 10 principal components



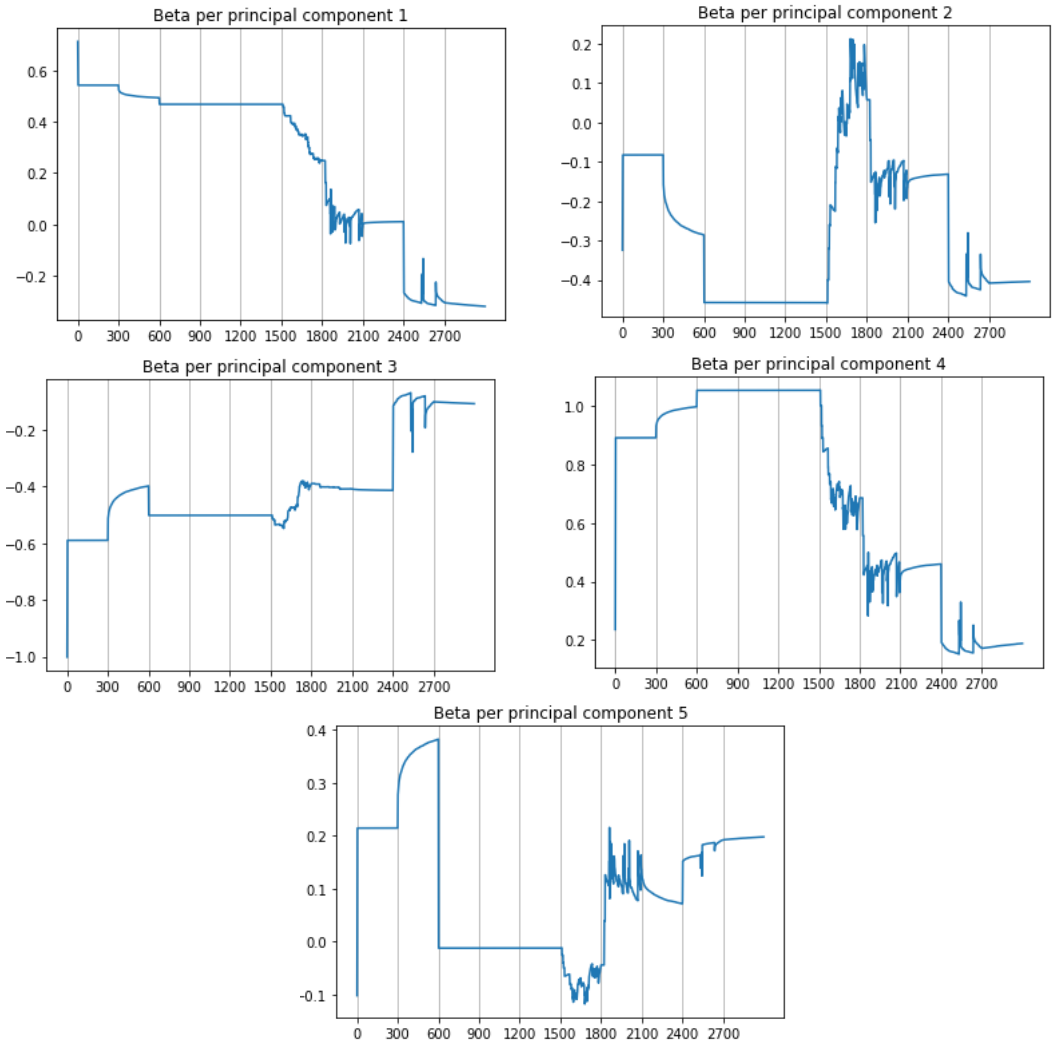
Instant error during time with all principal components

The first evaluation is that what we expected respect the MIE trend in terms of peaks is satisfied. Furthermore from these graphs, we can see, as the step size decreases, the model can no longer quickly change its trim to fit the current truth, maintaining an instant error quite big during time. The best configuration from this analysis is with five principal components included and 0.01 of step size:

- 5 principal components, in order to have not too big peaks like that obtained including more components, and a faster convergence to a 0 instant error with respect to model with fewer included.
- 0.01 as step size because it seems a good choice for the convergence of the model with respect to the other seen. (allow to make good prediction each 300 step)

Beta variation during time

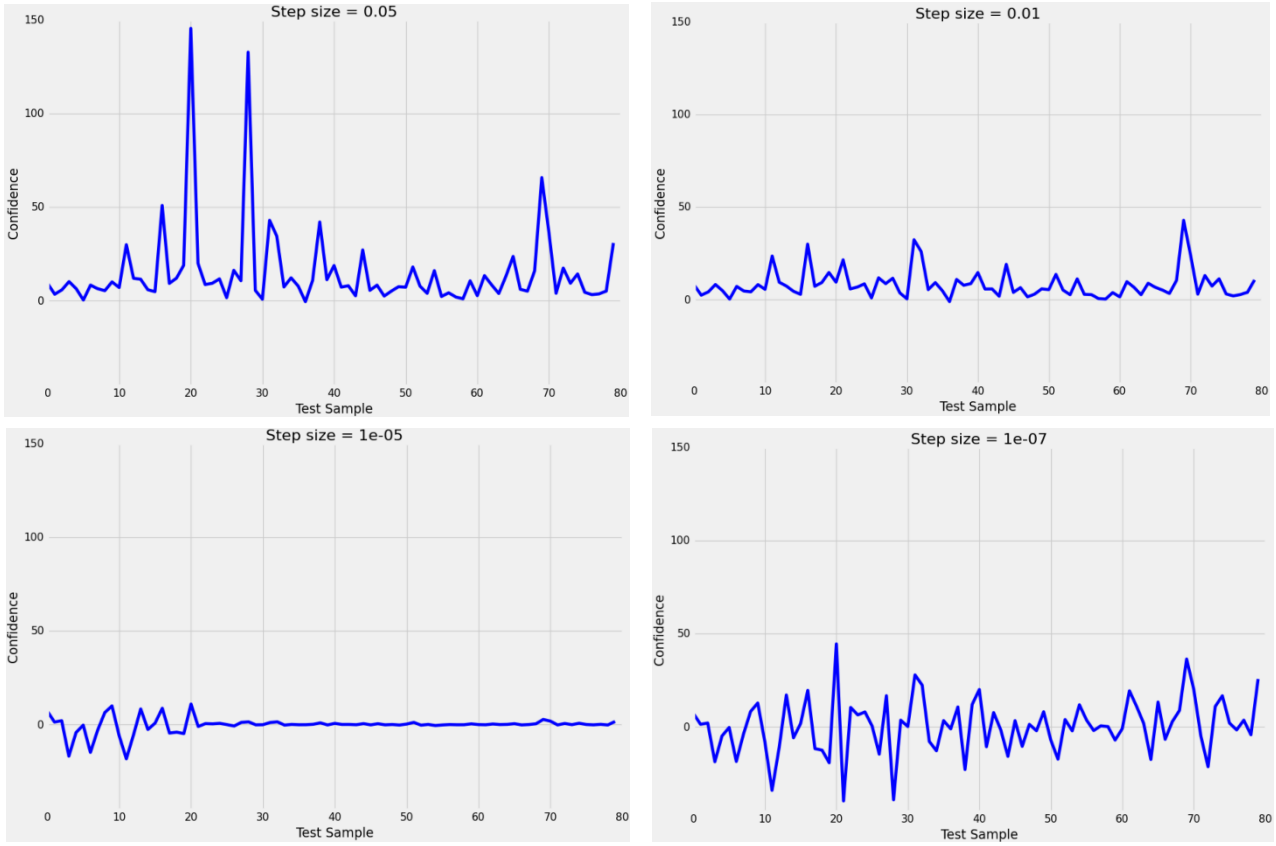
In this section, we will analyze the behavior of the different coefficients during the gradient algorithm. We see as when the underlying truth changes, the coefficients change their values but reach convergence after a few moments in most cases, as it is possible to see in the flatten zone.
The time required, after the variation, to achieve the settling is related to the step size that as we see above bring the instant error close to zero in few steps.
We also observe that there is a different behavior in the range 1500-2100, where the coefficient fails to reach stability and fluctuates significantly, we suppose that in this window the truth change continuously avoiding an assessment of the model.



2.5.2. Analysis on test samples predictions

In this section we focused on the samples of test that comes with a specific cadence, in particular we not only calculate the final quote “WntrComing” from the binary code of the entire string formed by zeros and ones, but we try to conduct further analysis always in the view of evaluate the parameter chosen through some comparison.

Predictions confidence:



With those plots we analyze the predictions done during the stochastic gradient descent algorithm on the test samples, with different step sizes. Starting from the final calculated quote “WntrComing” we obtained the binary code of the entire string formed by zeros and ones. In the stochastic gradient descent algorithm, we calculate the predictions for the test samples with the betas obtained with the updating rule explained in previous section:

$$Bs = Bs - StepSize * grad(eCost\ Function(xnew, ynew))$$

Using those betas, at the time instant specified by the parameters, we calculated a prediction multiplying betas for the test sample related to the specific time instant. This prediction is used to calculate the output of the classification that is 1 if the prediction is greater than zero, otherwise 0. So, taking into account the different predictions done in different time instants, we evaluated if the predicted value is correct or wrong comparing it to the correct one obtained from the binary code of the final quote. In these plots we analyze the level of confidence of the predictions; when a prediction is wrong, it will have a negative value in the plot, corresponding to its distance from 0 and, so, to the positive values (the correct prediction). Otherwise, when the prediction is correct, the value in plot is positive; the more positive it is, the more the prediction is confident for the specific test sample. We analyzed the predictions done with different step sizes: 0.05, 0.01, 0.00001, 0.0000001. For the last two step sizes we can notice that a lot of errors during the predictions are done, so the system is far from the correct estimations for the betas, more with the last one step size the commits greater errors than the penultimate. The first two step sizes obtain good results in term of predictions, so these two are a good choice for our system. As we have already seen during the analysis of the convergence of the SGD algorithm in previous sections, different step sizes have different time of convergence. Choosing 0.01 allows the system to be robust to the variation of the ground-truth with a good time assessment.

2.6. Comparison with a simple naïve bayes approach

The intent of this paragraph is to compare the approach implemented with the Gradient Descent algorithm, with a simpler solution that is the naïve Bayes approach. This approach takes this name because uses the Bayes rule to make prediction but in a naïve way, making many simplification assumptions. In particular about the generative model, it considers the different features as independent, given the hypothesis. So, the total generative model is:

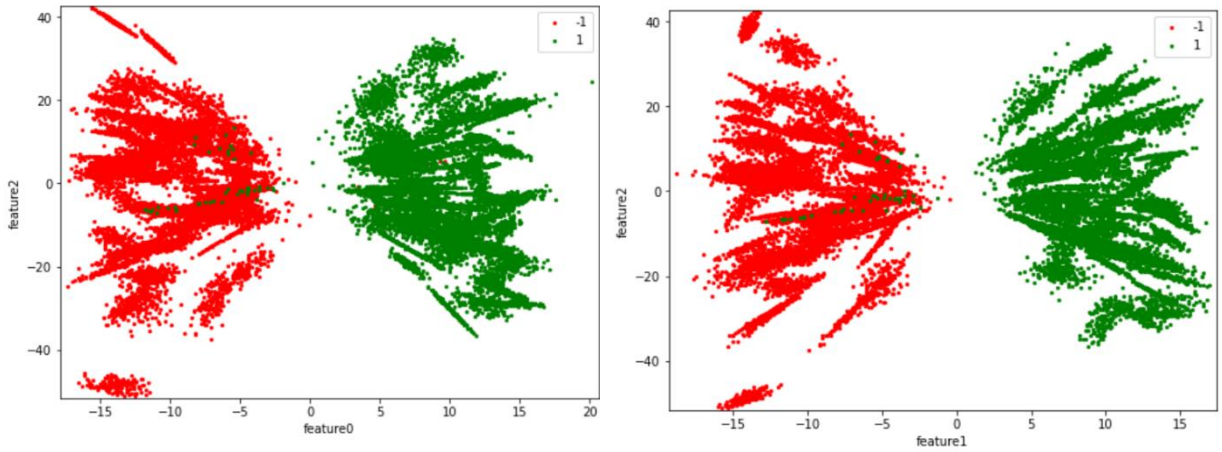
$$p(x|y) = \prod_{k=1}^{d} p_k(x_k|y)$$

in which d is the number of dimensions of the features, and the index k is used to underline the fact that the distributions could be different among different k-features.
In Naïve Bayes approach the generative model of the single feature is supposed to be a Gaussian distribution (for continue variables) or a Multinomial (for discrete one).
So, the approximations made by this method are:

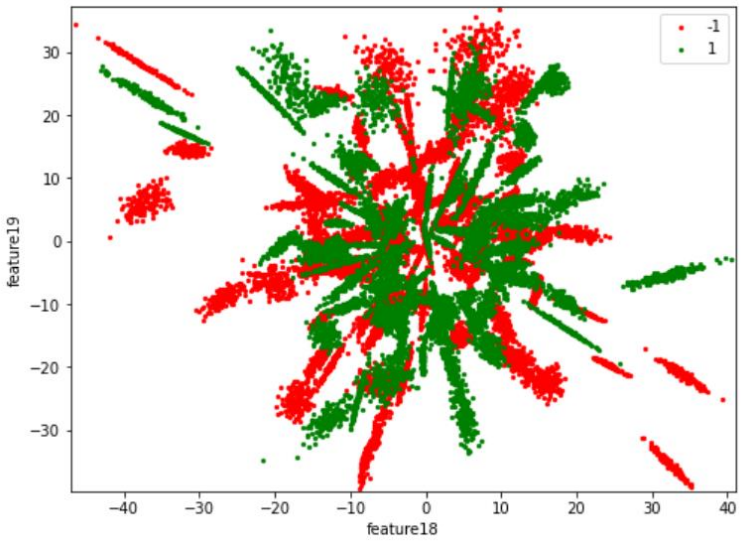
- the absence of dependence between the variables, obviously there are many situations in which not considering dependency relationships can create problems;
- the assumption on the gaussian shape of the $p_k(x_k|y)$, in particular the mean and the variance to particularize are estimated from the dataset, so it is necessary to have a representative one.

Regarding the estimation of the prior probability $p(y)$, it can be considered to be in an equally probable case, or it can be evaluated seeing the occurrences of the outputs into the training set.

Starting from the idea of seeing how a model like this can perform, we first analyze the features and their separability in order to evaluate which and how many could be useful to involve in the naïve bayes prediction. To analyze feature's separability, we decide to plot them in couples and analyze how the data of different classes are distributed in a space only described by the two analyzed features. Following are reported only the more significant graphs obtained during this analysis. In particular in the first two graphs it is possible to see how the data are distributed in a 2-dimensional space in which one of the dimension coincides with the features f0, or with the features f1. These graphs are extremely similar regardless of which feature (different from f0 and f1) coincides with the other dimension. In this distribution it is clear to see that data can be quite accurately separated from each other, except for some points which however have, for the analyzed features, values very different from data of the same class.

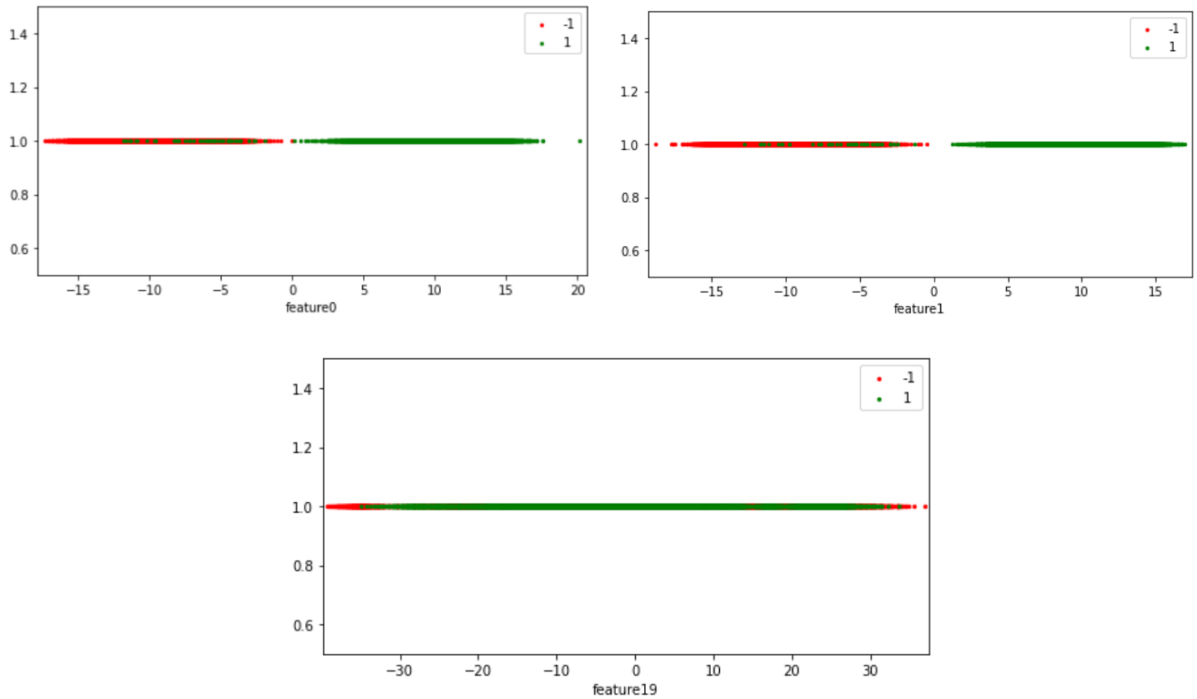


This level of separability is not even remotely achieved with features different from f0 or f1, as it is possible to see in the graph below. In particular we only report a single graph, that is however representative of a situation common to all features from f2 to f19. In this situation it is clear that features values are so similar that distinguish the two classes became impossible.

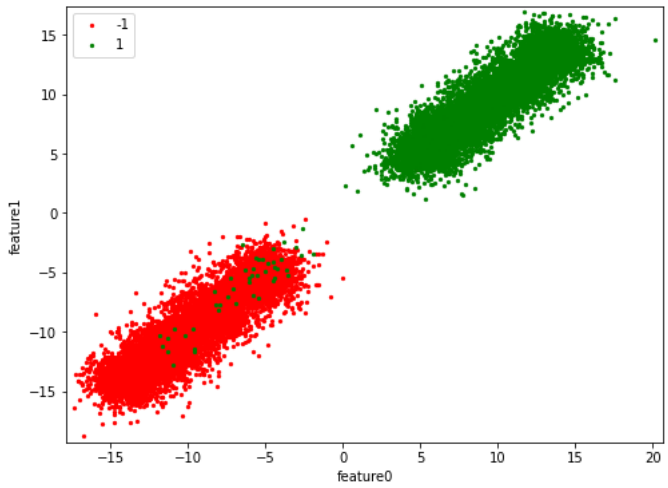


Starting from these considerations and from the observations of the graphs about features f0 and f1, it is clear that probably the greater capacity for separability lies precisely in the features f0 and f1. So, an experiment that was done, is that of trying to separate data using a 1-dimensional representation using only one of the features f0 or f1.

As it is possible to see from the first two graphs also these two single features can separate, in a quite accurate manner, the two classes, and it does not happen for the other features (as it is possible to see from the third graph, representative of what happens for all the features from f2 to f19).



So, a last interesting analysis to be done on the features is the inspection of the relationship between f0 and f1. As it is possible to see in the graph reported below, the combination of these two features enables the separation of the two classes, but also there is a correlation between these two features because the samples are distributed along the bisector.



So, to evaluate as a Naïve Bayes approach could work in a situation like this, we first use this analysis on the features to select a subset of these to be involved in the constructed model. We respected the online process of training data acquisition and of predictions, and so at each prediction step, we implement the process above explained using to estimate mean and variance, necessary to compute the gaussian distribution, the values obtained until that moment.
In detail at each prediction step, for the selected features we evaluate the mean and the variance of the values of that features whose label is respectively -1 or 1, and we used these values to compute the gaussian generative model conditioned to the values -1 or 1 respectively ($p_k(x_k|1)$ and $p_k(x_k|-1)$ for each k feature involved in the model) . Mean and variance are computed on the values obtained until the prediction moment.
Accordingly to Naïve Bayes approach the total generative model was computed multiplying that of each feature ($p(x|1) = \prod_{k=1}^{d} p_k(x_k|1)$ and $p(x|-1) = \prod_{k=1}^{d} p_k(x_k|-1)$ for each k feature).
The prior probability for -1 and 1 ($p(1)$ and $p(-1)$) was evaluated through the dataset distribution until prediction moment.
Then the posterior probabilities for 1 and -1 were computed:
$p(1|x) = p(x|1) * p(1)$ and
$p(-1|x) = p(-1) * p(x|-1)$)
and whichever was the higher one, the prediction was 1 or -1.

Different experiments have been done about which features to involve in the model. Firstly, because we evaluate that f0 and f1 are sufficient, we try a simple model with only one of these, and in both cases the predictions are correct. All other combinations of features that involved one of these two, or both works equivalently in terms of prediction but impose a useless computational burden. All the combinations of features that do not involve or f0 or f1 work very badly.
So, it is clear that in situations like that, it is important to choose the simplest model, and so **one with f0 or one with f1**.