Cervical Cancer (Risk Factors)

Politecnico di Torino

Xileny Seijas Portocarrero
s258129

Sharova Ekaterina
s270231

Introduction

The data is dedicated to classification problem related to the Cervical Cancer, which is one of the most treatable cancers when diagnosed at early stages. The rising global incidence of cervical cancer is estimated to have affected more than 600 000 women and nearly 350 000 women are predicted to have died from the disease in 2021 alone. It is keep happening mainly because of the economic cost and the difficulties in implementing effectives screening programs. Data Mining provides robust tools to verify the known causal relations and assess risk factors from medical datasets. Classification models can then exploit this scenario to help identifying groups of population at higher risk to improve planning of screening programs.

Worldwide, this type of cancer causes makes up 8% of the total cancer cases and deaths. It's the fourth (second in developing countries) cause of death from cancer in women. A lot of progress has been made, and nowadays it is one of the most curable: when diagnosed and treated at the earliest stages, the five-year survival rate can reach 95%. In this work, we investigate the possibility of classifying high-risk patients from demographic and other medical data but excluding the results of other related exams.

Dataset description

The dataset was collected at "Hospital Universitario de Caracas" in Caracas, Venezuela. The dataset is available at:
http://archive.ics.uci.edu/ml/datasets/Cervical+cancer+%28Risk+Factors%29.

36 attributes describe demographic information, sexual activity and frequency, pregnancies, age, smoke addiction, usage of contraceptives and diagnosed Sexually Transmitted Diseases of 858 women patients. There are missing values in dataset due to the decision of the patients to not disclose those details because of privacy concerns.

Attributes description

We'll introduce some context and clarifications on the attributes significance to outline the relationship with the disease.

• **Hormonal Contraceptives:** indicates if the patient uses hormonal contraceptives. **IUD** indicates the usage of the Intra Uterine contraceptive Device.
• **STDs:** indicates if the patient had at least one Sexually Transmitted Disease. It's positive when at least one of the **STDs:condylomatosis**, **STDs:cervical condylomatosis**, **STDs:vaginal condylomatosis**, **STDs:vulvo-perineal condylomatosis**, **STDs:syphilis**, **STDs:pelvic inflammatory disease**, **STDs:genital herpes**, **STDs:molluscum contagiosum**, **STDs:AIDS**, **STDs:HIV**, **STDs: Hepatitis**, **STDs: HPV** attributes is positive.
• **STDs:HPV:** indicates if the patient is infected by the Human Papilloma Virus.
• **Dx:HPV:** reveals if the patient was already diagnosed with HPV in the past.

• **Dx:Cancer** is positive if the patient already had cancer. It is not clear if this refers to Breast Cancer, as diagnosable by the Oncotype DX test or any type of cancer. The original paper presenting this dataset has no additional information on this.
• **Dx:CIN:** indicates if the patient was affected by Cervical intraepithelial neoplasia (CIN).
• **Dx:** is positive if at least one of the Dx variables is positive.
• **Hinselmann and Citology:** are two tests to detect Cervical Cancer. **Schiller** is another test able to identify abnormal areas to be biopsied and examined. The attribute Hinselmann has been chosen as our target.
• **Biopsy** is a medical procedure providing a confirmation of the diagnosis through a colposcopy, a magnified visual inspection of the cervix.

index	0	1	2	3	4	5	6	7	8	9
Age	18	15	34	52	46	42	51	26	45	44
Number of sexual partners	4.0	1.0	1.0	5.0	3.0	3.0	3.0	1.0	1.0	3.0
First sexual intercourse	15.0	14.0	?	16.0	21.0	23.0	17.0	26.0	20.0	15.0
Num of pregnancies	1.0	1.0	1.0	4.0	4.0	2.0	6.0	3.0	5.0	?
Smokes	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0
Smokes (years)	0.0	0.0	0.0	37.0	0.0	0.0	34.0	0.0	0.0	1.266972909
Smokes (packs/year)	0.0	0.0	0.0	37.0	0.0	0.0	3.4	0.0	0.0	2.8
Hormonal Contraceptives	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0
Hormonal Contraceptives (years)	0.0	0.0	0.0	3.0	15.0	0.0	0.0	2.0	0.0	0.0
IUD	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	?
IUD (years)	0.0	0.0	0.0	0.0	0.0	0.0	7.0	7.0	0.0	?
STDs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs (number)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:condylomatosis	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:cervical condylomatosis	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:vaginal condylomatosis	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:vulvo-perineal condylomatosis	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:syphilis	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:pelvic inflammatory disease	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:genital herpes	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:molluscum contagiosum	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:AIDS	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:HIV	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:Hepatitis B	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:HPV	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Dataset cleaning

We decided to make some changes in order to make the dataset eligible for the following methods for data augmentation and classification.

This dataset contains 858 instances and 36 attributes (columns).

	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	IUD	...	STDs: Time since first diagnosis	STDs: Time since last diagnosis	Dx:Cancer	Dx:CIN
0	18	4.0	15.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?	?	0	0
1	15	1.0	14.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?	?	0	0
2	34	1.0	?	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?	?	0	0
3	52	5.0	16.0	4.0	1.0	37.0	37.0	1.0	3.0	0.0	...	?	?	1	0
4	46	3.0	21.0	4.0	0.0	0.0	0.0	1.0	15.0	0.0	...	?	?	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
853	34	3.0	18.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	?	?	0	0
854	32	2.0	19.0	1.0	0.0	0.0	0.0	1.0	8.0	0.0	...	?	?	0	0
855	25	2.0	17.0	0.0	0.0	0.0	0.0	1.0	0.08	0.0	...	?	?	0	0
856	33	2.0	24.0	2.0	0.0	0.0	0.0	1.0	0.08	0.0	...	?	?	0	0
857	29	2.0	20.0	1.0	0.0	0.0	0.0	1.0	0.5	0.0	...	?	?	0	0

858 rows × 36 columns

26 attributes are composed of data type and only 10 attributes were numeric type.

```
Data columns (total 36 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Age                                   858 non-null    int64
 1   Number of sexual partners             858 non-null    object
 2   First sexual intercourse              858 non-null    object
 3   Num of pregnancies                    858 non-null    object
 4   Smokes                                858 non-null    object
 5   Smokes (years)                        858 non-null    object
 6   Smokes (packs/year)                   858 non-null    object
 7   Hormonal Contraceptives               858 non-null    object
 8   Hormonal Contraceptives (years)       858 non-null    object
 9   IUD                                   858 non-null    object
 10  IUD (years)                           858 non-null    object
 11  STDs                                  858 non-null    object
 12  STDs (number)                         858 non-null    object
 13  STDs:condylomatosis                   858 non-null    object
 14  STDs:cervical condylomatosis          858 non-null    object
 15  STDs:vaginal condylomatosis           858 non-null    object
 16  STDs:vulvo-perineal condylomatosis    858 non-null    object
 17  STDs:syphilis                         858 non-null    object
 18  STDs:pelvic inflammatory disease      858 non-null    object
 19  STDs:genital herpes                   858 non-null    object
 20  STDs:molluscum contagiosum            858 non-null    object
 21  STDs:AIDS                             858 non-null    object
 22  STDs:HIV                              858 non-null    object
 23  STDs:Hepatitis B                      858 non-null    object
 24  STDs:HPV                              858 non-null    object
 25  STDs: Number of diagnosis             858 non-null    int64
 26  STDs: Time since first diagnosis      858 non-null    object
 27  STDs: Time since last diagnosis       858 non-null    object
 28  Dx:Cancer                             858 non-null    int64
 29  Dx:CIN                                858 non-null    int64
 30  Dx:HPV                                858 non-null    int64
 31  Dx                                    858 non-null    int64
 32  Hinselmann                            858 non-null    int64
 33  Schiller                              858 non-null    int64
 34  Citology                              858 non-null    int64
 35  Biopsy                                858 non-null    int64
dtypes: int64(10), object(26)
```

We did a conversion of all these columns to numeric data type and there were “?” characters, which we converted to the “**NaN”** values.

As a result, we transposed our date from this:

1 to

index	0	1	2	3	4	5	6	7	8	
Age	18	15	34	52	46	42	51	26	45	44
Number of sexual partners	4.0	1.0	1.0	5.0	3.0	3.0	3.0	1.0	1.0	3.0
First sexual intercourse	15.0	14.0	?	16.0	21.0	23.0	17.0	26.0	20.0	15.0
Num of pregnancies	1.0	1.0	1.0	4.0	4.0	2.0	6.0	3.0	5.0	?
Smokes	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0
Smokes (years)	0.0	0.0	0.0	37.0	0.0	0.0	34.0	0.0	0.0	1.266972909
Smokes (packs/year)	0.0	0.0	0.0	37.0	0.0	0.0	3.4	0.0	0.0	2.8
Hormonal Contraceptives	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0
Hormonal Contraceptives (years)	0.0	0.0	0.0	3.0	15.0	0.0	0.0	2.0	0.0	0.0
IUD	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	?
IUD (years)	0.0	0.0	0.0	0.0	0.0	0.0	7.0	7.0	0.0	?
STDs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs (number)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:condylomatosis	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:cervical condylomatosis	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:vaginal condylomatosis	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:vulvo-perineal condylomatosis	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:syphilis	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:pelvic inflammatory disease	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:genital herpes	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:molluscum contagiosum	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:AIDS	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:HIV	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:Hepatitis B	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
STDs:HPV	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

To this:

```
Age                                   0
Number of sexual partners            26
First sexual intercourse              7
Num of pregnancies                   56
Smokes                               13
Smokes (years)                       13
Smokes (packs/year)                  13
Hormonal Contraceptives             108
Hormonal Contraceptives (years)     108
IUD                                 117
IUD (years)                         117
STDs                                105
STDs (number)                       105
STDs:condylomatosis                 105
STDs:cervical condylomatosis        105
STDs:vaginal condylomatosis         105
STDs:vulvo-perineal condylomatosis  105
STDs:syphilis                       105
STDs:pelvic inflammatory disease    105
STDs:genital herpes                 105
STDs:molluscum contagiosum          105
STDs:AIDS                           105
STDs:HIV                            105
STDs:Hepatitis B                    105
STDs:HPV                            105
STDs: Number of diagnosis             0
STDs: Time since first diagnosis    787
STDs: Time since last diagnosis     787
Dx:Cancer                             0
Dx:CIN                                0
Dx:HPV                                0
Dx                                    0
Hinselmann                            0
Schiller                              0
Citology                              0
Biopsy                                0
dtype: int64
```

We found out that there were 2 columns with 787 NaN values. If 787 values of those columns are undefined there is no reason to use this data, since it's not relevant. We decided to drop these two columns. Also, many columns were with around 100 NaN values, but not more than 117 rows. We removed all the rows with NaN values also.

As a result, our cleaned dataset contains **668** number of **instances** and **34** number of **attributes**.

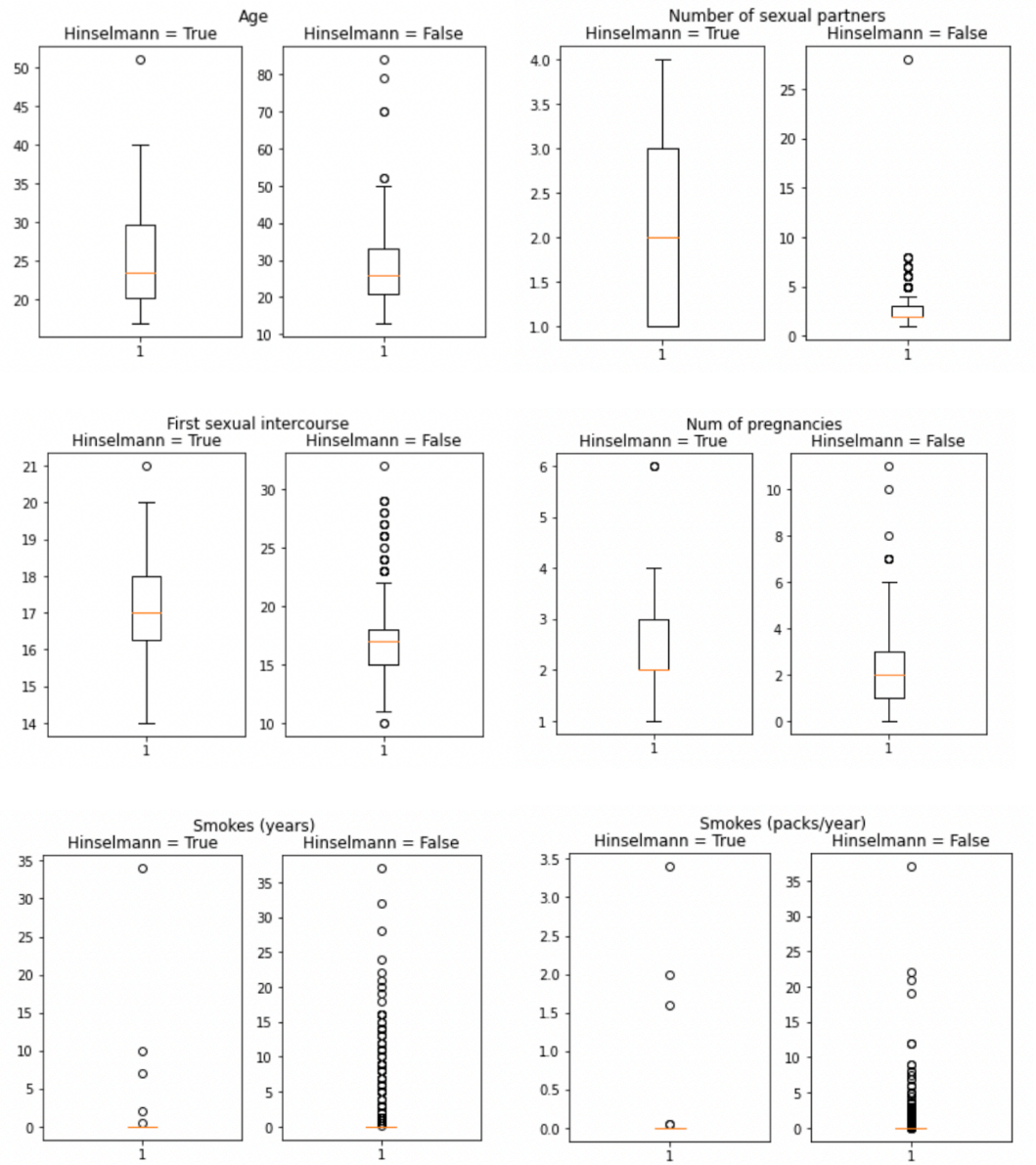
Before starting Dataset analysis we settled **binary categories**, which are: `'Smokes', 'Hormonal Contraceptives', 'IUD', 'STDs', 'STDs:condylomatosis', 'STDs:cervical condylomatosis','STDs:vaginal condylomatosis', 'STDs:vulvo-perineal condylomatosis', 'STDs:syphilis', 'STDs:pelvic inflammatory disease', 'STDs:genital herpes', 'STDs:molluscum contagiosum', 'STDs:AIDS', 'STDs:HIV', 'STDs:Hepatitis B', 'STDs:HPV', 'Dx:Cancer', 'Dx:CIN', 'Dx:HPV', 'Dx', 'Hinselmann', 'Schiller', 'Citology', 'Biopsy'`

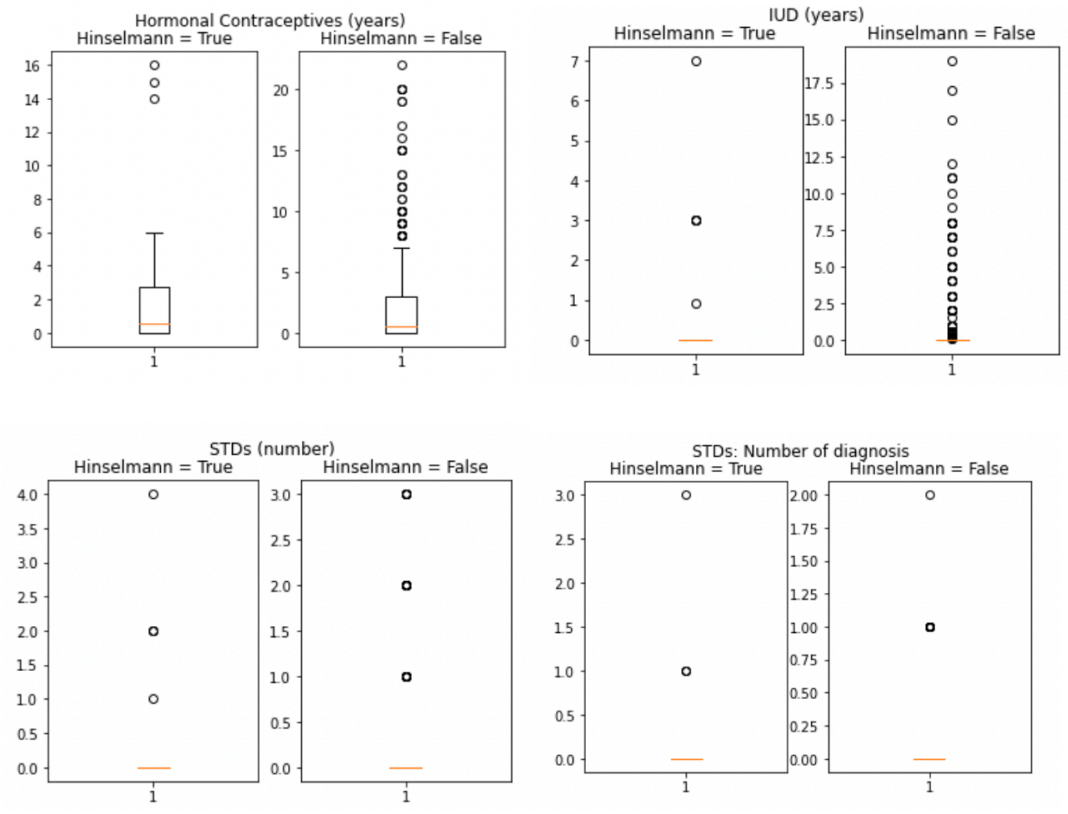
and **continuous categories:** `'Age', 'Number of sexual partners', 'First sexual intercourse', 'Num of pregnancies', 'Smokes (years)', 'Smokes (packs/year)', 'Hormonal Contraceptives (years)', 'IUD (years)', 'STDs (number)', 'STDs: Number of diagnoses'`].

The code for the above work can be found at:
https://github.com/s270231/ds_tesina_riskfactors/blob/master/0_data_cleaning.ipynb

Data analysis

In order to retrieve some information on how the attributes influence the exam result, we plotted the whisker boxplots of the continuous variables considering separately the cases in which the class label “Hinselmann” was positive and the cases in which it was negative. The figures are shown below:

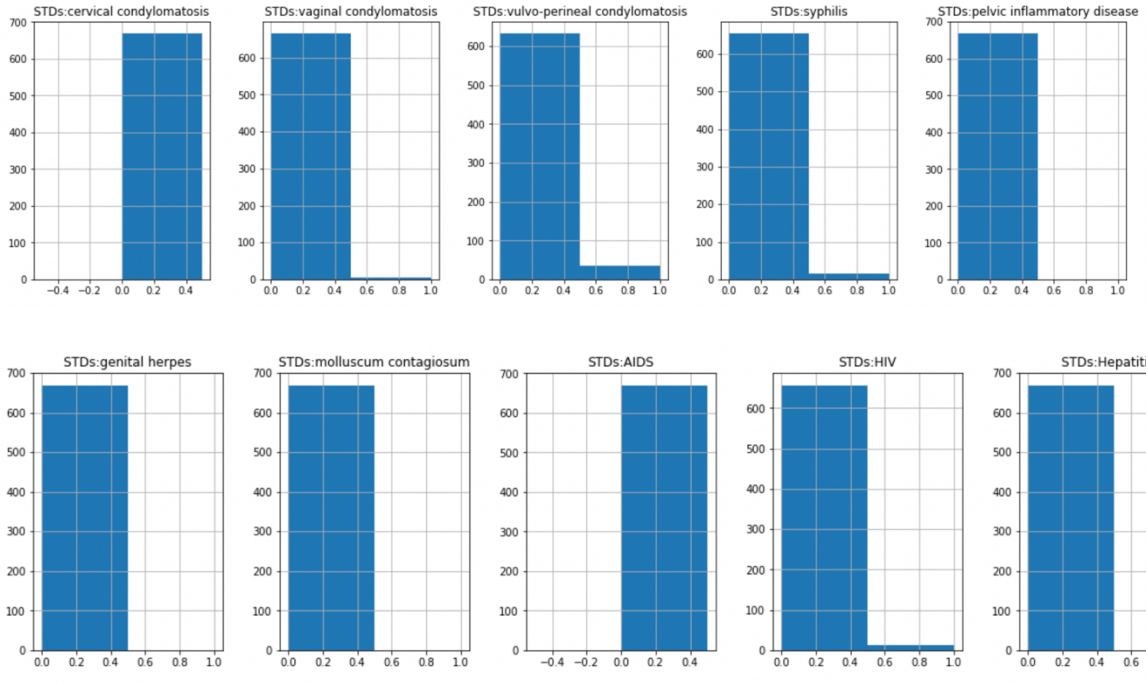


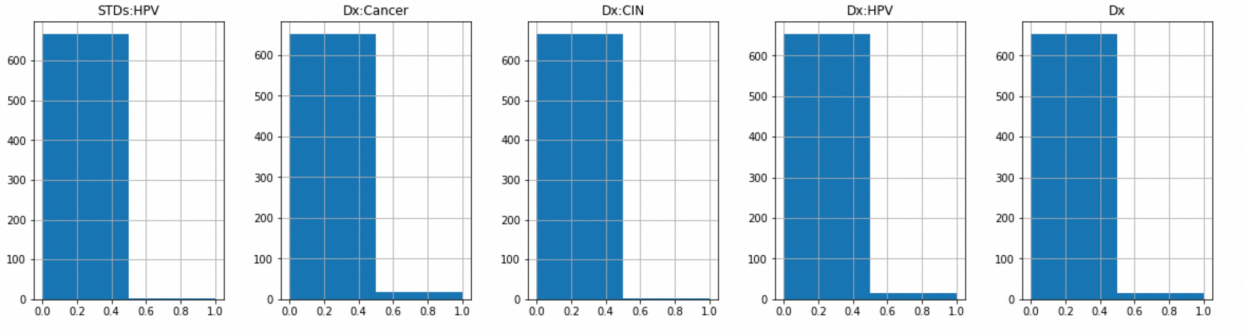


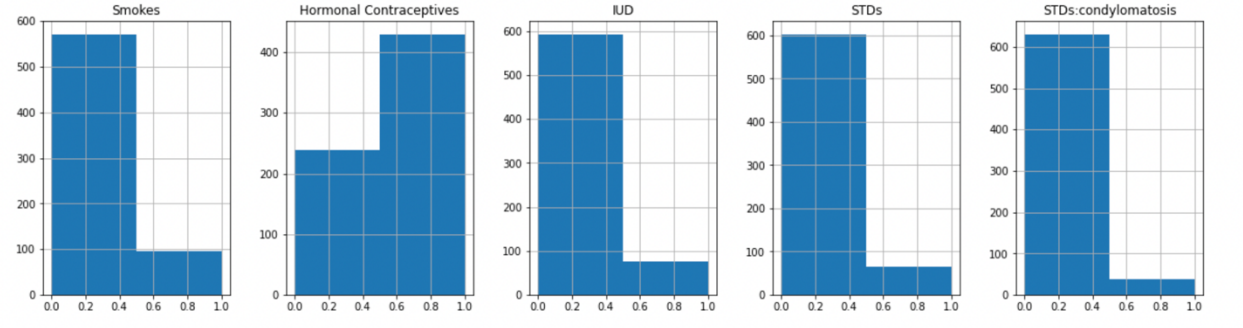
The result of the plotting shows that the *"Age", "Hormonal Contraceptives (years)", "Num of pregnancies"* attributes plays a more decisive role in the decision with respect to the other continuous attributes, as there are a more marked inter-class differences between the assumed values. This denotes the fact that there is a correlation between *"Age", "Hormonal Contraceptives (years)", "Num of pregnancies" attributes* and the class label.

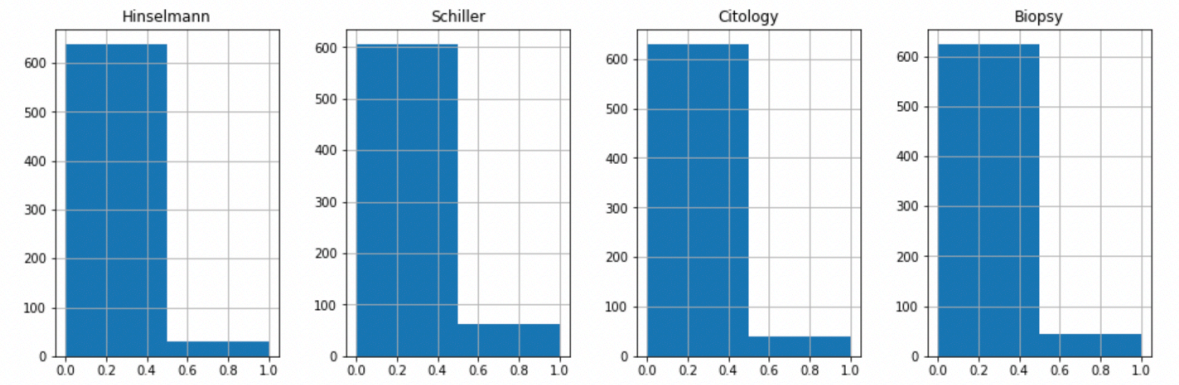
After that, in order to better visualize the distribution of all attributes, we plotted various histogram and bar graph.

For binary attributes:

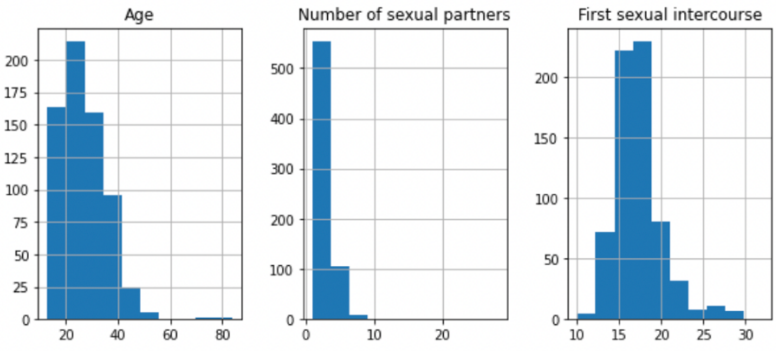


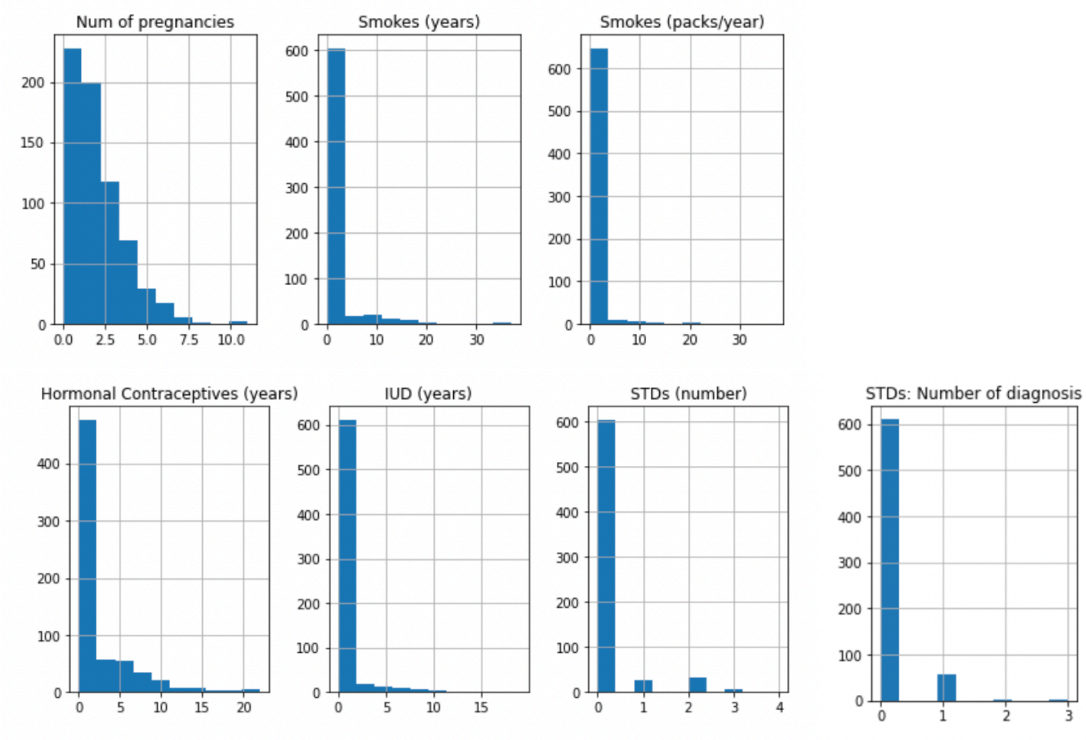






For continuous attributes:





Focusing on the label class *"Hinselmann"* we can note that the dataset is very unbalanced, since there are 30 positive samples and 638 negative samples. An analysis on unbalanced problems and the rebalancing solutions were explored in the next section.

In order to better understand the correlation between the different attributes, we used the correlation matrix (shown below). A **correlation matrix** is simply a table which displays the correlation coefficients for different variables. It is a powerful tool to summarize a large dataset and to identify and visualize patterns and relationships between the given data. The matrix has the features as rows and columns and the cells contain the correlation coefficients of the corresponding row and column.

The obtained correlation matrix shows that there are an obvious strong correlations between 'Smoke' Smoke (years)' and 'Smokes (packs/year)', also between 'Hormonal contraceptives' and 'Hormonal contraceptives (years)', and between 'STDs' and STDs:condylomatosis. But also, there is a correlation between 'Age' and 'Num of pregnancies'.

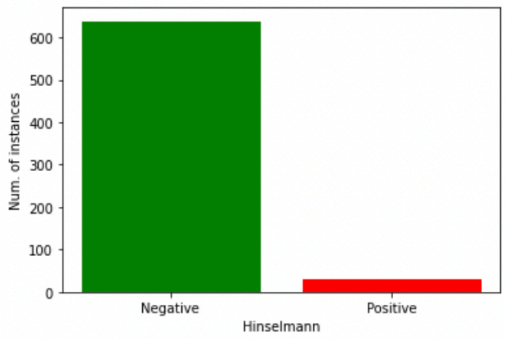
	Age	Number of sexual partners	First sexual intercourse	Num of pregnancies	Smokes	Smokes (years)	Smokes (packs/year)	Hormonal Contraceptives	Hormonal Contraceptives (years)	IUD	IUD (years)	STDs	STDs (number)	STDs:condylomatosis	STDs:cervical condylomatosis	STDs:vagina condylomatosi
Age	1.000000	0.086533	0.372709	0.562229	0.055570	0.225829	0.119631	0.087482	0.292857	0.260514	0.206688	0.006242	-0.018520	-0.029120	nan	0.00654
Number of sexual partners	0.086533	1.000000	-0.150872	0.098446	0.233542	0.157667	0.164829	0.010872	0.027334	0.042537	0.008065	0.024478	0.003054	-0.013521	nan	-0.04848
First sexual intercourse	0.372709	-0.150872	1.000000	-0.070213	-0.095313	-0.031433	-0.026922	0.028351	0.003205	-0.046032	-0.042287	-0.007522	0.009743	0.024659	nan	0.07782
Num of pregnancies	0.562229	0.098446	-0.070213	1.000000	0.087312	0.202741	0.105120	0.155756	0.214918	0.219399	0.156386	0.034435	0.000200	-0.031127	nan	-0.00388
Smokes	0.055570	0.233542	-0.095313	0.087312	1.000000	0.719698	0.479872	0.010725	0.044432	-0.051074	-0.033530	0.124675	0.116616	0.068705	nan	0.07883
Smokes (years)	0.225829	0.157667	-0.031433	0.202741	0.719698	1.000000	0.719100	-0.013412	0.036247	0.039985	0.046639	0.082558	0.091425	0.049278	nan	0.12529
Smokes (packs/year)	0.119631	0.164829	-0.026922	0.105120	0.479872	0.719100	1.000000	0.001055	0.012486	0.014682	0.020436	0.025540	0.032321	0.012565	nan	0.04458
Hormonal Contraceptives	0.087482	0.010872	0.028351	0.155756	0.010725	-0.013412	0.001055	1.000000	0.457789	0.046750	-0.040411	-0.029968	-0.036626	-0.011170	nan	-0.06380
Hormonal Contraceptives (years)	0.292857	0.027334	0.003205	0.214918	0.044432	0.036247	0.012486	0.457789	1.000000	0.099699	-0.010854	-0.002528	-0.011114	0.002213	nan	-0.04254
IUD	0.260514	0.042537	-0.046032	0.219399	-0.051074	0.039985	0.014682	0.046750	0.099699	1.000000	0.745261	0.043232	0.047683	0.079724	nan	0.03385
IUD (years)	0.206688	0.008065	-0.042287	0.156386	-0.033530	0.046639	0.020436	-0.040411	-0.010854	0.745261	1.000000	0.007269	0.005202	0.018716	nan	-0.01086
STDs	0.006242	0.024478	-0.007522	0.034435	0.124675	0.082558	0.025540	-0.029968	-0.002528	0.043232	0.007269	1.000000	0.919104	0.737544	nan	0.23640
STDs (number)	-0.018520	0.003054	0.009743	0.000200	0.116616	0.091425	0.032321	-0.036626	-0.011114	0.047683	0.005202	0.919104	1.000000	0.901502	nan	0.36418
STDs:condylomatosis	-0.029120	-0.013521	0.024659	-0.031127	0.068705	0.049278	0.012565	-0.011170	0.002213	0.079724	0.018716	0.737544	0.901502	1.000000	nan	0.32052
STDs:cervical condylomatosis	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	na
STDs:vaginal condylomatosis	0.006542	-0.048487	0.077829	-0.003888	0.078830	0.125290	0.044585	-0.063809	-0.042545	0.033859	-0.010868	0.236400	0.364189	0.320523	nan	1.00000
STDs:vulvo-perineal condylomatosis	-0.026257	-0.011577	0.029963	-0.030068	0.072313	0.051873	0.014031	-0.016247	0.004540	0.062118	0.017401	0.726934	0.891135	0.985613	nan	0.23927
STDs:syphilis	0.009298	0.031690	-0.110380	0.111444	0.110728	0.024115	0.000558	-0.013834	-0.000952	-0.053900	-0.040170	0.461627	0.321246	0.051646	nan	-0.01176

The code for the above work can be found at:
https://github.com/s270231/ds_tesina_riskfactors/blob/master/1_data_analyzing.ipynb

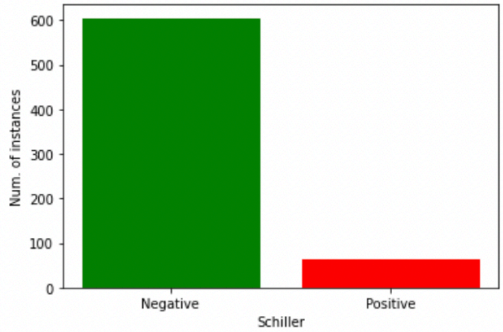
Dataset rebalancing

Due to the fact, that our dataset is very unbalanced, it is necessary to rebalance our dataset in order to improve the performance of the different algorithms and reduce the bias.

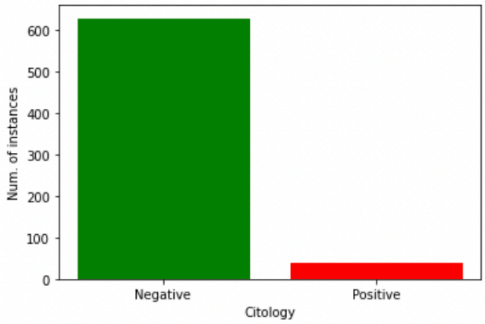
As an overview here are some examples of Number of instances and positive/negative samples for the main 4 examinations Hinselmann, Schiller, Citology and Biopsy:



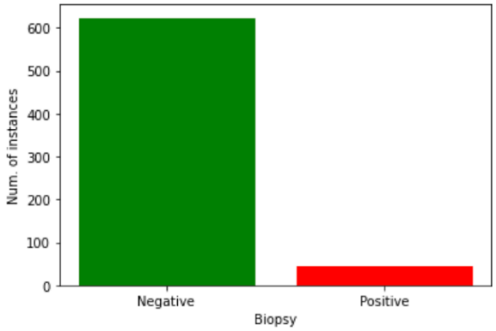
[638, 30]



[605, 63]



[629, 39]



[623, 45]

First, we split the entire dataset in training set and test set, assigning 70% of data points to the former and the remaining 30% to the latter, in a stratified way. This means that in the training and test sets, we have the same proportion of positive and negative samples as we had in the original dataset. Therefore, we trained the models using the training set and then validated the models on the test set.

After the splitting we obtain a training set composed of 467 samples and a test set 201 of samples. As a second step, we rebalanced only the training set, leaving intact the test set.

In our research we decided to use a technique **SMOTE**.

SMOTE

For example, the Random Oversampling technique balances the class distribution but does not increase the variety of the dataset. An improvement over the duplication of the examples from the minority class is about synthesizing new examples from the minority class that were not present in the original dataset. This is a type of data augmentation for tabular data, and it can be very effective. The most widely used approach to synthesizing new examples is called **Synthetic Minority Oversampling Technique (SMOTE).**

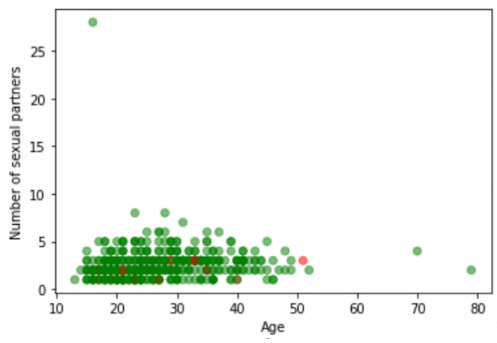
SMOTE works by utilizing a k-nearest neighbor algorithm to create synthetic data. First, it starts by choosing random data from the minority class, then the k-nearest neighbors from the data are selected, where **k** is an algorithm's parameter that was set to 5. Therefore, a randomly selected neighbor is chosen, and a synthetic example is created at a randomly selected point between the two examples in feature space. In other words, SMOTE first randomly selects a minority class instance **a** and finds its k nearest minority class neighbors. Among the k data points, one of them gets randomly selected and used to synthesize the new data point, we will refer to this point as **b**. The synthetic instance is thus created by connecting **a** and **b** to form a line segment in the feature space. In mathematical terms, the synthetic instances are generated as a convex combination of the two chosen instances **a** and **b**:

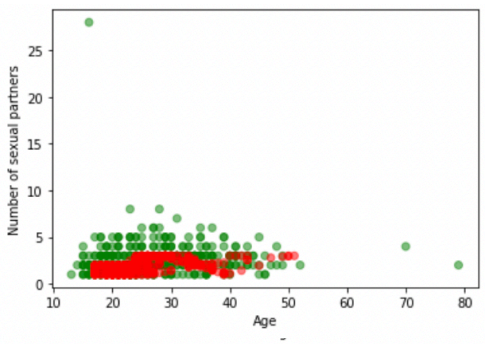
$$x_{new} = a + rand(0,1) * |a - b|$$

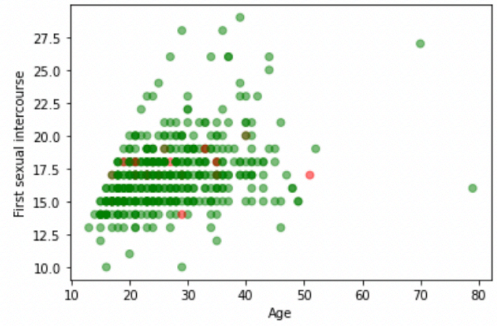
in which rand(0, 1) represents a real random number between 0 and 1.

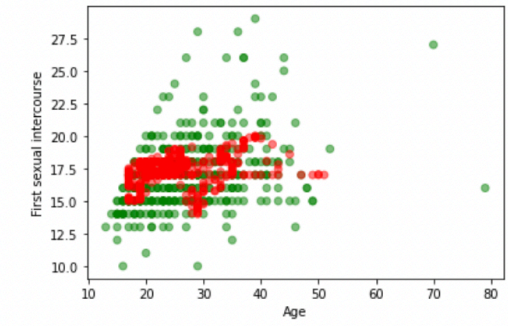
The procedure is repeated enough times until the minority class has the same proportion as the majority class.

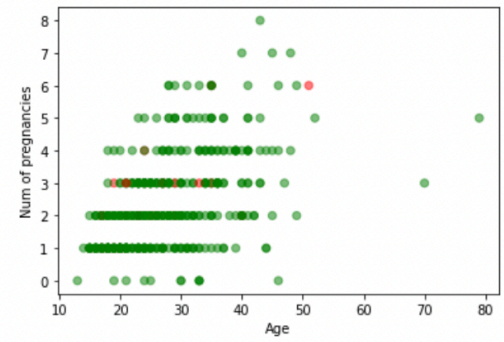
The examples of the dataset before (left) and after (right) SMOTE application are shown below (for Hinselmann):

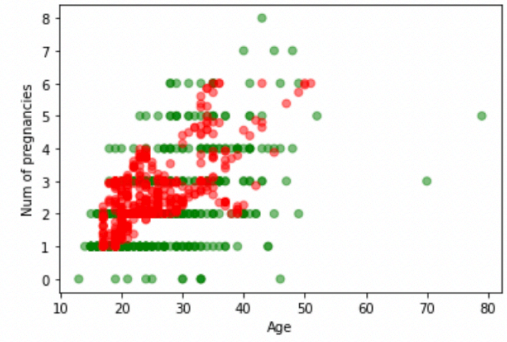

Number of sexual partners
Age

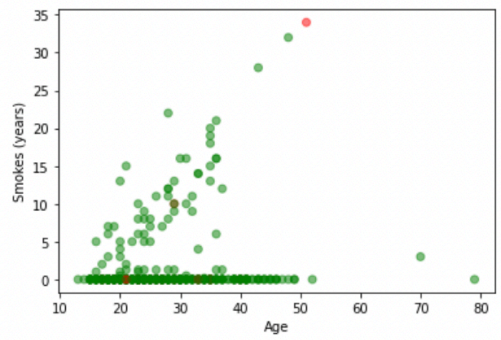

Number of sexual partners
Age

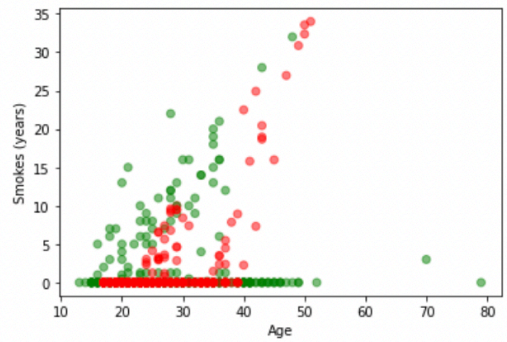

First sexual intercourse
Age

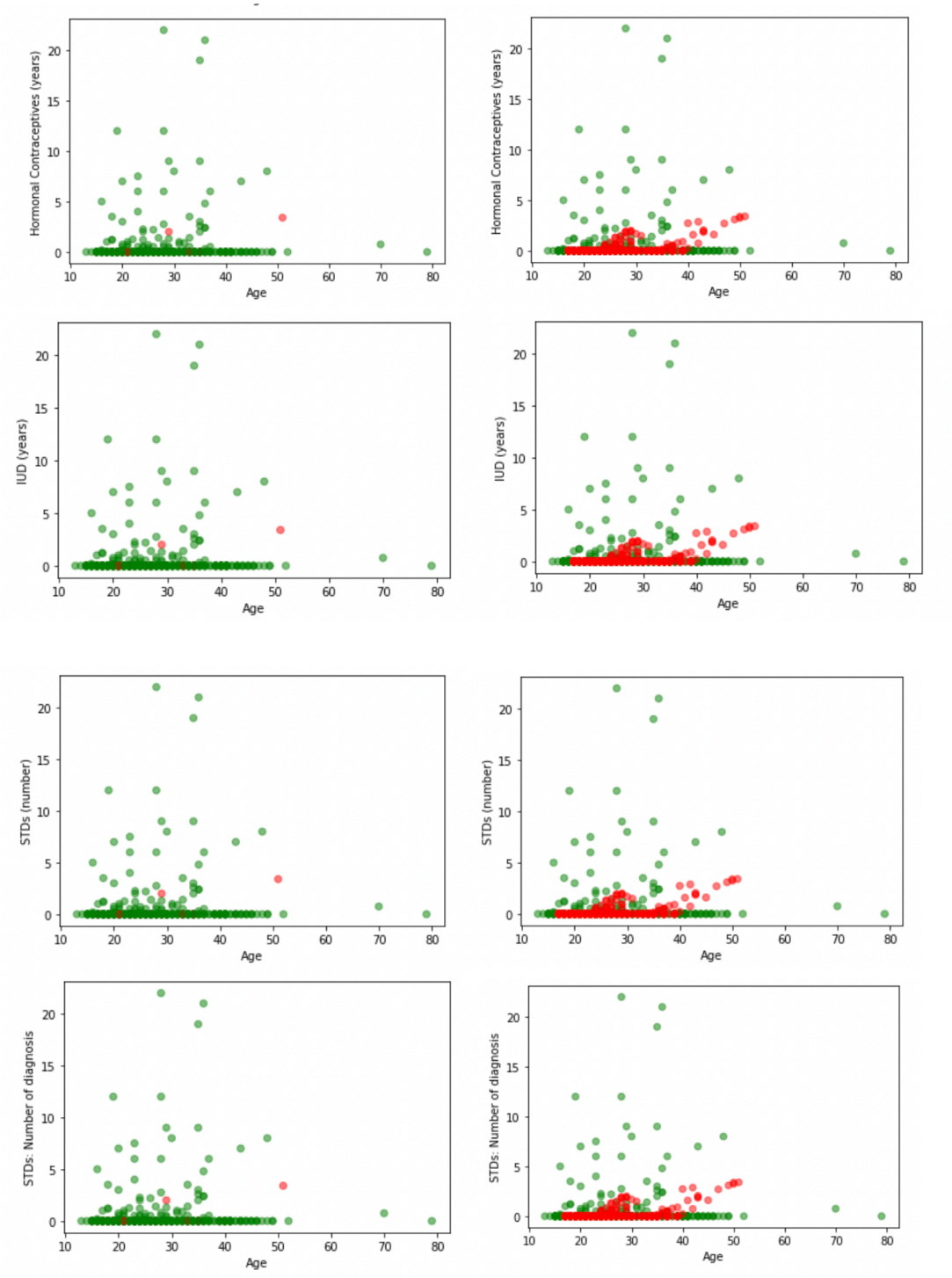

First sexual intercourse
Age


Num of pregnancies
Age


Num of pregnancies
Age

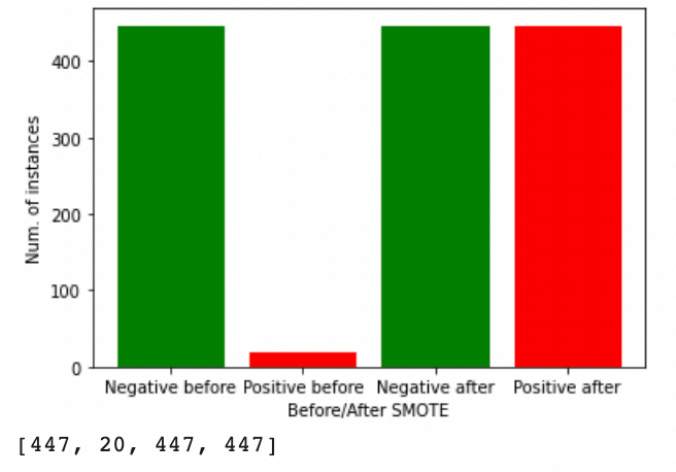

Smokes (years)
Age


Smokes (years)
Age



On the left there are plots of the data before SMOTE application, while on the right there are plots of the data after SMOTE application. Green points are those belonging to the negative class, while red points are those belonging to the positive class (minority class). As we can see in the figures representing the oversampled categories, there are many new generated samples belonging to the positive class which were not present before.

As a result: the number of positive and negative samples of the obtained new balanced training set, are different than the number before applying this function and now both negative and positive values are balanced, this will allow our model to understand better when is the positive case.




```
[447, 20, 447, 447]
```

Finally, we applied SMOTE on the obtained PCA dataset. The choice to use SMOTE instead of SMOTE-NC was since, this time, we had continuous variables (as a clarification we use an example of Age category, which changes with different values, but not because it is floating).

The code for the above work can be found at:
https://github.com/s270231/ds_tesina_riskfactors/blob/master/2_data_splitting_and_oversampling.ipynb

Principal Component Analysis

Principal Component Analysis (PCA) is a technique commonly used for dimensionality reduction by projecting each data point onto a lower-dimensional space, through a linear mapping, while preserving as much of the data's information as possible. This compressing procedure is called **data encoding**, while the reverse procedure is called **data decoding**. So PCA produces as a solution a linear mapping such that the distance between the original data and the reconstruction (encoding and then decoding) of the data is as small as possible. In mathematical terms, it is expressed in the following way:

$$\arg\min_{U,W} \sum_{i=1}^{m} \|\boldsymbol{x_i} - UW\boldsymbol{x}_i\|_2^2$$

where $W \in R^{n,d}$ and $U \in R^{d,n}$ are the matrices used for encoding and decoding respectively. Given *(U, W)* as a solution of the previous problem, it was demonstrated that the column of *U* are orthonormal and $W = U^T$. The PCA Theorem states that, given a data matrix *X*, which contains as lines the samples $x^T_1, \ldots, x^T_m \in R^d$, and given a matrix $A = X^T X$, called **scatter matrix**, the columns of *U* are the eigenvectors $u_1, \ldots, u_n$ associated to first *n* eigenvalues of *A*, ordered by magnitude ($\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq \lambda_{n+1} \geq \ldots \geq \lambda_p$).

The quantity $\lambda_{n+1} + \lambda_{n+2} + \ldots + \lambda_p$ is called **reconstruction error** and represents the distance between the original and the reconstructed point.

In terms of variance, the first principal component can be equivalently defined as a vector lying on the direction that maximizes the variance of the projected data. In other words, given a set of features X_1, X_2, ..., X_p, the first principal component is the normalized linear combination of the features:

$$PC_1 = \alpha_{11}X_1 + \alpha_{21}X_2 + \ldots + \alpha_{d1}X_d = \sum_{i=1}^{d} \alpha_{i1}X_i \qquad with \sum_i a_{i1}^{\ 2} = 1$$

and has the direction of largest variance on the feature space. After the first principal component PC_1 has been determined, we can find the second principal component PC_2 which is itself a normalized linear combination of the p features and has maximal variance among all linear combinations that are uncorrelated with PC_1.

It turns out that constraining PC_2 to be uncorrelated with PC_1 is equivalent to constraining the directions of the two principal components to be orthogonal. We can continue in this way until we find the *p*-th principal component.

In mathematical terms, the problem of PCA in term of variance can be formulated as:

$$PC_i = \max_{\|PC\|=1} Var(PC) \quad | \quad \langle PC, PC_j \rangle = 0 \ \ \forall\, j < i$$

So, in general, we can say that the *i*-th principal component can be taken as a direction orthogonal to the first *(i-1)*-th principal components that maximizes the variance of the projected data.

In order to estimate the variance explained by each component and to understand the importance of each component we must compute the **PVE (Proportional Variance Explained)**.

Given a centered dataset *X* composed by *n* observations, the total variance is defined as:

$$\sum_{j=1}^{p} Var\left(X_j\right) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$$

and the variance explained by the *m*-th principal component is:

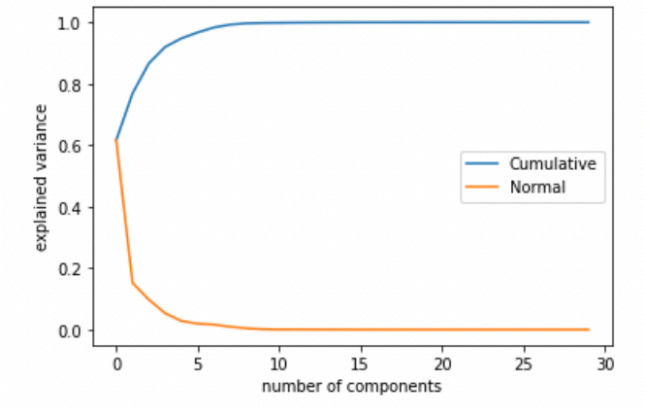
$$Var(PC_m) = \frac{1}{n} \sum_{i=1}^{n} z_{im}^2$$

where z_{im} are the component of PC_m. Therefore, the **PVE** of the *m*-th principal component is given by the positive quantity between 0 and 1:

$$PVE_m = \frac{Var(PC_m)}{\sum_{j=1}^{p} Var(X_j)} = \frac{\lambda_m}{\sum_{i=1}^{p} \lambda_i}$$

The cumulative PVE is simply the incremental sum of PVE of different principal components.

After plotting the values of cumulative and normal explained variance, we decided to take the first 9 principal components (values) that explain a cumulative variance of almost 90% (Cumulative explained variance for 9 principal components: 0.9966405340853833), as shown by following figure:



We have noticed that after 9 components the results are almost the same.

The code for the above work can be found at:
https://github.com/s270231/ds_tesina_riskfactors/blob/master/3_principal_component_analysis.ipynb

Classification

Random Forest

Random forest is a type of ensemble method of machine learning. It deals with overfitting problems and increases accuracy compared to a simple decision tree model. This method grows multiple trees, on bootstrapped training samples, which are then combined to yield a single consensus prediction. Indeed, combining a large number of trees can often result in dramatic improvements in prediction accuracy. However, one of the cons of random forest is the fact that it is less interpretable than a simple decision tree.

When building the decision trees, each time a split in a tree is considered, a random selection of *m* predictors is chosen as split candidates from the full set of *p* predictors. The split is allowed to

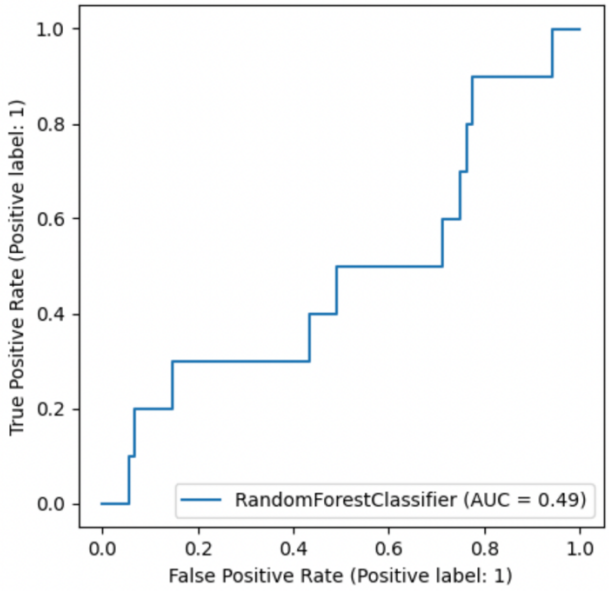
use only one of those m predictors. So, while in Bagging we use a number *m* which is equal to the total number *p* of predictors, in the case of Random Forest we use a random selection of *m* < *p* predictors. This choice of m predictors is repeated for each bootstrapped set. This means that, in the case of Random Forest, we do a bagging on the dataset, but also a feature bagging, drawing random subsets of the predictors. Good practice is to use $m \approx p$ predictors. This trick **decorrelates** the multiple trees and so helps to reduce the variance when we average them.

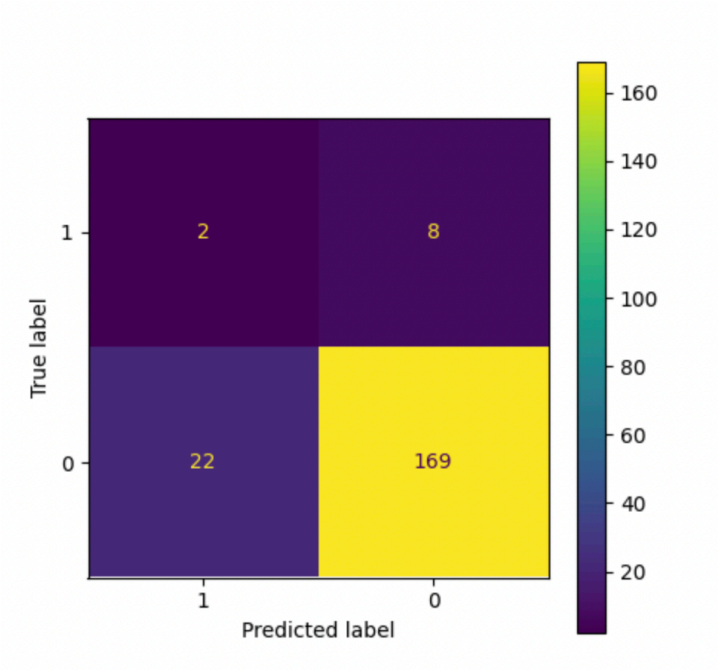
For classification tasks (which is our goal), the output of the random forest is the class selected by most trees.

Among the most important hyperparameters of the random forest we have the **number of trees** that it trains, the **number of features** to consider when looking for the best split, and other hyperparameters that we have already seen in the case of decision tree, such as the **criterion** used for the splitting (Gini index or Entropy), the **max depth** and the **minimum number of samples** required to split an internal node.

Also in this case, we found the best hyperparameters using grid search cross validation. From the obtained results we noticed that gini index outperforms entropy in all cases except in the case of PCA dataset.

In the case of **accuracy,** the obtained results are shown below:





Decision tree

Our next step into classification was made using decision trees, since thanks to their interpretability they offer an insight on the classification task and let us dive into separability concerns. Despite their lower accuracy when compared to other classifiers, we considered it important since some hyperparameters tuned in this task were also used for the random forest algorithm.

As discussed in the previous section, we have already divided our dataset in train and test splits, for each of the different flavors of the dataset: retaining outliers, relieving outliers, using data augmentation (RO / SMOTE), and using PCA.

A **decision tree** is the composition of all the possible solutions generated splitting the datasets into two or more homogeneous sets. The operation of splitting into two or more homogenous sets is repeated in a recursive fashion until all the leaves of the generated tree are totally homogenous or another stop condition is met (for example, if the maximum number of levels is reached).

We have different formulas to determine whether a split is homogenous or not, in our work we examined the two most used: **Gini Index** and **Entropy**.

The **Gini Index** is calculated by subtracting the sum of the squared probabilities of each class from one, and it can be expressed with the following formula:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Where p_{mk} indicates the proportion of training samples in the **m-th** region that are from the **k-th** class, while ***K*** is the number of classes.

On the other hand, **Entropy** is a concept taken from information theory, where it is defined as the average level of "information" in all possible outcomes of a variable.

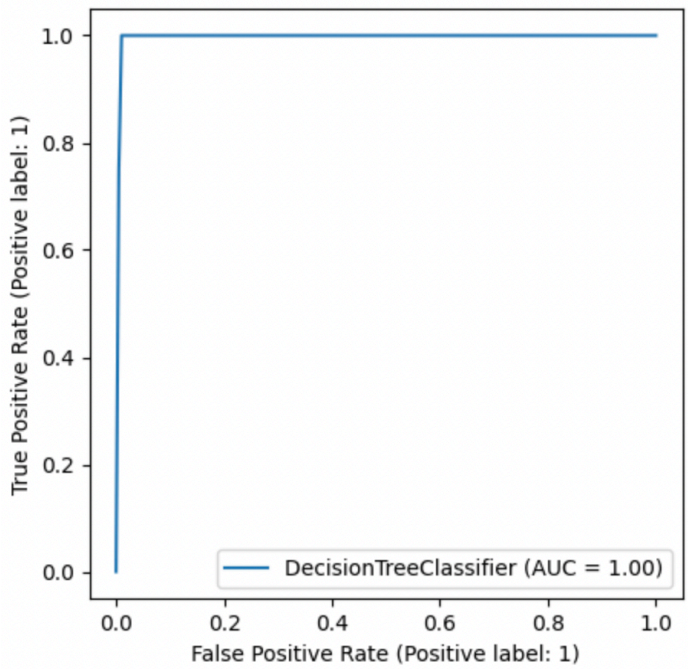
The entropy can be expressed as:

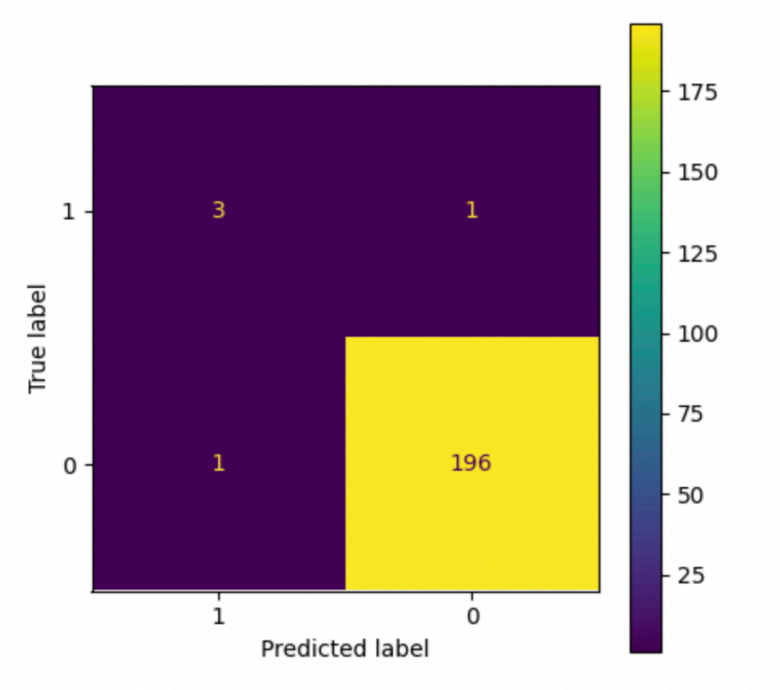
$$G = \sum_{k=1}^{K} -\hat{p}_{mk} log(\hat{p}_{mk})$$

Both Gini Index and Entropy are measures that indicate how heterogenous the set is, since we want our splits to be as homogenous as possible, we calculate the measures both before and after the split and we only select the split on the attributes and on the threshold that minimize the measure after the split (i.e. that maximize the gain, which is calculated as the difference between the information before the split and the information after the split).

We performed a **grid search cross validation** in order to find the best criterion between Gini Index and Entropy, altogether with other hyperparameters such as the maximum levels of the tree and the minimum number of samples to perform the split.

We used different grid search scoring metrics and different datasets, and in our experiments it turned out that the gini index performed better in almost all the cases. The best recall values were reached with the dataset with SMOTE oversampling, while it performed poorly on the other datasets.





Logistic Regression

The logistic regression model is used to model the probability that the response variable Y belongs to a particular category. In order to model this probability logistic regression uses the *logistic function*:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

The logistic function ensures that p(X) is always between 0 and 1 and will always produce an S-shaped curve, and so regardless of the value of X, we will obtain a sensible prediction.

Our problem consists in predicting a binary response using multiple predictors. So we use the following *logistic function*:

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

Where X_1, $X_2, \ldots, X_p$, are the features and β_1, $\beta_2, \ldots, \beta_p$ are the weights to be learned.

Thus, at validation time, given a set of features X_1, $X_{2, \dots,} X_p$ and a set of weights β_1, $\beta_{2, \dots,}\beta_p$, our logistic function will return the probability that the sample belongs to the positive class. Logistic regression in this setting can be used only for binary classification, which is our case.

We used the variable *Hinselmann=True* as positive class, and *Hinselmann =False* as negative class.

To estimate the β parameters we can use different approach, one of them is the **Maximum Likelihood Estimation** (MLE), which is a technique consisting in maximizing the following likelihood estimation:

$$\ell(\beta_0, \beta) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i)).$$

This function consists of the product of the probabilities that our data are labeled with 1 ($y_i = 1$) time the product of the probability that they are labeled with 0 ($y_i = 0$).

However, this formulation was not used because it was not included in the library that we used. Instead, we defined a cost function and tested different solvers to find the minimum of this function.

Given **β** = [β_1, $\beta_{2, \dots,}$ β_p] the vector containing all the weights, $\boldsymbol{X_i} = [x_{i1}, x_{i2}, \dots, x_{ip}]$ a sample containing *p* features, and y_i the target for the i-th sample, the cost function is defined as:

$$\sum_{i=1}^{N} ln(e^{-y_i X_i^T \beta} + 1)$$

And, eventually, two penalizations can be added, **l1** (lasso regression) and **l2** (ridge regression):

$$||\beta|| + C \sum_{i=1}^{N} ln(e^{-y_i X_i^T \beta} + 1)$$

$$\frac{1}{2}||\beta^T \beta|| + C \sum_{i=1}^{N} ln(e^{-y_i X_i^T \beta} + 1)$$

Where **C** is a parameter that weighs the impact of the penalization and will be found through cross validation.

To minimize those cost functions, we explored different solvers, and they are:

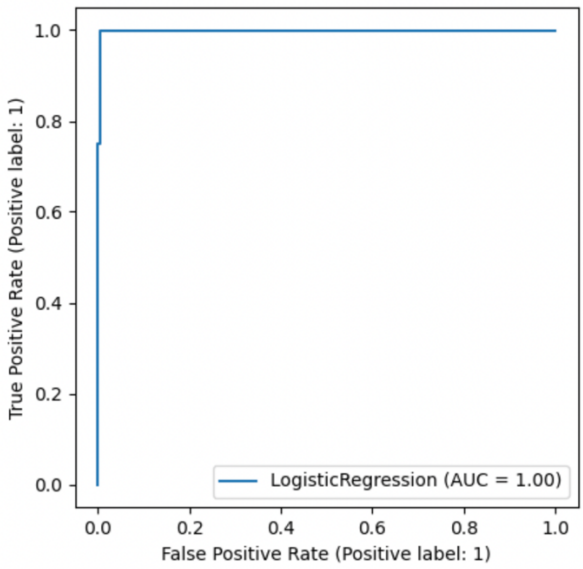
- **Netwon method**, which approximates the cost function locally to a quadratic function, and then takes a step towards the maximum/minimum of that quadratic function
- **Broyden–Fletcher–Goldfarb–Shanno Algorithm**, which works similarly to Newtonmethod but approximates the derivative to estimate the quadratic function

- **liblinear**, which uses a Coordinate Descent algorithm to solve the optimization problem, by successively performing approximate minimization along coordinate directions or coordinate hyperplanes
- **SAG** and **SAGA**, which approximates the cost function surface to a finite number of smooth convex functions, then uses stochastic gradient descent to find the minimum

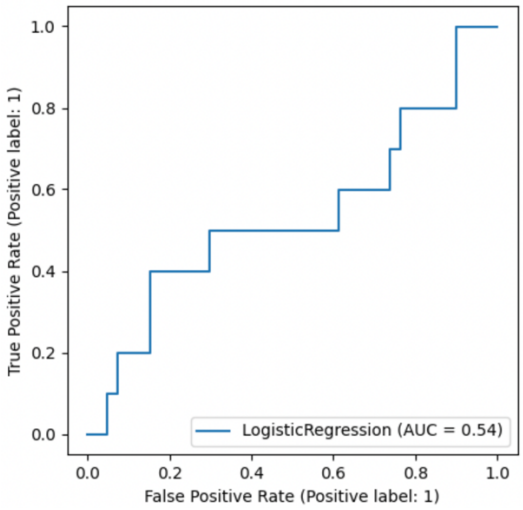
The penalization and the parameter **C** have been object of research through cross validation, while the solver has been manually examined and chosen by heuristics.

Once again, we executed the cross validation on all the datasets and used both accuracy and recall as parameters to find the best hyperparameters.

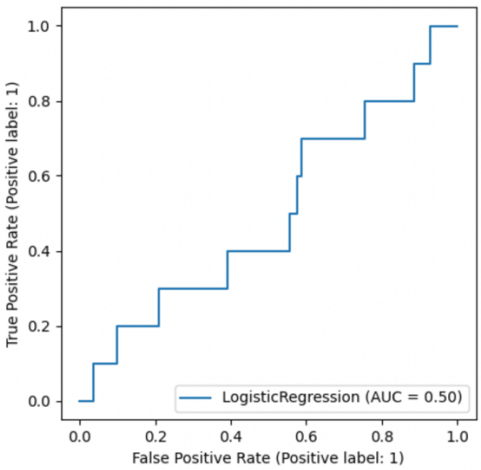
Accuracy logistic regression with no oversampling is shown here:



The accuracy logistic regression based on PCA roc is shown below:

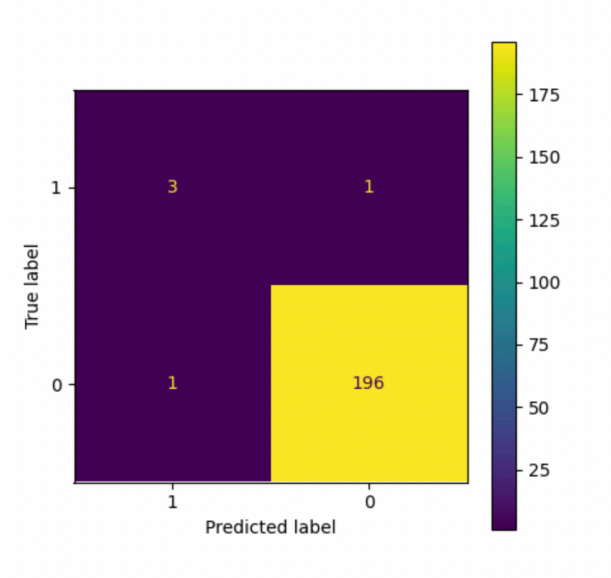


Recall of logistic regression is this one:

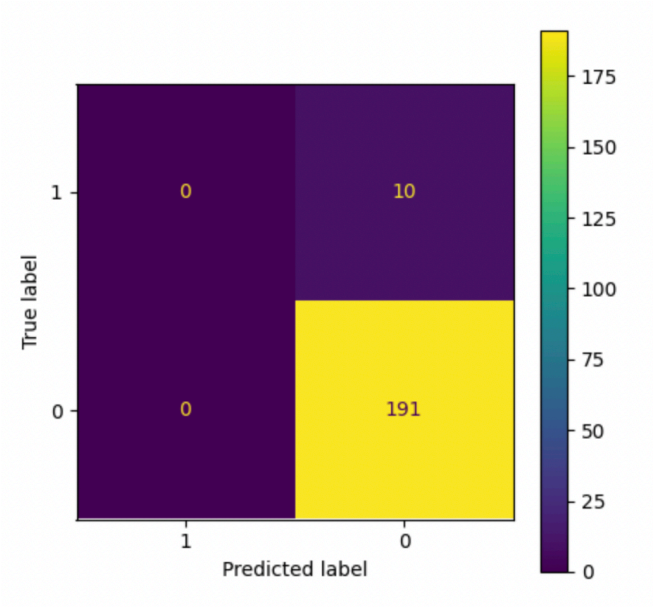


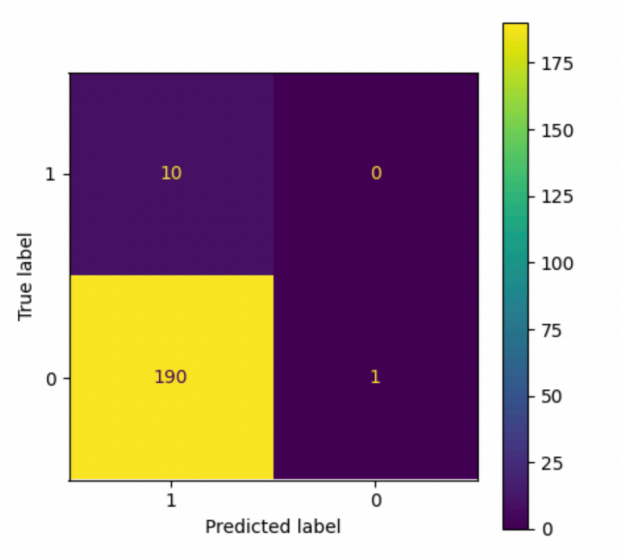
All the datasets reached an accuracy greater or eql than 50%, even if the unbalanced ones may be misleading.

The recall result is:



The accuracy result is:





Support Vector Machine

We decided to use support vector machine since it can achieve good performance also in high dimensional datasets and the use of kernels prevents us from making assumptions on the distribution of the data and it is thus versatile.

The aim of support vector machines is to find a hyperplane which is able to separate two classes. Differently to other linear classifiers, like perceptron, the support vector machines will always find an optimal hyperplane which maximizes the margin, i.e. the minimal distance between the hyperplane and the closest points of the two classes.

The support vector machine classifier can be formalized in the following formula:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M \quad \text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \;\; \forall\, i = 1, \ldots, n$$

Where **M** is the margin of the hyperplane as defined above, $\mathbf{x_{ij}}$ represents the points of the dataset, and $\boldsymbol{\beta_i}$ represents the weights to multiply the points and they are the values that we mean to find to optimize the problem. In other words, the optimization problem chooses $\boldsymbol{\beta_0}$, $\boldsymbol{\beta_1}$, **...,** $\boldsymbol{\beta_p}$ to maximize **M**.

The formula above is valid in the case a separating hyperplane exists, and it's named "hard margin definition". Though, there are cases where a separating hyperplane does not exist, and we thus introduce the concept of "soft margin definition" in order to adapt the SVM problem:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M \quad \text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$
$$\epsilon_i \geq 0, \;\; \sum_{i=1}^{n} \epsilon_i \leq C,$$

This formulation is similar to the previous one, but we added the ε vectors which represents the slack variables in the optimization problem, and **C** is a non-negative parameter that is inversely proportional to how "soft" our support vector machine can be. For values of **C** that tend to infinity, the formulation gets closer to the hard margin one.

The parameter **C** has been the object of tuning in the grid search for this classifier.

There are cases where a soft margin formulation is not enough to get a high accuracy, due to the non-linear nature of the data. The kernel trick comes to our aid since it lets us project the data in a non-linear space just replacing the inner product to a kernel function.

The kernel trick can be applied just replacing the inner product each time it is needed in the SVM with a generalization of the inner product of the form:

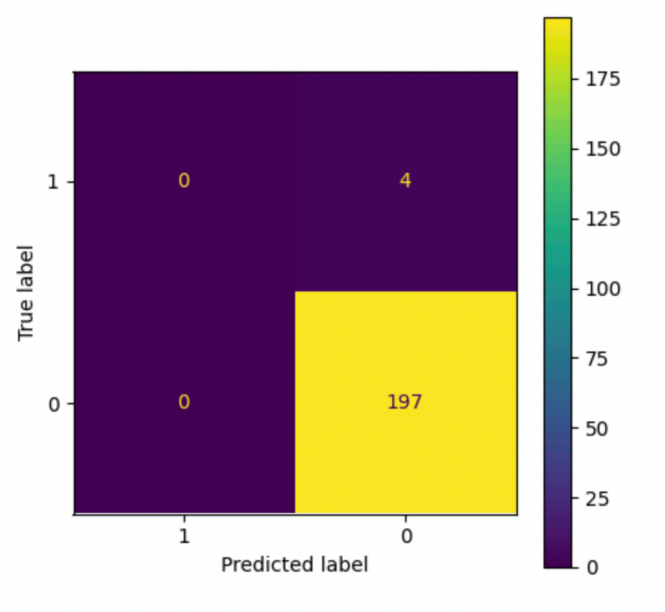
$$K(x_i, x_{i'}),$$

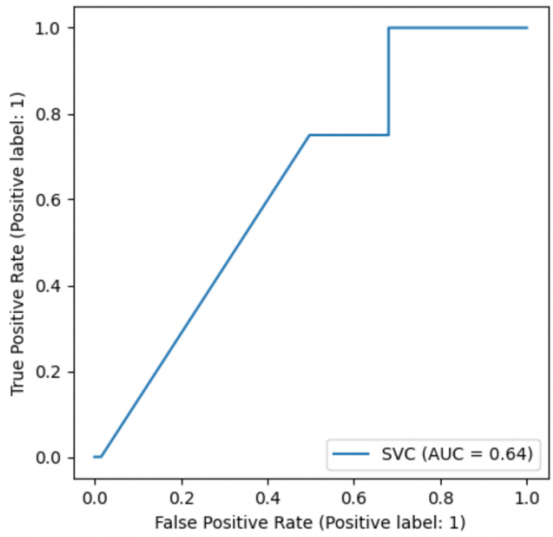
Where **K** is a special function that we call kernel.
We used the **RBF kernel**, which inner product can be expressed in this way:

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right)$$

In **SVC-RBF** is defined the gamma parameter as the inverse of the standard deviation of the RBF kernel, which is used as similarity measure between two points. A small gamma value defines a Gaussian function with a large variance. In this case, two points can be considered similar even if they are far from each other. On the other hand, a large gamma value means defining a Gaussian function with a small variance and in this case, two points are considered similar just if they are close to each other. Thus, the gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'.

Also the gamma parameter was tuned using grid search.





KNN

K Nearest Neighbors is a simple supervised learning algorithm that uses all available samples in order to classify new samples based on a similarity measure (e.g., distance functions).

Given a set of *m* training samples $\{x_i, y_i\}$ and a new point x that we want to classify, we compute the distance $d_i = d(x, x_i)$ *with i = 1, ..., m* between the new point x and all other points of the training set. After that, we order them based on their distance from x ($d_{i1} \geq d_{i2} \geq .... \geq d_{im}$) and take the first k nearest point. The label y* assigned to the new point x is the one that is the most frequent among the k closest points $x_{i1}, ..., x_{ik}$.

One of the most important parameters of KNN is the **distance metric**. It defines the similarity between 2 samples and can have a huge impact on the performance of the algorithm. The most used metric is Minkowski distance:

$$L_p(x, y) = \left(\sum_{k=1}^{m} \left(x_k - y_k \right)^p \right)^{1/p}$$

p is another important parameter of the algorithm. For p=2 Minkowski distance is also known as ***Euclidean distance***, while for p=1 we talk about ***Manhattan distance***.

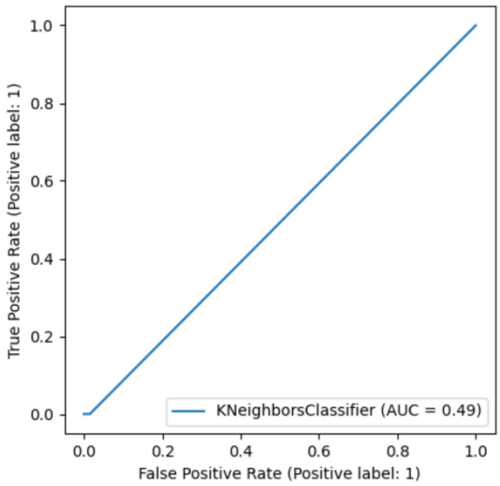
We took in consideration also the “weights” parameter that can assume two different values:

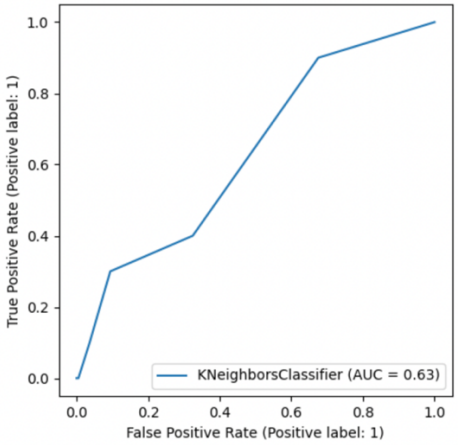
- ● *‘**uniform**’*: all points in each neighborhood are weighted equally,
- ● ‘***distance***’ : weight points by the inverse of their distance. The last important parameter that we took in consideration is the number of nearest neighbors **K**,

which is the core deciding factor. K is generally an odd number if the number of classes is 2. If K=1, then the sample is simply assigned to the class of its nearest neighbor. However, this is usually not a good choice since it can lead to overfitting.

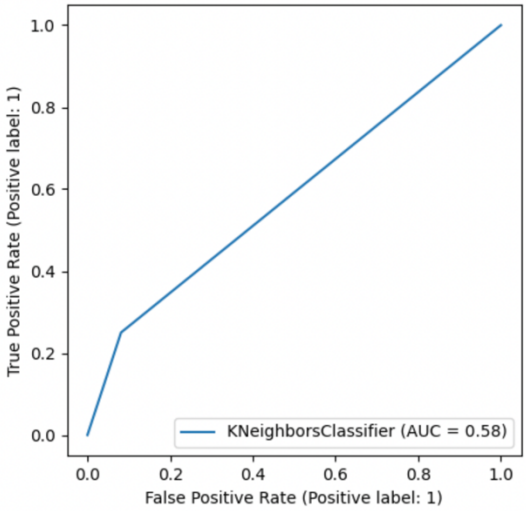
In order to find the best hyperparameters, we performed a grid search cross validation using different scoring metrics and datasets. As result we obtained that Manhattan distance outperforms the Euclidean one in all cases except in the case of dataset with outliers using the recall scoring, while the uniform weight generally outperforms the distance one, especially in the case of recall score.

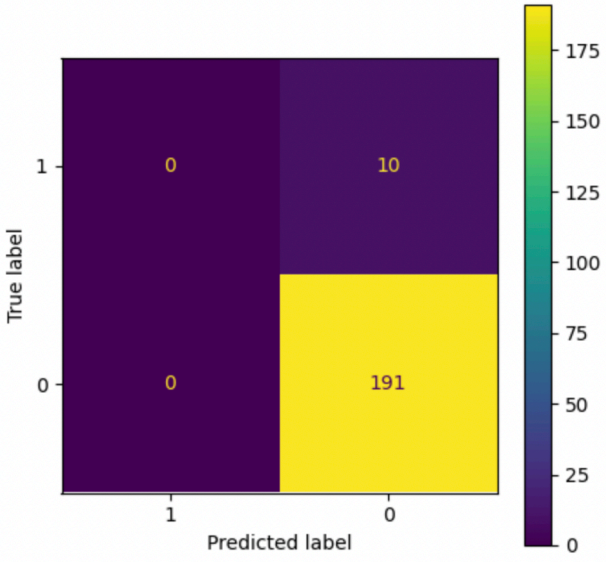
Recall with no oversampling:


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
KNeighborsClassifier (AUC = 0.49)

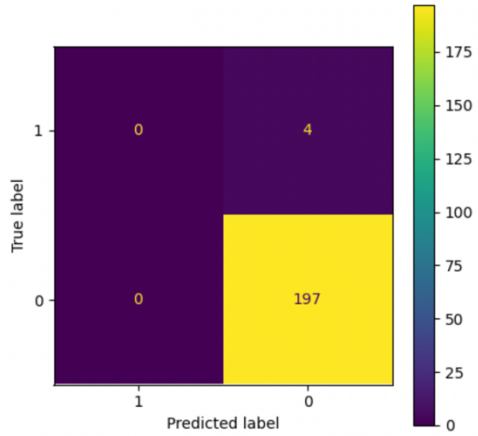

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
KNeighborsClassifier (AUC = 0.63)

Accuracy:


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
KNeighborsClassifier (AUC = 0.58)


True label
Predicted label
0
10
0
191

Accuracy no oversampling:



The code of all classifiers can be found at:
https://github.com/s270231/ds_tesina_riskfactors/blob/master/4_classifiers.ipynb

Result and conclusions

Our aim was to explore different classifiers and consider the impact of different manipulations applied to the initial dataset. The main difficulties in our work were linked to the scarce dimension of the dataset and the imbalance between the two classes. The dataset considering that many models relied only on a handful of them, we can assert that many features were not relevant to determine the target label.

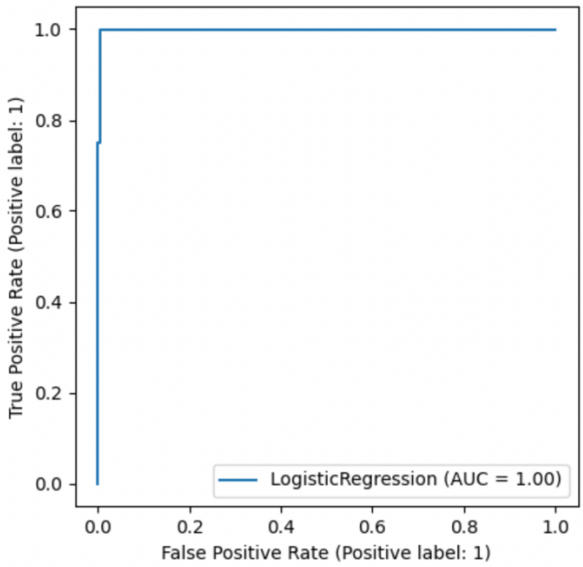
To sum up the score reached by all the classifiers over all the datasets, we report the following accuracy table:

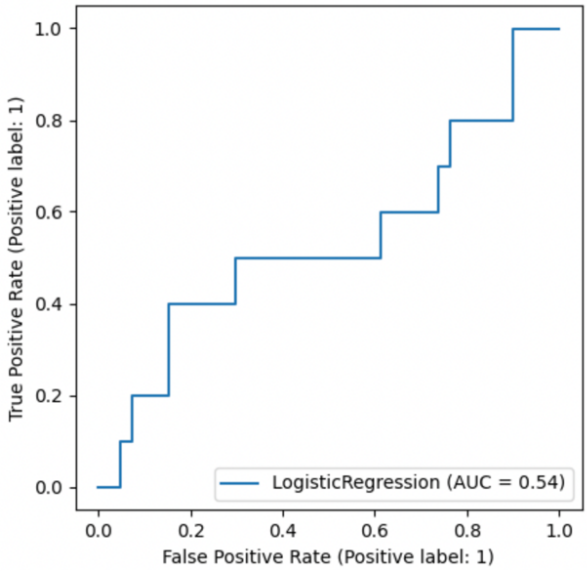
Accuracy	No oversampling	PCA
Logistic Regression	0.99004975	0.95024876
KNN	0.9800995	0.95024876
Desition Tree	0.99004975	0.94527363

Recall	No oversampling	PCA
Logistic Regression	0.75	1.
KNN	0.	0.
Desition Tree	0.75	0.

We saw that the best results are achieved with Random Forest model using dataset with PCA.

A slightly better result has been reached considering the PCA dataset:





The ROC curve represents the trade-off between the True Positive Rate and False Positive Rate across the different values of discrimination threshold.

Downstream our work, despite the results being mostly unsatisfactory, we also obtained different classifiers with a high score of accuracy and recall, that can be of some meaning for a in-field application.