Unsupervised analysis and prediction on American stocks

Bersani Michele [*], *Bosio Carlo* [†], *Lusiani Federico* [‡]

Topics in Statistical Learning, Academic Year: 2020/2021

Date: 1 April 2021

Abstract

This project aims at predicting yearly price variations of American stocks, given cross-sectional econometric measurements. Since different techniques (logistic regression, linear regression, k-NN, neural network) did not perform well on this dataset, a simple dataset is parallely analized with the same pipeline to ensure the correctness of the applied method and prove that in the stocks dataset accurate prediction requires further information.

1 US stocks dataset description

This dataset consists in the financial indicators of more than 4000 stocks, measured every year from 2014 to 2018 (figure 1). Since no time dependecy is accounted, the five years are merged into a single dataset, and different occurrences of the same stock across different years are simply considered as different samples.

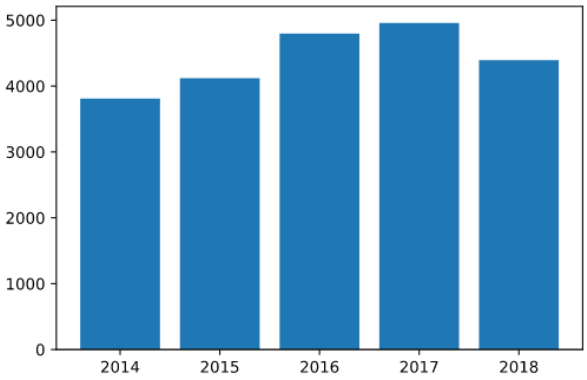


Figure 1: N° of stocks per year

The dataset[1] includes 227 numerical variables and a monovariate output, *Price Var*,

[*]Sant'Anna School of Advanced Studies, m.bersani@santannapisa.it
[†]Sant'Anna School of Advanced Studies, c.bosio@santannapisa.it
[‡]Sant'Anna School of Advanced Studies, f.lusiani@santannapisa.it

[1]`https://www.kaggle.com/cnic92/200-financial-indicators-of-us-stocks-20142018`

representing the percent price variation of each stock for the year. The 227 variables are all continuous, but of heterogenous nature: some are ratios, such as *Payout Ratio* and *Quick Ratio;* some others are quantities, like *Revenues* and *Investments*.

The presence of *nulls* is significant and very sparse throughout the entire dataset.

2 Preprocessing

The dimension of the dataset, the presence of *nulls* and the diversity between different variables requires different stages of preprocessing. The pipeline we developed comprises the following steps:

- Rows dropping if $Price\ Var > 150\%$
- Rows dropping when when nulls exceed 20% of the variables
- Columns Normalization
- Logarithmic Transformation of Skewed Columns
- Nulls imputation with custom MICE
- Columns Normalization

2.1 Data dropping

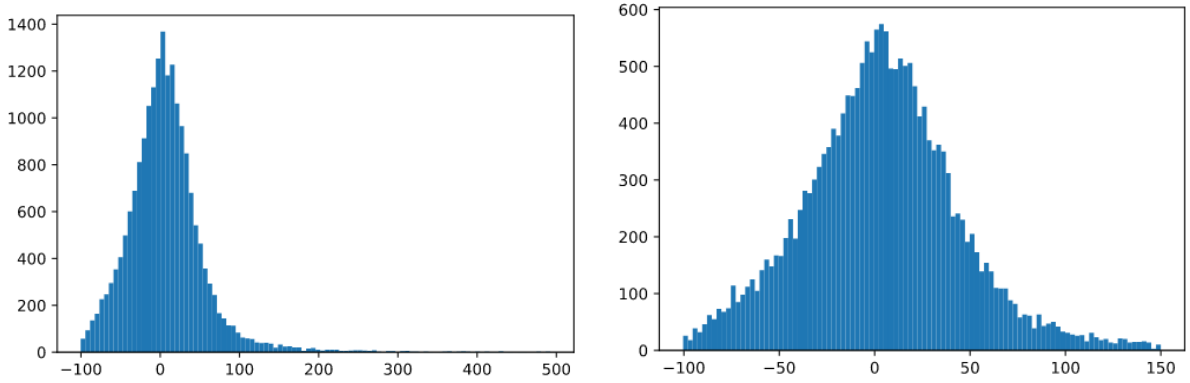
The first preprocessing step consists of removing from the dataset the data (samples or variables) that clearly represent anomalous values or behaviors. In this project, two criteria for data removal have been adopted:

- Dropping the sample if $Price\ Var > 150\%$;
- Dropping the variable (or pattern) when the number of its NULL occurrences exceed a certain threshold.

The first criteria seems reasonable since the patterns represent real US stocks. It is possible for a stock to increase its value more than 150%, but quiet exceptional. In order to not deliberately push the limits of the utilized statistical learning methods, only the "ordinary" samples have been kept. In figure 2 the *Price Var* occurrences are showed.

The second criteria aims to drop the most "unknown" variables(or samples), on which a significant part of the occurrences should be determined by imputation. To analyze the amount of NULL occurrences over the dataset, the columns in a first place and the rows successively, have been sorted by increasing number of NULL occurrences. In figure 3 the results of this operation are represented.

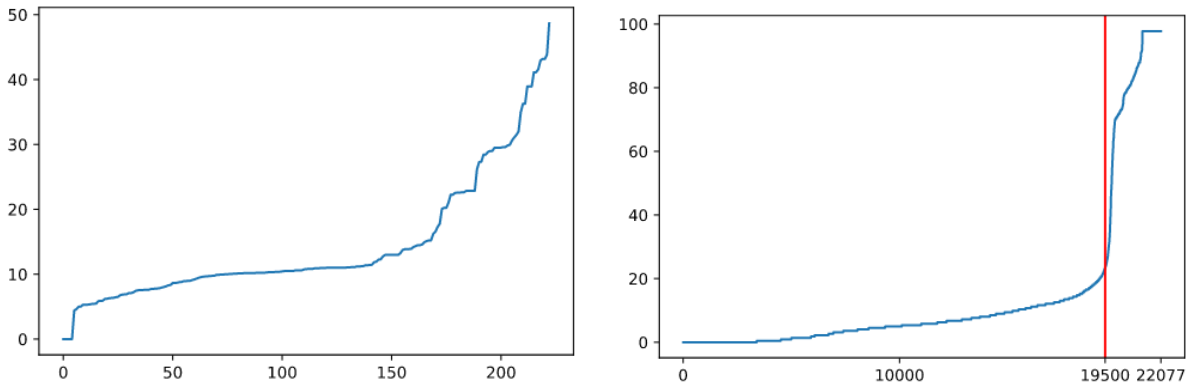
As it is possible to observe, an elbow is present in the rows NULL distribution (and a saturation of 100% is reached by some samples). All the patterns beyond the elbow have been dropped. A similar trend has not been observed in the variables NULL distribution, so no column has been dropped.



(a) Up to 500%

(b) Up to 150%

Figure 2: *Price Var* distributions



(a) % of NULLs per column

(b) % NULLs per row

Figure 3: NULLs distributions

2.2 Logarithmic Transformation

The logarithmic transformation is a common operation for handling skewed variables distributions. The term "skewed" means that the variable is characterized by an asymmetric distribution, with the tail on a side. The concept is represented in figure 4. The "skewness" is therefore a measure of the distribution asymmetry.

Several parameters and indexes exist for evaluating skewness. For the sake of simplicity we adopted the following measure:

$$s(X) = \frac{\mu(X) - \nu(X)}{\sigma(X)} \tag{1}$$

Where s, μ, ν, σ are the skewness measure, the mean value, the median value, the standard deviation and X is the reference variable. The logarithmic transformation allows to make a distribution more symmetric (and reduce the skewness). In fact, the $\log : \mathbb{R}^+ \to \mathbb{R}$ operator, applied to a variable x, tends to compress and approach to zero

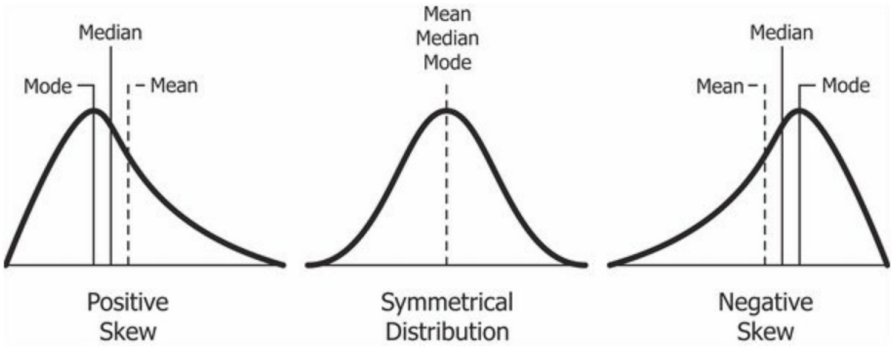


Figure 4: Mean and median values in asymmetric distributions

the $x > 1$ values and expand the $0 > x > 1$ values.

In order to handle the skewed variables, the s parameter defined in equation 1 is computed for all the columns. A positive and a negative threshold are defined to spot the positive and negative skewed columns. In particular a distribution is labeled as skewed if $s < -0.05$ or $s > 0.05$. The logarithmic transformation is able to adjust the positive skewed distributions, so the negative skewed ones are multiplied by -1. At this point all the skewed columns are ready to be transformed with the formula:

$$X \leftarrow \log(X - \min(X) + 0.01) \tag{2}$$

Each column is translated by its minimum value and a small positive constant is added in order to not try to compute the logarithm of a negative quantity.

2.3 Nulls imputation with custom MICE

The standard strategy of imputing nulls by the mean of the corresponding variable is considered, but discarded because in many cases this leads to evidently poor reconstructions of the original data: for instance, imputing Apple's revenues as the mean of other companies is not acceptable. Multiple Imputation by Chained Equations (Algorithm [1]) is instead a more complex method, which exploits iterative linear regression to obtain accurate imputation.

Algorithm 1 Multiple Imputation by Chained Equations

Replace all the missing values with the mean of the corresponding column
while Imputed values have not stabilized yet **do**
  for *col* in $\{Columns\}$ **do**
    $\mathbf{D_k} \leftarrow$ patterns where *col* was originally imputed
    $\mathbf{D_k}.col \leftarrow null$
    $\mathbf{D_j} \leftarrow \{Patterns\} \setminus \mathbf{D_k}$
    Train a linear regressor on $\mathbf{D_j}$, with $\{Columns\} \setminus col$ as independent variables and *col* as objective
    Regress *col* for $\mathbf{D_k}$
  end for
end while

The convergence criteria that marks the end of MICE is obviously harder to reach as dataset size increases. In this case, with more than 19000 rows and 200 columns, it is not even guaranteed for the algorithm to converge. This is why we implemented a custom version of MICE, that puts together groups of 10 columns and works on each group separately.

It is reasonable that better imputation can be obtained by splitting columns into balanced groups. This means that each group should compensate variables with high noise and many missing values, with others presenting low noise and few missing values. Since we have no a-priori information regarding the amount of noise per variable, the only splitting criteria we account is the number of nulls. Our custom implementation of MICE is shown in Algorithm 2.

Algorithm 2 Custom MICE by grouping columns

completeData $\leftarrow$ [ ]
Init D as empty deque
for *col* in $\{Columns\}$ **do**
  D.push(*col*)
end for
Sort D by number of nulls per column
while D is not empty **do**
  newGroup $\leftarrow$ [ ]
  for j in $\{0, \ldots, 4\}$ **do**
    newGroup.append(D.popFront())
    newGroup.append(D.popBack())
  end for
  completeData.append(MICE(newGroup))
end while

3 Data exploration

After the preprocessing stage, we apply unsupervised methods to explore the information present in the dataset, and apply further refinements to the data.

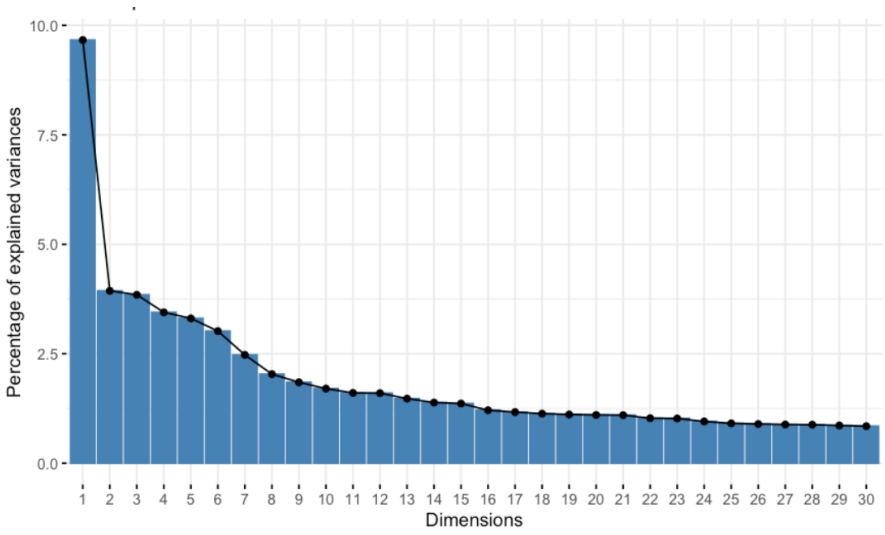


Figure 5: Percentage of variance explained by the first 30 PCA dimensions.

PCA. The results of Principal Component Analysis on the dataset are shown in Figure 5, 6 and 7. As we can see from Figure 5, about 9% of the dataset variance can be expressed by the first PCA dimension. Figure 7 shows the projection of the original features on the first two PCA dimensions. It is interesting to note that most features are closely aligned to either the dimension 2 or dimension 1 axis (especially the latter), meaning that these two dimensions can be well described in function of two distinct sub-sets of the original features.

Figure 6 shows that by projecting the whole dataset on the first 65 PCA dimensions, we can retain about 80% of the original variance. Although it could be possible that by doing so we are loosing valuable information, it appears to be a reasonable trade-off in order to reduce the high dimensionality of the original feature-space. For this reason, we decided to perform K-means clustering in this reduced space, hoping to obtain better clustering.

K-Means clustering. After projecting the data on the first 65 PCA dimensions, we are interested in finding out whether the data can be divided into clusters, by applying K-means with euclidean distance. Since applying the K-means algorithm on the whole dataset (19000 patterns) would have been computationally heavy, we have decided to use a sub-set of 100 patterns, randomly sampled from the dataset. The results are shown in Figure 8.

By looking at the plots in Figure 8, we can see that the algorithm was not able to find proper clusters on the sample, even after dropping some problematic outliers from the

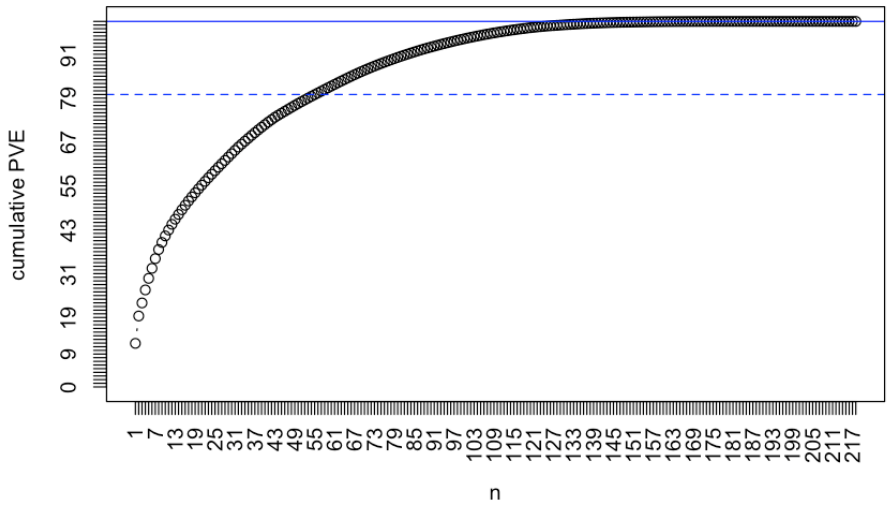


Figure 6: Cumulative PVE in function of the number of PCA dimensions retained

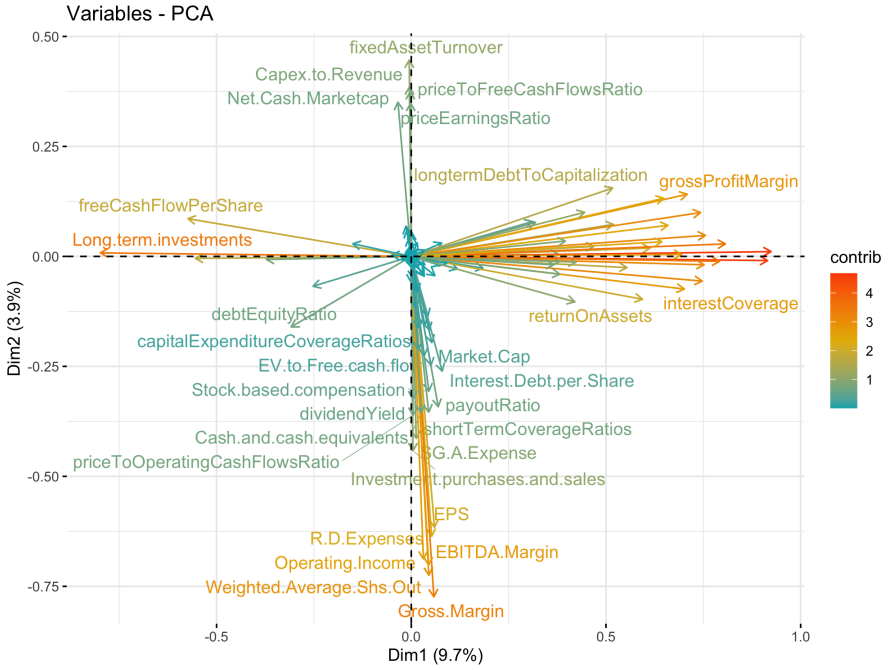
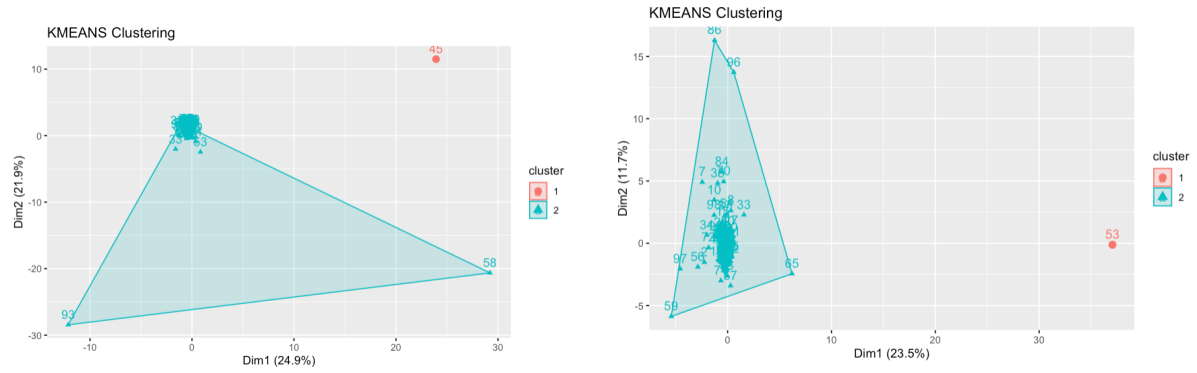


Figure 7: Projection of the original features on the first 2 PCA dimensions

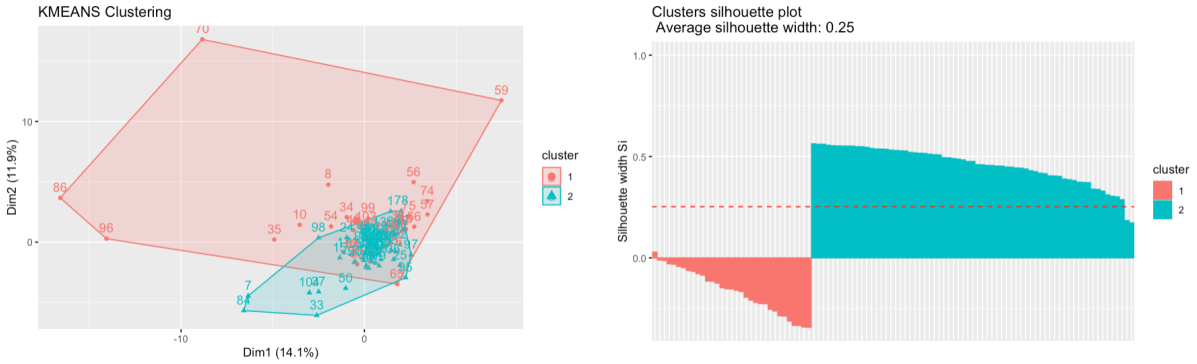
sampled distribution: the two clusters overlap on most patterns, and cannot therefore prove to be meaningful. This is evidenced by the silhouettes values of the patterns, which appear to be negative for the quasi totality of the patterns belonging to the first cluster.

Further attempts at clustering, using different values of K, have been made. The



(a) First result

(b) After dropping 3 outliers



(a) After dropping 4 outliers

(b) Silhouettes

Figure 8: K-means results over a sampled distribution of patterns.

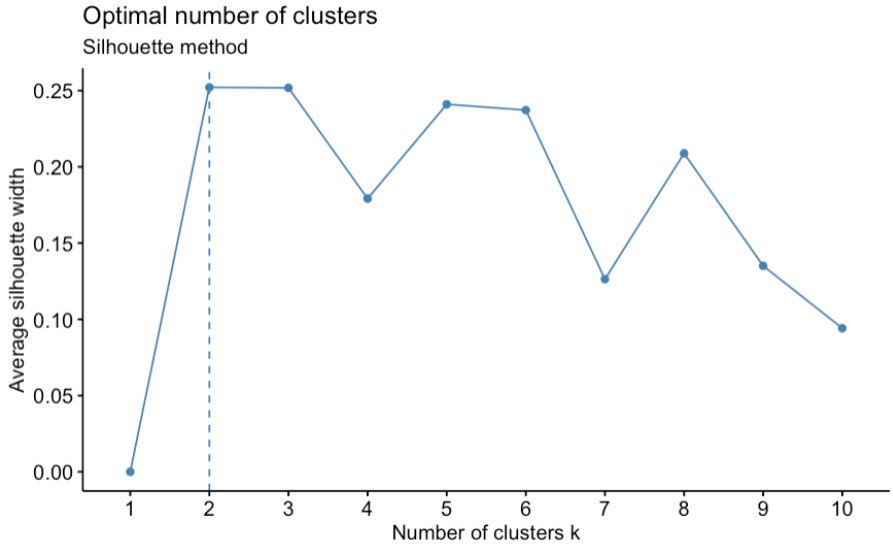
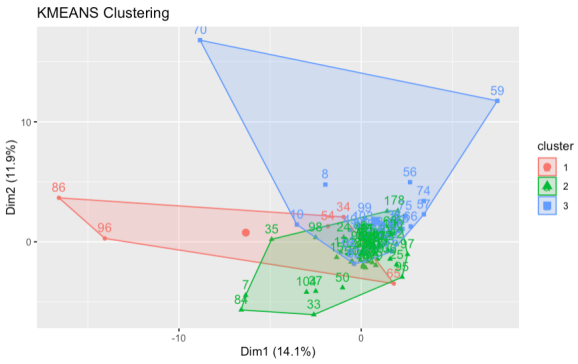


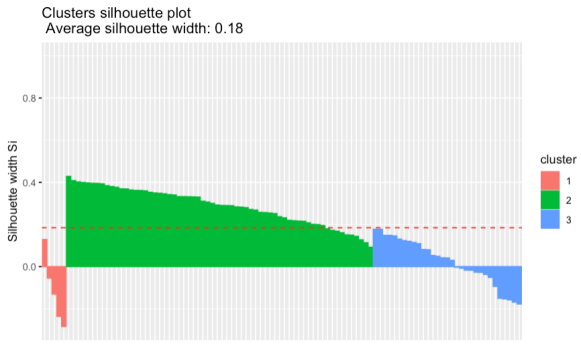
Figure 10: Average silhouettes values in function of the number of clusters.

plot in Figure 10 shows the average silhouettes value over the whole sampled distribution, which can be interpreted as a measure of quality of the resulting clustering. As the

plot shows, grouping the data in 3 or 5 clusters seems to represent another promising option. The results are shown in Figure 11. As we can see, these values of K did not bring a noticeable improvement in the clustering of the patterns.

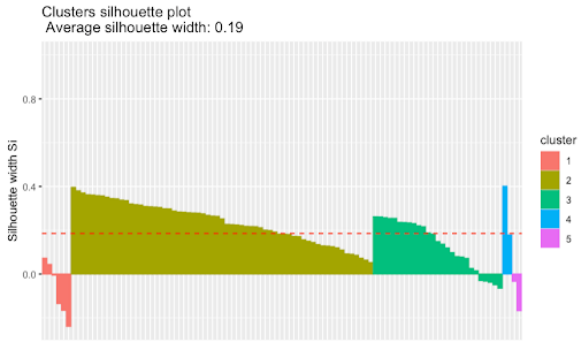


(a) Clustering with $k = 3$



(b) Silhouettes with $k = 3$

(a) Clustering with $k = 5$



(b) Silhouettes with $k = 5$

Figure 11: Further attempts with K-Means clustering

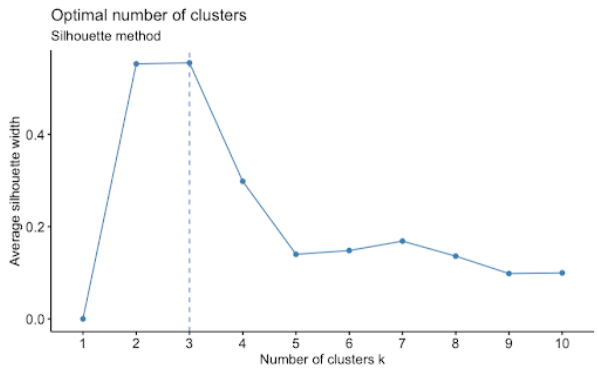
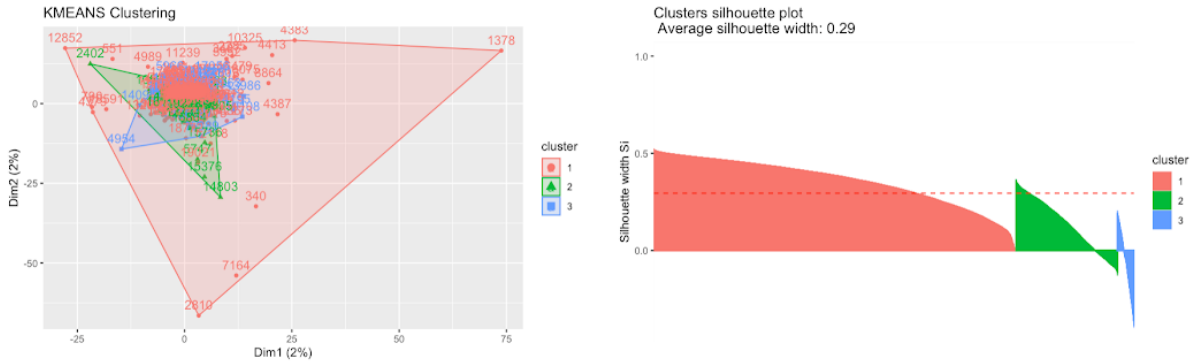


Figure 13: Average silhouettes values in function of the number of the number of clusters (log-transformed data).

The third K-Means clustering analysis has been carried out on the log-transformed

dataset. Again, we have computed the average silhouettes value in function of different values of K, in order to identify an optimal number of clusters. The plot in Figure 13 shows that $K = 3$ appears to be optimal.



(a) Clustering with $k = 3$

(b) Silhouettes with $k = 3$

Figure 14: K-Means clustering of log-transformed data

Looking at the results plotted in Figure 14, we can see that (as in the previous case), the data does not appear to form proper clusters. Most of the samples are grouped in one main cluster and a significant part of the remaining patterns have negative silhouette values.

4 Learning and Predicting

The core section of this work regards finding a learning model which can be trained by observing part of the dataset and then predict *Price Var* for incoming patterns by only looking at the input variables. We selected the following statistical models:

- A logistic regressor, to predict a boolean output (0 or 1 if *Price Var* is respectively negative or positive)
- A linear regressor, to predict the exact value of *Price Var*.

Both of the models were evaluated through 5-fold Cross-Validation, by dividing the dataset into a different partition Training Set (80%) and Validation Set (20%) at every fold. Cross-Validation plays a crucial role in model selection, not only refining the estimate of the error by averaging over 5 Validation Sets, but also providing a measure of the variability of the model, through the variance over the folds.

All the models were implemented and trained through Scikit-Learn off-the-shelf library.

4.1 Logistic Regression

Logistic Regression obtained a mean validation error rate of **39.2%**. This performance is quite poor, in fact - being the output very well balanced - a random guesser would averagely achieve ∼50%.

Nevertheless, a boolean output is not that significant for finantial applications; in fact, unless the error rate falls to 0, no trader would invest his money without an estimation of *how* the stock price is varying .

4.1.1 Features Extraction

Restricting the input domain through Principal Components Analysis did not improve the performance, obtaining an almost identical error rate. At least, as predictable, it sped up the training phase.

4.2 Linear Regression

Linear Regression is carried out with L2 regularization to avoid overfitting the training data: given the weights $\mathbf{w} = \{w_1, \ldots, w_n\}$, a penalty term $\alpha\|\mathbf{w}\|_2^2$ is added to the cost fuction. This is also known as Ridge Regression.

It is evident that α becomes a hyper-parameter of the model and we leveraged the 5-fold Cross-Validation to tune it.

The results were even worse than with Logistic Regression: the best Mean Validation Error achieved was **30.5** (considering an error of 1% as the unit of measurement). This is very meaningful if compared to the mean absolute value of *Price Var*, that is 31.4: a regressor with all null coefficients (or even a trivial model predicting always 0) would averagely achieve an error of 31.4.

4.3 Neural Network

Observing the failures of the previous models, we wondered whether a more complex model would have performed better and hence tried to perform regression by a fully connected neural network. However, even by increasing the number of nodes in the hidden layers, the Neural Network was not able to perform significantly better than Linear and Logistic regressors.

4.4 Conclusions

These poor performances suggest that the input variables do not contain enough information to predict the output, at least with standard data pre-processing and without

performing hand-crafted transformations of the input variable, looking at the regression performance and iterating transformations to improve it.

5 Pima Indians Diabetes Dataset Description

This dataset has been analyzed with the aim of verifying the correctness of the methods and procedures applied. The objective of this analysis is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements.

The data are obtained from 768 patients, all female of at least 21 years old. The dataset[2] includes 8 numerical variables (e.g. *Glucose Level*, *Blood Pressure...*) and a boolean output, representing the outcome of the diabetes diagnostics. No NULL values are present.

As mentioned above, the same pipeline developed for the Stock dataset analysis has been implemented.

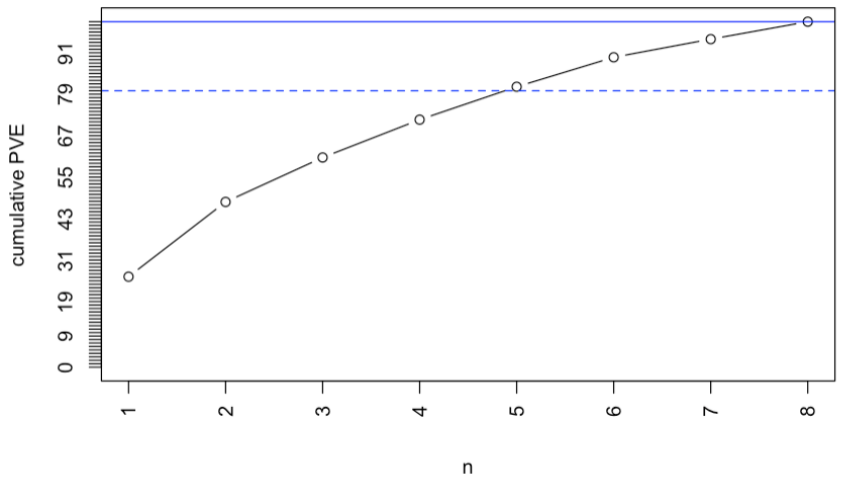


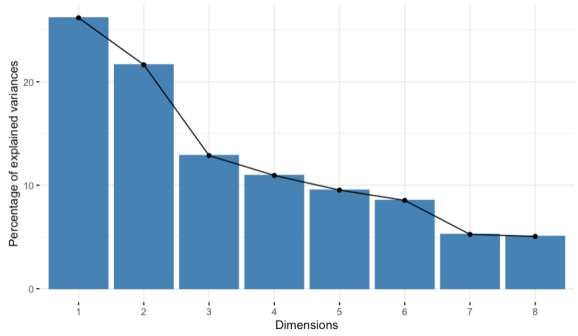
Figure 15: Cumulative PVE in function of the number of PCA dimensions retained

6 Data exploration

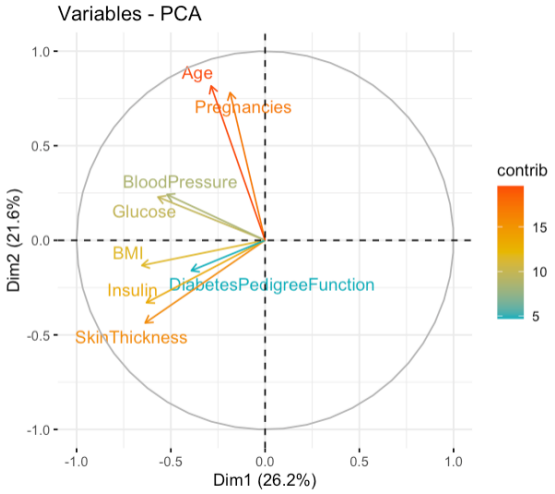
As in the previous analysis, we apply unsupervised methods to explore the information present in the dataset.

PCA. The results of the Principal Component Analysis are shown in figure 16a, 15 and 16b. As we can see from figure 16a, about 26% of the dataset variance can be expressed by the first PCA dimension. Figure 16b shows the projection of the original features on the first two PCA dimensions.

[2] `https://www.kaggle.com/uciml/pima-indians-diabetes-database`



(a) Percentage of variance explained by the PCA dimensions



(b) Projection of the original features on the first 2 PCA dimensions

Figure 16: PCA results

Figure 15 shows that by projecting the whole dataset on the first 5 PCA dimensions, we can retain more than 80% of the original variance.

K-Means clustering. Since the dataset dimension is is significantly lower than the previous case, the clustering has been performed on both the original variables space and their projection on the first 5 principal components.

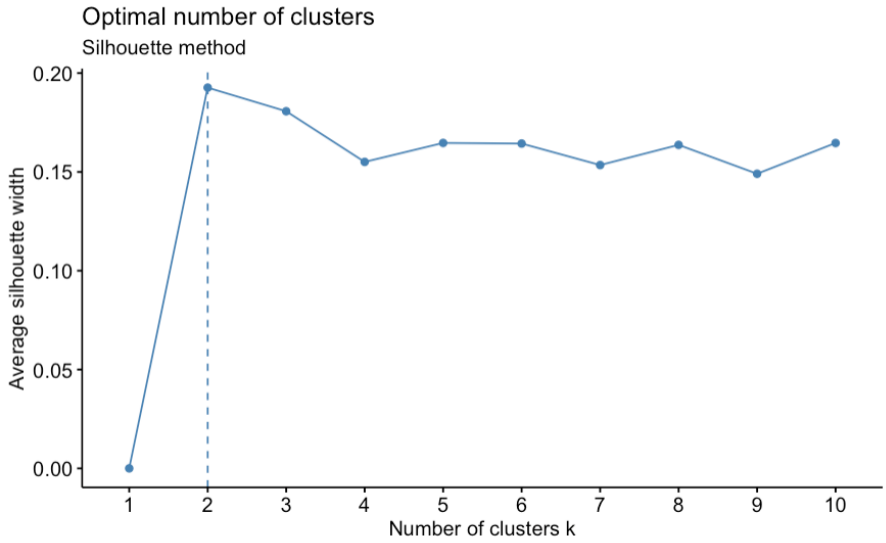
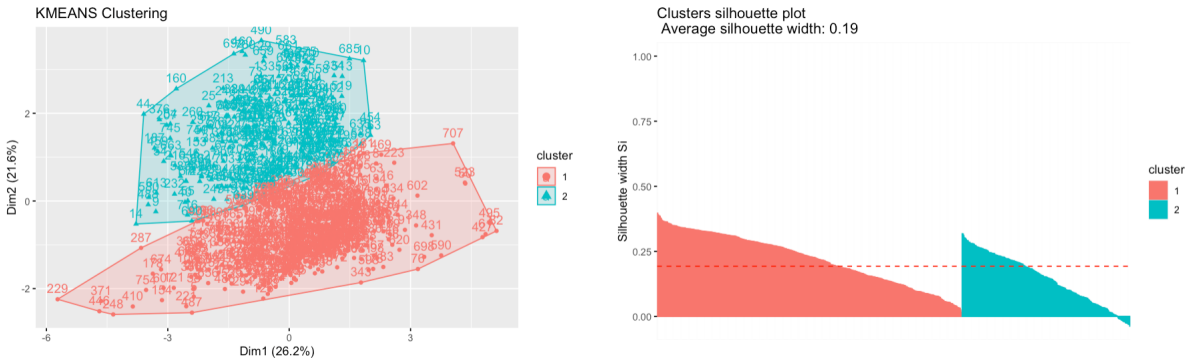


Figure 17: Average silhouettes values in function of the number of clusters.

As we can see in figure 17, the optimal number of cluster is 2. The results of the clustering in the original components space is showed in figure 18. As it is possible to

note, this time the clustering led to good results: the algorithm managed to find two well separated clusters and almost no overlapping.

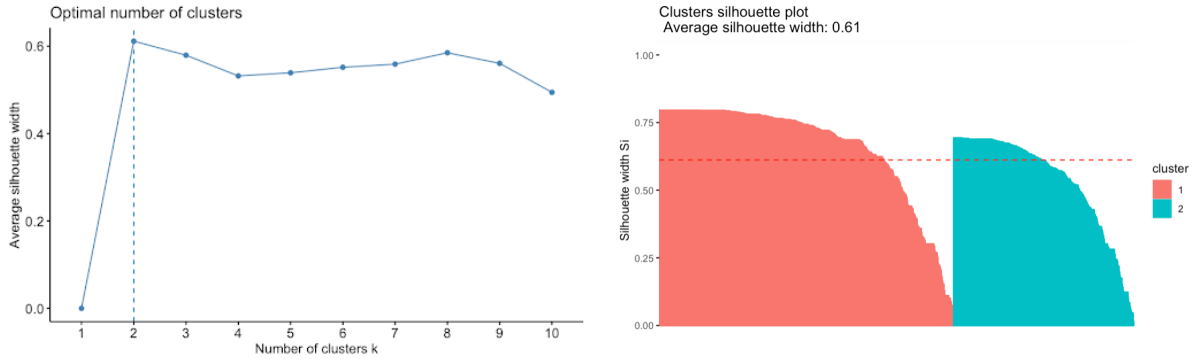


(a) First two components projection

(b) Silhouettes

Figure 18: K-means results on the whole dataset

The same analysis has been carried out with the principal components projected variables.



(a) Optimal k

(b) Silhouettes

Figure 19: K-means results on the projected dataset

The results are expressed by the figure 19 plots, that, also in this case, show a highly consistent clustering (with $k = 2$).

7 Learning and Predicting

Being the output a boolean value, the first regressor we test is the logistic regressor introduced before. After that, we explore two more learning models:

- a K-Nearest Neighbours (K-NN), that, due to lower dimension and more separable data, appears more suitable for this dataset than for the stock dataset,

- a fully connected Neural Network, with binary output.

All this model were evluated with 5-fold Cross Validation, as introduced in section 4.

7.1 Logistic Regression

The performance of the logistic regressor was acceptable, with an error rate of $\sim$**22%**. To improve the accuracy, we tried a conditional log-transformation for the skewed columns, but the results did not change.

Figure 20 shows the coefficients learnt by the Logistic regressor, highlighting the contribute of each variable to the result. Of course this should not be intended as the direct impact of a parameter on developing diabetes (insulin being negatively correlated to diabetes would be a strange medical result).

Figure 20: Logistic Regression coefficients

7.2 k-Nearest Neighbours

k-NN requires first of all a tuning of the hyperparameter k. This is as well carried out through 5-fold Cross Validation and the optimal value is $k = 21$, presenting an error rate of $\sim$**23.5%**.

Figure 21 presents the accuracies obtained during this hyperparameter search.

7.3 Neural Network

As for the stock dataset, a fully connected neural network was tested to make a comparison with other regressors. Even after increasing the dimension of the network, the accuracy of this model did not significantly overcome the others.

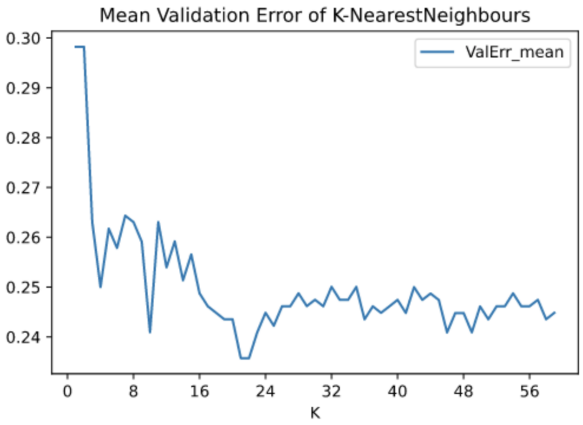


Figure 21: Logistic Regression coefficients

8 Conclusions

The Stock dataset and the Diabetes dataset prove to be very different.

First of all, the considerable dimension of the stock dataset affected the possibility of obtaining robust clusters, even after log-scaling the skewed columns and dropping the outliers. The high dimensional space of the input data may not be the only cause of this result: the stock dataset includes many complex econometric measurements and, as plausible in real datasets, the patterns may be very sparse, without natural clusters to distinguish. Moreover, the relations between the columns may be way more complex than linear correlation: in this case, heuristic pre-processing of the data might solve part of the issue.

On the other side, the diabetes dataset was perfectly clustered by 2-means algorithm and appeared simpler.

Regarding the main goal of this work, the prediction of the output, we obtained different quantitative outcomes for the two datasets: the stock dataset was basically impossible to regress (either with binary or numerical output), whereas the diabetes dataset the accuracy almost reached the 80%.

However, for both the datasets, the impression was that there actually exists an intrinsic information limit which no regressor (even a big neural network with all its flexibility) is able to overcome. The reasons behind this is probably that the input measurements are not representative enough of the phenomena to perform accurate predictions. Expanding the datasets with more patterns may improve the results, but will never fill the lack of representation that the input space presents when aiming to predict the output.

References

[1] Ian R. White, Patrick Royston, and Angela M. Wood. Multiple imputation using chained equations: Issues and guidance for practice. *Statistics in Medicine*, 30(4):377–399, 2011.