Tesina

Angelica Urbanelli S271114

Data Spaces, a.y. 2020/2021

Contents

1 Introduction

-spiegare il problema importanza di distinguere buoni e cattivi pagatori statistiche?
-interpretabilità? -falsi buoni peggio di falsi cattivi
-tutto il codice si trova nella repo github xx -per navigarlo usare il link xx di jupyternotebook

2 Dataset description

2.1 History

The dataset used comes from the UCI Machine Learning Repository [3], under the name "South German Credit (UPDATE) Data Set" [5].
Ulrike Grömping, professor at the Beuth University in Berlin, in her paper [4] provides the history of this dataset, her considerations about the data and corrections on the code table.
Basically, the data come from a large regional bank in the southern Germany that have been collected from 1973 to 1975, and have been originally provided to UCI in 1994 by Professor Dr. Hans Hofmann from Hamburg University [2] as part of a group of datasets in the context of the EU Statelog Project.
Because of many inconsistencies, found while trying to interpret the final results of her experiments, Grömping decided to research the story of this data, that she found in the German literature together with the same dataset with some differences. These informations helped her to fix the code table (a file that explains the encoding of categorical variables) of this dataset and consequently to provide the correct one (now attached in the .zip downloadable from UCI).
Grömping also explained that it was worth it because, although the dataset contains very old data, it is widely used in many researches in the domain of interpretable machine learning, indeed there are various R packages that include this data. In addition, it *is one of the few data sets on credit scoring that has a meaning attached to variables and their levels*, which is a very important feature when using this kind of data to do experiments whose interpretability is a key point of research.

2.2 General structure

The dataset contains 1000 samples, each one characterized by 20 features and classified as `good` or `bad` `credit risk`, in particular there are 700 good ones and 300 bad ones [Figure 1]. Customers with good credits perfectly complied with the conditions of the contract, while customers with bad credits did not comply with the contract as required.
As reported in the aforementioned paper [4], the actual percentage of bad credits was around 5%, and examples of bad credit risk have been heavily oversampled.

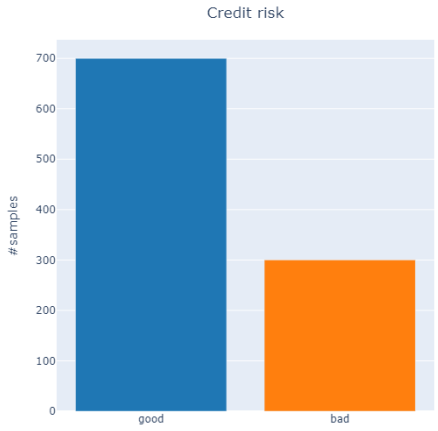

Figure 1: `good` and `bad` credit risk distribution

2.3 Features description

Among the 20 features, there are 3 numerical discrete variables:

- **duration**: credit duration in months
- **amount**: credit amount in DM[1]; original values are not available, the ones present in the dataset are the result of an unknown monotonic transformation
- **age**: age of the debtor in years

10 ordinal variables; most of theme were numerical ones on which binning has been applied, that means that they have been aggregated into a fixed number of intervals, so that they can be treated as ordinal features:

- **employment_duration**: duration of debtor's employment with current employer (unemployed; <1 year ; ≥ 1 and <4 years; ≥ 4 and <7 years; ≥ 7 years)
- **installment_rate**: credit installments expressed as a percentage of debtor's disposable income (≥ 35; ≥ 25 and <35; ≥ 20 and <25; <20); it is the only ordinal feature expressed in a decreasing order
- **present_residence**: from how many years the debtor lives in the present residence (<1 year ; ≥ 1 and <4 years; ≥ 4 and <7 years; ≥ 7 years)
- **number_credits**: number of credits including the current one the debtor has (or had) at this bank (1; 2 or 3; 4 or 5; ≥ 6)
- **people_liable**: number of people who financially depend on the debtor (i.e., are entitled to maintenance) (from 0 to 2; 3 or more)
- **status**[2]: status of the debtor's checking account with the bank in DM (no checking account; <0; $0 \leq ... < 200$; ≥ 200 / salary for at least 1 year)
- **savings**[2]: debtor's savings in DM (unknown/no savings account; <100; $100 \leq ... < 500$; $500 \leq ... < 1000$; ≤ 1000)
- **credit_history**[2]: history of compliance with previous or concurrent credit contracts (delay in paying off in the past; critical account/other credits elsewhere; no credits taken/all credits paid back duly; existing credits paid back duly till now; all credits at this bank paid back duly)
- **job**: quality of debtor's job (unemployed/unskilled - non-resident; unskilled - resident; skilled employee/official; manager/self-employed/highly qualified employee)
- **property**: the debtor's most valuable property, i.e. the highest possible code is used (unknown / no property; car or other [savings don't fall into this category]; building society savings agreement (mortgage)/life insurance; real estate)

7 categorical variables:

- **purpose**: purpose for which the credit is needed (others; car (new); car (used); furniture/equipment; radio/television; domestic appliances; repairs; education; vacation; retraining; business)
- **personal_status_sex**: combined information on sex and marital status; sex cannot be recovered from the variable because male singles and female non-singles are coded with the same code; in addition, female widows are not listed in any of the categories (male divorced/separated; female non-single or male single; male married/widowed; female single)

[1] stands for Deutsche Mark, was the official currency of West Germany from 1948 until 1990 and later the unified Germany from 1990 until 2002 [10]

[2] Those are considered as categorical by Grömping [4], but in my opinion their labels can be ranked and also considering that to properly manage categorical features an encoding have to be done, thus likely this brings to a huge number of features. Possibly, the position of *no checking account* in `status` feature could be discussed with a domain expert.

- **other_debtors**: whether there is another debtor or a guarantor for the credit (none; co-applicant; guarantor)
- **other_installment_plans**: installment plans from providers other than the credit-giving bank (bank; stores; none)
- **housing**: type of housing the debtor lives in (for free; rent; own)
- **telephone**: whether there is a telephone landline registered on the debtor's name; of course this variable would have no meaning nowadays, but this data come from 1970s (yes; no)
- **foreign_worker**: whether the debtor is a foreign worker (yes; no)

Figure 2: Features' distributions

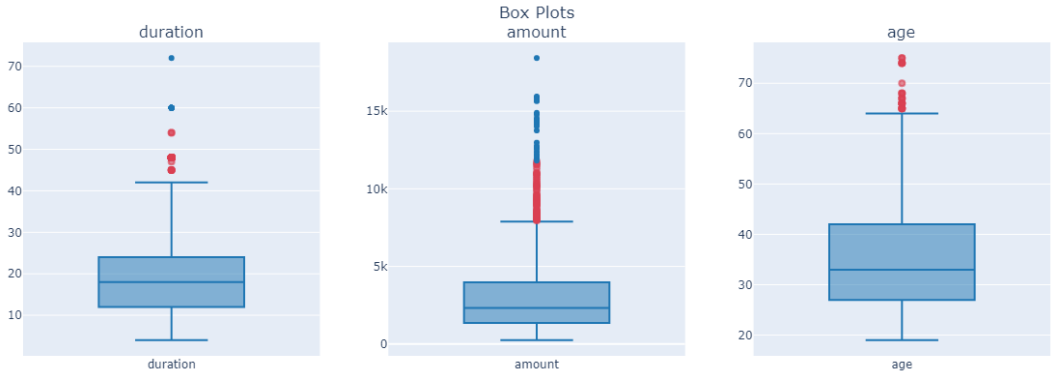

Figure 3: Box Plots of `duration`, `amount` ad `age`

3 Dataset Analysis

3.1 Features' distributions

Here [Figure 2] the distribution of the various features can be seen. Some of them are highly imbalanced, for instance it can be noticed that almost all customers are not foreigners and have neither another debtor nor a guarantor for the credit.
It might be interesting to take a look at the box plots of the numerical discrete features: `duration`, `amount` ad `age` [Figure 3]. Given that Q_1 and Q_3 are, respectively, the first and the third quartile, and that the interquartile range $IQR = Q_3 - Q_1$, in the chosen representation the whiskers are: the largest observed point that falls within Q_3 and $Q_3 + 1.5 \cdot Q_3$ and the lowest observed point that falls within Q_1 and $Q_1 - 1.5 \cdot Q_1$ [9]. The single points, instead, are highlighted in red if they fall within the lowest whisker and $4 \cdot Q_1 - 3 \cdot Q_3$ or within the highest whisker and $4 \cdot Q_3 - 3 \cdot Q_1$ [8], those are called *suspected* outliers; while the blue points are the ones outside these ranges, thus they can be considered outliers beyond any doubt.
In this case, since those plots represent the univariate distribution, those points are not considered as outliers; they will be better evaluated and managed in section 4.3.

3.2 Features distribution per class

In figure [4] and [5], the same plots of section 3.1 are shown, highlighting the distributions separately for the two class labels.

4 Data preprocessing

4.1 Labels encoding

All the features' labels are expressed as integers: both categorical and ordinal ones have been previously mapped to integers by the donor of the data, using label encoding either from 0 to N-1 (only for **purpose** and **credit_history**) or from 1 to N (all the other variables) where N is the number of labels for a certain feature. Apparently there is no particular reason for this distinction. For what concerns class labels (`credit risk`), they have been mapped to 0 for `bad` and 1 for `good`.

In order to make variables more uniform, two small changes have been done:

- the variable **installment_rate** is the only one among the ordinal variables having a decreasing order, so its mapping has been inverted; thus now label `1` means < 20, label `2` means ≥ 20 and < 25 and so on;

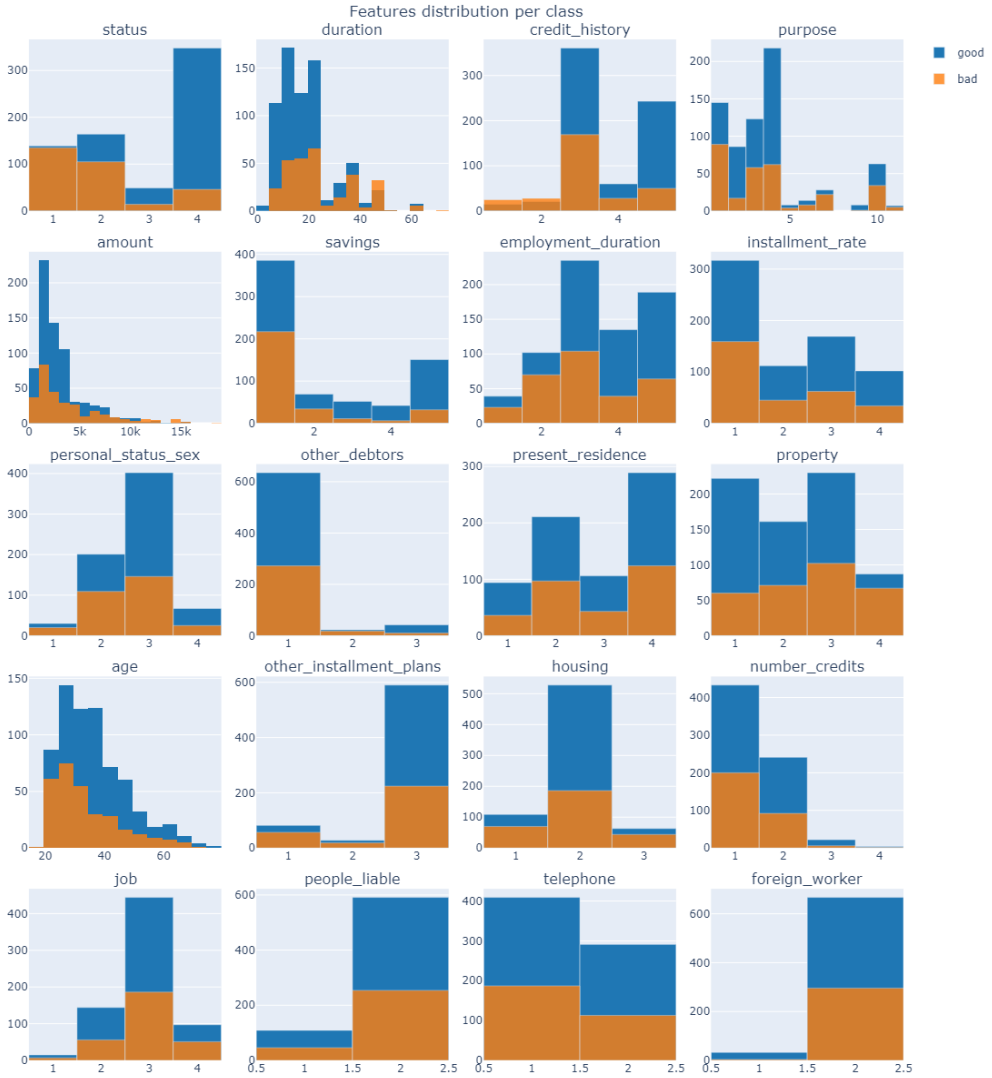

Figure 4: Features' distributions per class

- **credit_history** and **purpose**'s mappings have been changed from $0 \to N-1$ to $1 \to N$.

In addition, all categorical variables have to be encoded in a different way with respect to the actual one ($1 \to N$). Indeed this encoding gives an arbitrary ranking to features that do not have one, and this is a problem when applying distance-based classification algorithms because those could potentially exploit this fictional structure created by the mapping itself.

4.2 One hot encoding vs binary encoding

In order to perform this mapping, two encoders have been taken into account

One hot encoding For each unique value in a variable, a new column is created, whose values are either 1s or 0s, depending on whether the value matches the column header.
It is very simple, it does not interfere with interpretability since every new column has a specific meaning, and it allows very well to separate categorical features' labels. However, the downside is that we may end up with a huge number of features, especially if we need to map variables with an high number of labels.

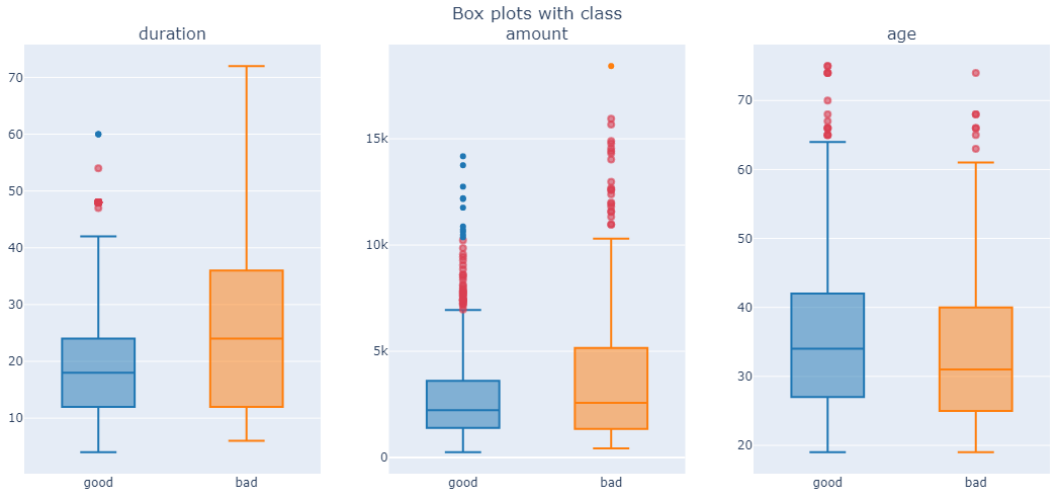

Figure 5: Box Plots of `duration`, `amount` ad `age` divided by class

Binary encoding Values are firstly converted into their binary code, and then the digits are split into separate columns.
With this method, the overhead due to the increase in the number of features is limited, because a feature with N distinct values is mapped into $\log_2 N$ columns instead of N. The drawbacks are that the resulting features still have a weak binding between them and the new columns do not really have a specific meaning on their own.

Given these considerations, among the categorical variables:

- **purpose** (11 labels) has been encoded with binary encoding;
- **other_debtors**, **other_installment_plans**, **housing** and **personal_status_sex** (respectively with 3, 3, 3 and 4 labels) have been encoded using one hot encoding, since the benefit of binary encoding is not worthwhile with these small number of labels;
- **telephone** and **foreign_worker** have only 2 labels, therefore do not need any encoding.

Moreover, since one of the 3 labels of **other_debtors** is `none`, it can be encoded with only two columns, where `none` is encoded with both at 0.

4.3 Missing values and outliers

The dataset does not contain any missing value.

Regarding outliers detection, by looking at figures [3] and [5] (for more boxplots see:) it might seem that the dataset has a huge number of outliers; however, observing one feature at a time can be misleading, since the whole situation should be taken into account. For this reason, instead of considering the univariate distributions separately, can be a good idea to perform a multivariate outliers detection.
For this purpose, various distance metrics and techniques exist. Among them, Mahalanobis Distance (introduced by Mahalanobis in 1936 [1]) is an effective one, its goal is to find the distance between a point $\vec{x}$ and a distribution. Differently from other techniques, it can manage distributions where every feature has a different scale and variance. Furthermore, by using the covariance matrix, it is able to detect outliers basing on the distribution of points, unlike e.g. the Euclidean distance [Figure 6].

Given a point $\vec{\mathbf{x}} = (x_1, x_2, ..., x_N)^T$ extracted from a distribution of m points in $\vec{\mathbf{X}} = (X_1, X_2, ..., X_N)^T$ where $X_1, X_2, ..., X_N$ are the random variables of the dataset; given $\vec{\mu} = (\mu_1, \mu_2, ..., \mu_N)^T$ the mean of the

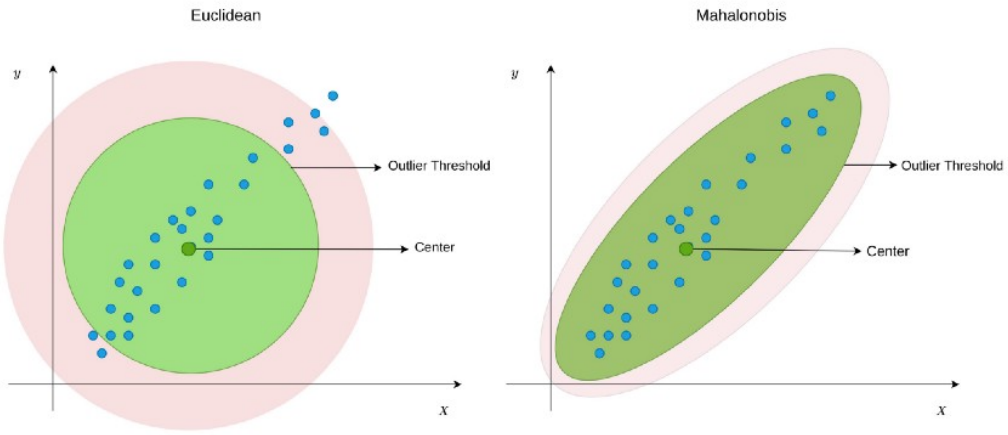

Figure 6: Euclidean distance vs Mahalonobis distance, image by [6]

set of observations whose entries $\mu_i = \frac{1}{m}\sum x_i$, and the covariance matrix Σ whose entry

$$\Sigma_{(i,j)} = cov[X_i, X_j] = E[(X_i - E[X_i])(X_j - E[X_j])] \tag{1}$$

thus, the Mahalanobis Distance from a point $\vec{\mathbf{x}}$ and the set of points it has been extracted from is

$$d_M(\vec{\mathbf{x}}, \vec{\mu}) = \sqrt{(\vec{\mathbf{x}} - \vec{\mu})^T \Sigma^{-1} (\vec{\mathbf{x}} - \vec{\mu})} \tag{2}$$

The covariance matrix (equation 1) is estimated using the `numpy.cov` function, see the documentation [7] and the code for better understanding.

Once computed the Mahalanobis distance for every point, its distribution can be evaluated to detect multidimensional outliers. As can be seen in figures [7] and [8], there are a few points (seven) whose distance is greater than $Q_3 + 1.5 \cdot Q_3$, thus they could be considered as multidimensional outliers. However, they are a very small number of points and, among those, four have class label `0` (`bad` credit risk), that is the one with the lowest number of observations. That means that they could seem outliers because there are not so many samples of this label (considering that the ones present have already been oversampled), thus they could be significant points in detecting bad credit risk. In addition, all those points fall in the range of the so called *suspected outliers*, while there are no points $< 4 \cdot Q_1 - 3 \cdot Q_3$ or $> 4 \cdot Q_3 - 3 \cdot Q_1$ (that are the ranges where points are very likely to be outliers). For these reasons, those seven points are not removed from the dataset.

4.4 Data normalization

When performing analysis with datasets whose features have different scales, data normalization is a very important step in the preprocessing procedure. Indeed, especially when applying some specific techniques that rely on distance or variance (e.g. distance-based classification algorithm, PCA and so on), if features are not rescaled, the distance between points and data variance are more affected by those features that have larger scale or higher values. Thus, the features end up not to have the same importance.
In this case, the min-max scaling has been applied to features, that is, for each feature X, the new one $\hat{X}$:

$$\hat{X} = \frac{X - min(X)}{max(X) - min(X)} \tag{3}$$

in this way, every feature has been rescaled in the range $[0, 1]$.

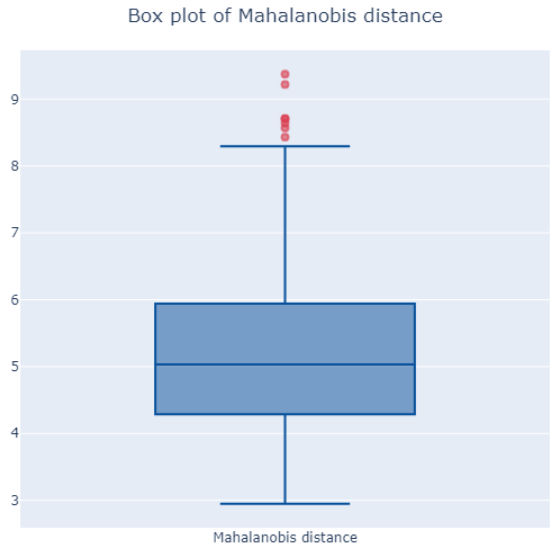

Figure 7: Box plot of Mahalonobis distance distribution

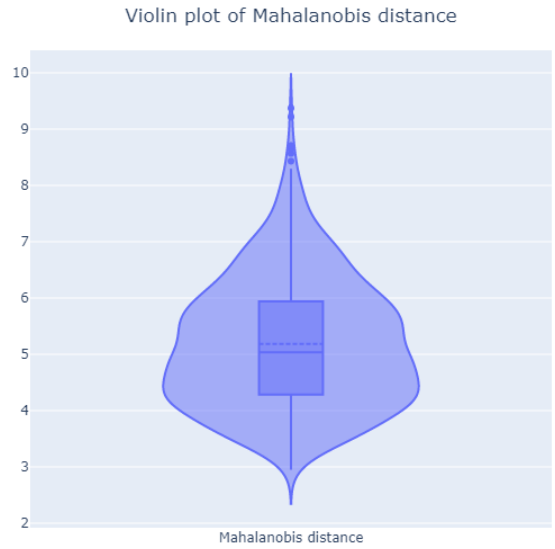

Figure 8: Violin plot of Mahalonobis distance distribution

Figure 9: Training set

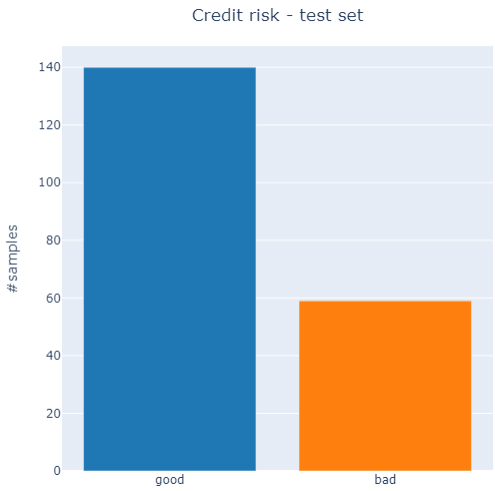

Figure 10: Test set

4.5 Training-test split

To proceed in the classification phase, the dataset has been split into training and test set respectively for 80% and 20% in a stratified way, that means that in both sets, the proportion of `good` and `bad` credit risk of the original dataset has been kept [Figure 9 and 10]. This is particularly important, because the test set (the one on which the trained classification algorithm will be evaluated), should be as much similar as possible to real data, that means keeping the proportion of label distribution, do not applying any further transformation to those data and do not use them during training.

5 Dimensionality reduction

5.1 Curse of dimensionality

After the encoding phase, the number of features (label excluded) went from 20 to 31. Of course, the encoding was needed, but in data analysis having to deal with many features can be problematic for a lot of reasons. Indeed this is called *the curse of dimensionality*. When the dimensions increase, the volume of the hypercube containing the data increases as well, so all the points become more distant between them, thus making the dataset very sparse. To overcome this problem, the number of data points should increase exponentially with the dimension. Basically, all points become distant and at the same time they have similar distances among them; therefore, the concepts of nearest neighbours and distance become pointless. So, in cases like this, it is always a good idea to take into account using a dimensionality reduction technique.

5.2 Correlation based

5.3 PCA

Principal Component Analysis is one of the most popular and simple techniques of dimensionality reduction. Its goal is to find a linear mapping that project the m data points in a low dimensional space, from d features to n, with $n << d$, in such a way that the information lost is as small as possible.
This technique has two formulations that lead to the same solution from different points of view.

- the **minimum error** formulation, that finds a solution to the problem

$$\underset{U \in \mathbb{R}^{n,d}}{\operatorname{argmin}} \sum_{i}^{m} \left\| \vec{\mathbf{x}}_i - UU^T \vec{\mathbf{x}}_i \right\|_2^2, \quad \text{subject to } U^T U = \mathbb{I} \tag{4}$$

  The idea is that we want minimise the error, that is the information we loose, in projecting the data in a lower dimensional space. Indeed, the error is computed between each original point $\vec{\mathbf{x}}$ and its reconstruction $\hat{\vec{\mathbf{x}}} = UU^T \vec{\mathbf{x}}$.
  Assuming that the data points are centred ($\vec{\mu} = \vec{\mathbf{0}}$), and given the matrix of data points $X \in \mathbb{R}^{m,d}$, m number of points and d number of features, the solution to find the linear mapping (the orthogonal matrix U^T) is finding the n eigenvectors $(\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, ..., \vec{\mathbf{u}}_n)$ corresponding to the n biggest eigenvalues $(\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq ... \geq \lambda_d)$ of the covariance matrix of X. Those eigenvectors (the *principal components*) are the column of the matrix U, and the sum of the remaining eigenvalues is the error done in projecting the data in low dimensional space. However, since the data distribution is not known, the covariance matrix is estimated through the scatter matrix, also called sample covariance matrix

$$S = X^T X \tag{5}$$

  still assuming $\vec{\mu} = \vec{\mathbf{0}}$.

- the **maximum variance** formulation, that aims to find as principal components the directions where the variance of the projection of data points is maximized, since the idea is that in the directions where the variance is maximum, are the ones where also the information carried is maximal. Those directions are found in the following way: given the matrix of data points $X \in \mathbb{R}^{m,d}$, and given the d random

variables $\vec{\mathbf{z}}_1, \vec{\mathbf{z}}_2, ..., \vec{\mathbf{z}}_d$ from which the columns of X has been sampled, the first principal component $\vec{\mathbf{u}}_1 = (\alpha_1, \alpha_2, ..., \alpha_d)$, $\alpha_i \in \mathbb{R}$ is the one that satisfies (6).

$$\begin{cases} \underset{\vec{\mathbf{u}}_1}{\text{argmax}}\, \text{Var}(\sum_i^d \alpha_i z_i) \\ \sum_i^d \alpha_i^2 = 1 \end{cases} \tag{6}$$

$$\begin{cases} \underset{\vec{\mathbf{u}}_2}{\text{argmax}}\, \text{Var}(\sum_i^d \alpha_i z_i) \\ \sum_i^d \alpha_i^2 = 1 \\ \langle \vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2 \rangle = 0 \end{cases} \tag{7}$$

Then, once found, the second principal component is the vector in the second best direction in terms of variance, and that is orthogonal with respect to the first one. Thus $\vec{\mathbf{u}}_2 = (\alpha_1, \alpha_2, ..., \alpha_d)$ must satisfy (7), and so on until the n principal components are found, each one satisfying the constraints of equation 6 and the constraint of being orthogonal to the previous principal components.
By using Lagrange multipliers, this turns out finding $\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, ..., \vec{\mathbf{u}}_n$ by solving

$$S\vec{\mathbf{u_i}} = \lambda_1 \vec{\mathbf{u_1}}, \quad \text{subject to } \|\vec{\mathbf{u}}_i\| = 1 \text{ and } \vec{\mathbf{u}}_i \perp \vec{\mathbf{u}}_1, ..., \vec{\mathbf{u}}i - 1 \tag{8}$$

Where S is again the sample covariance matrix (equation 5), that means that $\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, ..., \vec{\mathbf{u}}_n$ are the eigenvectors of the matrix S, corresponding to the eigenvalues $\lambda_1, lambda_2, ..., \lambda_n$. In addition, since the variance of the points projected along $\vec{\mathbf{u}}_i$ can be rewritten as $\vec{\mathbf{u}}_i^T S \vec{\mathbf{u}}_i$, each eigenvalue of its corresponding principal component is exactly equal to the variance of the dataset along that direction, and, in particular, the first n eigenvectors found with the maximization variance method, are the ones corresponding to the n maximum eigenvalues of the matrix S. Thus, the two methods are exactly equivalent.

In order to decide how many principal components to take, a trade off must be found in order to take neither too little information, nor too many features. Here [Figure 11] both the explained variance (the eigenvalues

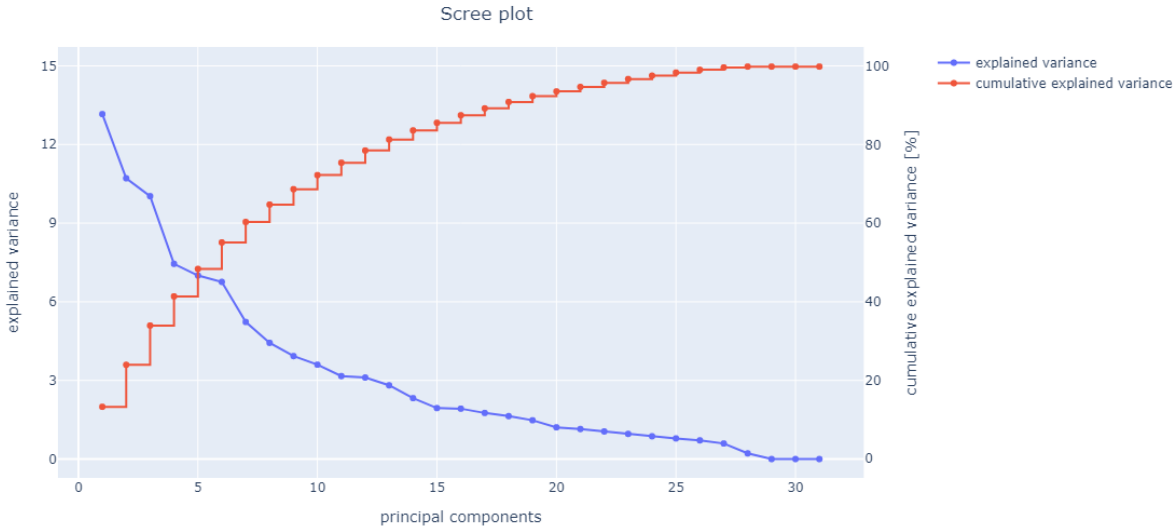

Figure 11: Scree plot of principal components

drown in descending order) and the cumulative explained variance (for each eigenvalue, its ratio w.r.t. the sum of all eigenvalues is computed, then drown in a cumulative way). In this case, since there is an elbow at 15 principal components, and since at that point the cumulative explained variance is sightly higher that 85%, I decided to take 15 as number of principal component to keep.

5.4 KPCA

5.5 mRMR

-correlazione variabili -t-SNE -eliminare/aggregare variabili correlate -pca per capire quali sono le feature più importanti -lineare -kpca -eventualmente di nuovo correlazione tra variabili

6 dataset balancing

-oversampling -undersampling -smote

7 classificazione

-logistic regression (also + regularization ridge or lasso) -tree -random forest (less simple to explain results) -svm (linear, rbf, polinomial, sigmoid) -knn -fda -qda? -lda? bayes? -eventually simple mlp

8 results

-per ogni combinazione (dim-reduction - algoritmo) fare ROC, accuracy, confusion matrix, recall, precision, F1 (armonic mean between precision and recall) -istogrammi ?

References

[1] P. C. Mahalanobis. "On the generalised distance in statistics". In: *Proceedings of the National Institute of Sciences of India 2*. Vol. 1. 1936, pp. 49–55.

[2] H. Hofmann. *Statlog (German Credit Data) Data Set*. 1994. URL: https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29.

[3] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[4] U. Grömping. *South German Credit Data: Correcting a Widely Used Data Set*. Tech. rep. 4/2019, Reports in Mathematics, Physics and Chemistry, Berlin: Department II, Beuth University of Applied Sciences, Apr. 2019. URL: http://www1.beuth-hochschule.de/FB_II/reports/Report-2019-004.pdf.

[5] U. Grömping. *South German Credit (UPDATE) Data Set*. 2020. URL: http://archive.ics.uci.edu/ml/datasets/South+German+Credit+%28UPDATE%29.

[6] Sergen Cansiz. *Multivariate Outlier Detection in Python*. URL: https://towardsdatascience.com/multivariate-outlier-detection-in-python-e946cfc843b3.

[7] numpy documentation. *numpy.cov*. URL: https://numpy.org/doc/stable/reference/generated/numpy.cov.html.

[8] Plotly. *Box Plots in Python*. Accessed: 2021-08-09. URL: https://plotly.com/python/box-plots/.

[9] Wikipedia, the free encyclopedia. *Box plot*. Last edit: 2021-07-07 Accessed: 2021-08-09. URL: https://en.wikipedia.org/wiki/Box_plot.

[10] Wikipedia, the free encyclopedia. *Deutsche Mark*. Last edit: 2021-07-10 Accessed: 2021-08-09. URL: https://en.wikipedia.org/wiki/Deutsche_Mark.