University of Paris

Deep learning project

Wacim Belahcel and Imad Oualid Kacimi

27/01/2021

Table des matières

Introduction

As part of the "Deep Learning" course followed in the Machine Learning for Data Science Master 's Degree at the university of Paris, we have been asked to discuss and implement different neural network models on three different imbalanced datasets. In addition to deep learning algorithms, we will use different sampling methods in order to find the right balance between datasets classes and to improve initial results of our models.

In the following sections, we will discuss each dataset and the pre-processing performed on them, we will follow by explaining the different methods used for resampling and model used for training, to finally compare our different results and conclude.

Datasets

Bank marketing

Data description

This dataset is based on "Bank Marketing" UCI dataset, it contains data related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y which is our target variable).

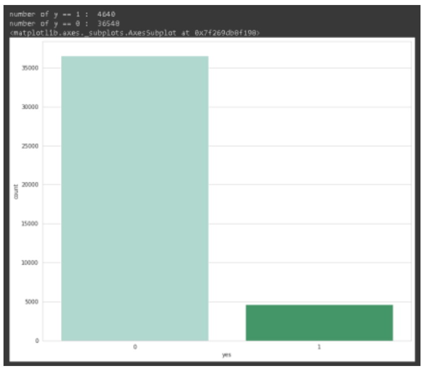
The dataset itself is composed of different variable that could be categorized in 4 main groups, we first have client data (age, job, education), with a total of 7 client related variables.

The dataset also contains information related to the last contact of a campaigns (communication type, month of last contact, duration of last contact, day of week).

We then have 5 socio economic variables followed by 4 "other" variables (number of day after last contact, number of contact performed, outcome of previous campaign…).

Data exploration

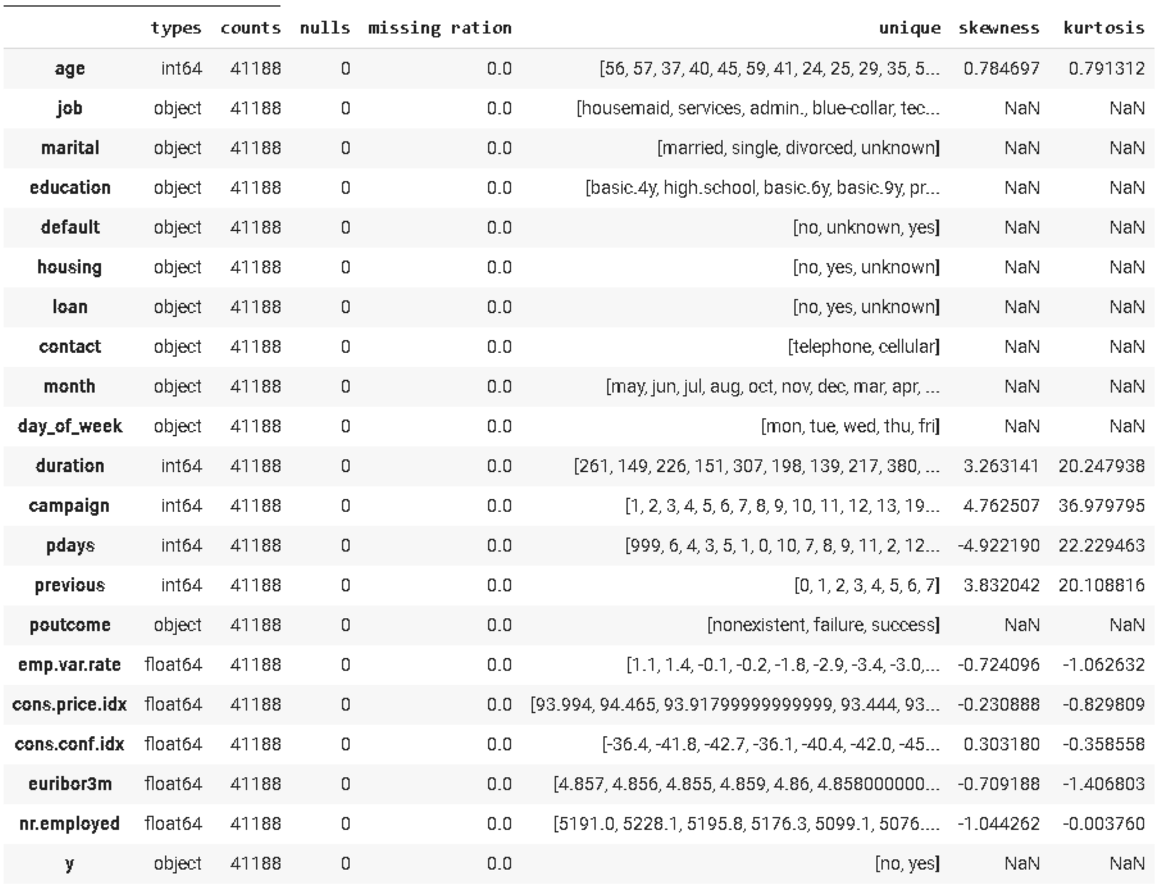
First thig we can notice from the figure below is that the dataset is highly imbalanced with our minority class only representing approximately 10% of the overall dataset



```
number of y == 1 :  4640
number of y == 0 :  36548
<matplotlib.axes._subplots.AxesSubplot at 0x7f269db0f190>
```

Figure 1 class distribution bank dataset

Moreover, if we look closely to our variable as shown in the table below, at first glance it seems that there are no missing values, but as mentioned in the original dataset description, missing values here are tagged as unknown, in our study we will consider them as a modality of their own.



	types	counts	nulls	missing ration	unique	skewness	kurtosis
age	int64	41188	0	0.0	[56, 57, 37, 40, 45, 59, 41, 24, 25, 29, 35, 5...	0.784697	0.791312
job	object	41188	0	0.0	[housemaid, services, admin., blue-collar, tec...	NaN	NaN
marital	object	41188	0	0.0	[married, single, divorced, unknown]	NaN	NaN
education	object	41188	0	0.0	[basic.4y, high.school, basic.6y, basic.9y, pr...	NaN	NaN
default	object	41188	0	0.0	[no, unknown, yes]	NaN	NaN
housing	object	41188	0	0.0	[no, yes, unknown]	NaN	NaN
loan	object	41188	0	0.0	[no, yes, unknown]	NaN	NaN
contact	object	41188	0	0.0	[telephone, cellular]	NaN	NaN
month	object	41188	0	0.0	[may, jun, jul, aug, oct, nov, dec, mar, apr, ...	NaN	NaN
day_of_week	object	41188	0	0.0	[mon, tue, wed, thu, fri]	NaN	NaN
duration	int64	41188	0	0.0	[261, 149, 226, 151, 307, 198, 139, 217, 380, ...	3.263141	20.247938
campaign	int64	41188	0	0.0	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 19...	4.762507	36.979795
pdays	int64	41188	0	0.0	[999, 6, 4, 3, 5, 1, 0, 10, 7, 8, 9, 11, 2, 12...	-4.922190	22.229463
previous	int64	41188	0	0.0	[0, 1, 2, 3, 4, 5, 6, 7]	3.832042	20.108816
poutcome	object	41188	0	0.0	[nonexistent, failure, success]	NaN	NaN
emp.var.rate	float64	41188	0	0.0	[1.1, 1.4, -0.1, -0.2, -1.8, -2.9, -3.4, -3.0,...	-0.724096	-1.062632
cons.price.idx	float64	41188	0	0.0	[93.994, 94.465, 93.91799999999999, 93.444, 93...	-0.230888	-0.829809
cons.conf.idx	float64	41188	0	0.0	[-36.4, -41.8, -42.7, -36.1, -40.4, -42.0, -45...	0.303180	-0.358558
euribor3m	float64	41188	0	0.0	[4.857, 4.856, 4.855, 4.859, 4.86, 4.858000000...	-0.709188	-1.406803
nr.employed	float64	41188	0	0.0	[5191.0, 5228.1, 5195.8, 5176.3, 5099.1, 5076....	-1.044262	-0.003760
y	object	41188	0	0.0	[no, yes]	NaN	NaN

Figure 2 bank dataset attributs

Also looking at skewness of our variables, we can see that some variables are also highly skewed like, which made us look closely at each variable, grouping them in two different types, categorical and numerical variables.

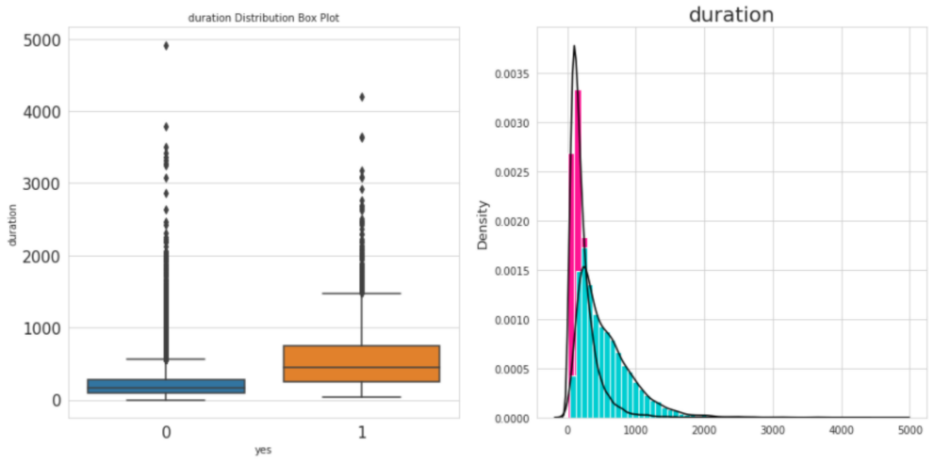


Figure 3 duration attributs speed

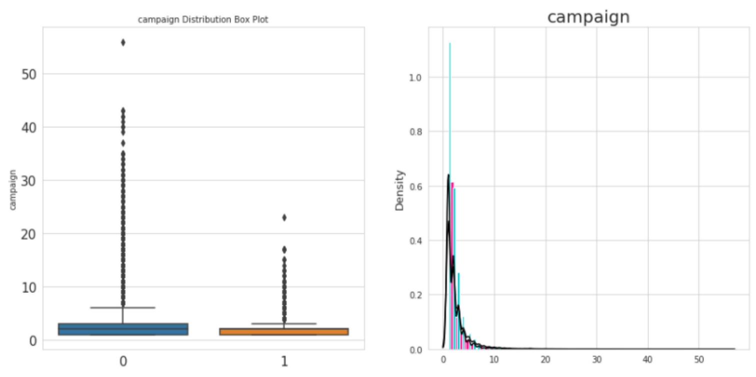


Figure 4 campaign attribut skewed

Starting with numerical variables, we can see on the box and distribution plot below two examples of numerical variables in our dataset using a boxplot and a distribution plot, separating our two class; we can first see from the distribution histogram, that some of our variables, much like “duration” and “campaign”, are highly skewed and extend over a wide range.

Nevertheless, we can clearly see that after some data preprocessing, a variable like “duration” will come handy to discriminate between our two classes, as we can clearly see two different distributions for our two classes.

Then using the box plot, we can also see that both classes contains a lot of outliers that will have to be removed.

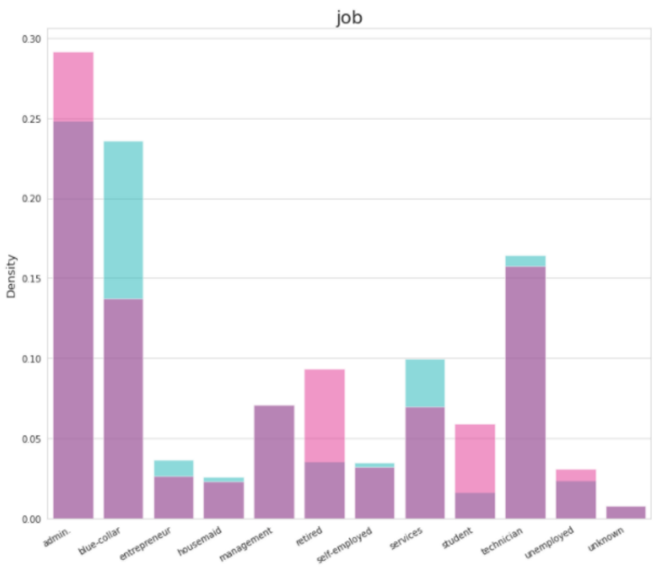


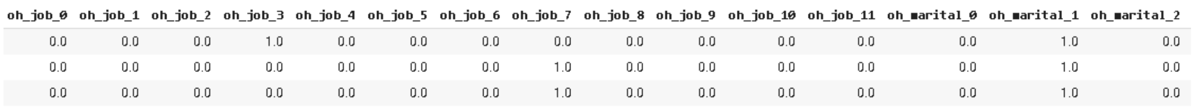
Figure 5 Job modalities distribution

We also represented our categorical variables using a bar plot, a perfect example of how our categorical variable will affect our final result would can be seen in the figure above which represent the distribution of our two classes for each modality of our “job” categorical variables.

As we can see, some modality like “technician” or “management” does not affect the target variable as they are perfectly balanced for both classes, but if we take a look at a modality like “student”, “retired” or “blue-collar”, we can see that they highly influence the target variable and we should focus on those case to make our final predictions.

Data preparation

The data preparation for this dataset consisted on many different steps and was different depending of the variable type (categorical or numerical), first, looking at our categorical variables, as shown in the table below, we had to recode each categorical variables using one hot encoding, which was an efficient way to separate each modality into a different column to be able to keep only feature relevant to our prediction process, the variable "age" was also converted to a categorical variable using different modality for each range of ages.



oh_job_0	oh_job_1	oh_job_2	oh_job_3	oh_job_4	oh_job_5	oh_job_6	oh_job_7	oh_job_8	oh_job_9	oh_job_10	oh_job_11	oh_marital_0	oh_marital_1	oh_marital_2
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

Figure 6 one hot transformation

We then used a Chi2 dependency test to find which modality is correlated with our target "y" variable, with an alpha of 0.05, we will keep only variables that are dependent to our target, which safely removed some useless modality like the one discussed a in the previous section.

Next focusing on our continuous numerical variables, we saw in the previous section that we saw that there are many outliers that could be removed, we applied a simple outlier removal technique using Interquartile bounds to remove outliers from both classes on each of our variables.

We then saw that some variables "duration" and "campaign" were highly skewed and extended over a wide range, we used a log transform to reduce this effect which made those two variables less skewed, an example can be seen below with "duration".

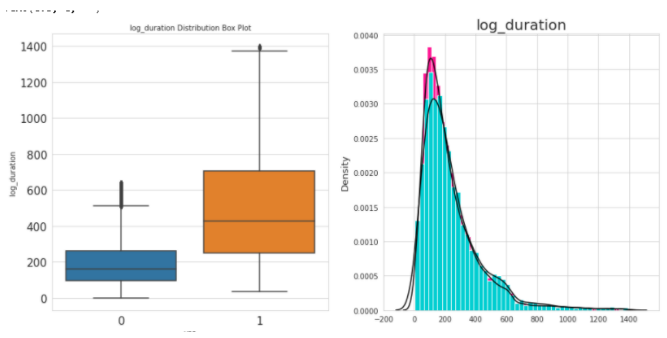


Figure 7 log normalized duration

We then standardized all our numerical variable to give them a mean of zero and a variance of one so that they each participate in prediction in an equitable fashion.

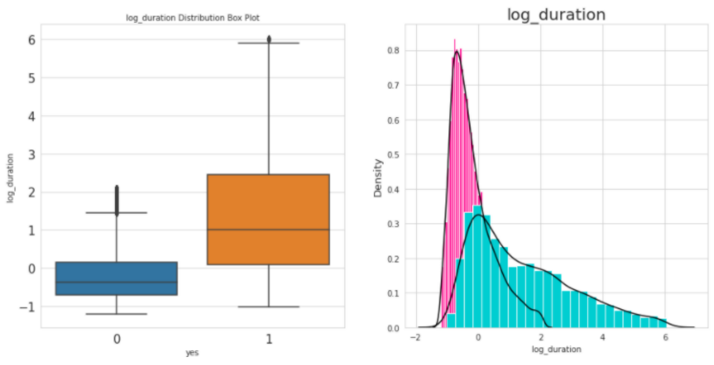


Figure 8 standardized log normalized duration

As we can see below, this process highly increased our numerical variables correlation to the target "y":

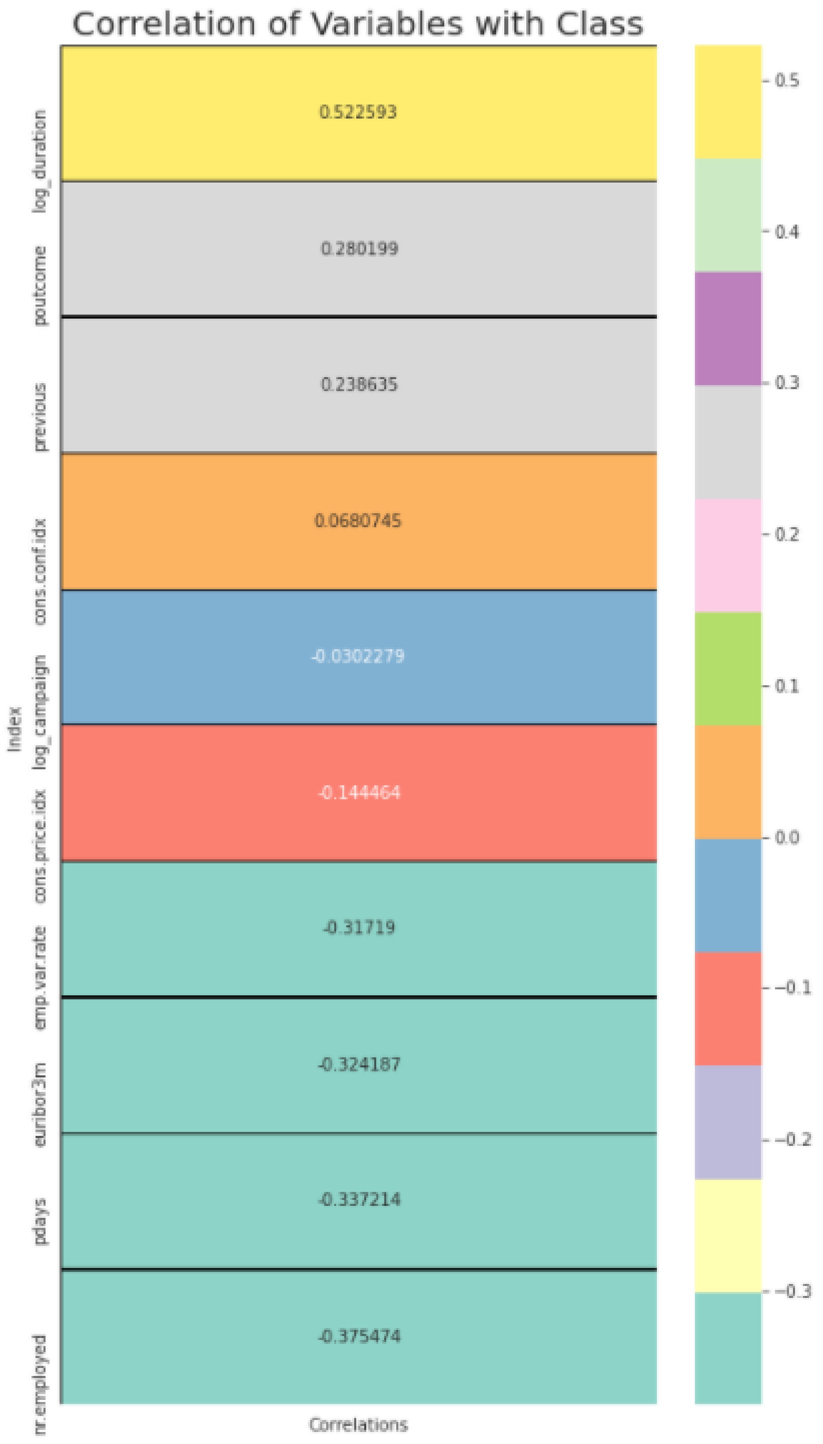


Figure 9 quantitative data correlation to target

Finally, the whole dataset was divided into three separate Test, validation and training set.

Fraud Dataset:

Data description

The initial dataset has been created with the collaboration of both Wordline and the machine learning group of ULB, with the primary goal to identify fraudulent credit card transactions.

The dataset is composed of transactions made by credit cards and that occurred in two days interval. This corpus contains only numerical variables, 28 of these features are obtained by the application of pca on the original variables and thus we do not have details about those variables. The other 2 features are TIME and AMOUNT.

TIME represent the seconds elapsed between each transaction and the first transaction in the dataset. AMOUNT is the transaction amount.

Each sample from the dataset is either a fraudulent or non-fraudulent example. The data is highly imbalanced and contains 492 frauds out of 284,807 transactions.

Data exploration

As we can see in Figure 11, the dataset is constituted of 284 807 samples with no missing values. We can also notice that the Amount feature is highly skewed since the median is very low compared to the mean, this can be due to the presence of outliers and can be handled using a log transform to reduce the skewness.

The V1 to V28 Feature are the result of PCA application and thus we can't rely on the statistics description since we do not know their origins.

The Figure 11 shows that the dataset is highly skewed and only 0.173% of the existing samples are fraudulent data.

```
number of non fraudulent data :  284315
number of fraudulent data :  492
<matplotlib.axes._subplots.AxesSubplot at 0x7f155e3db6a0>
```

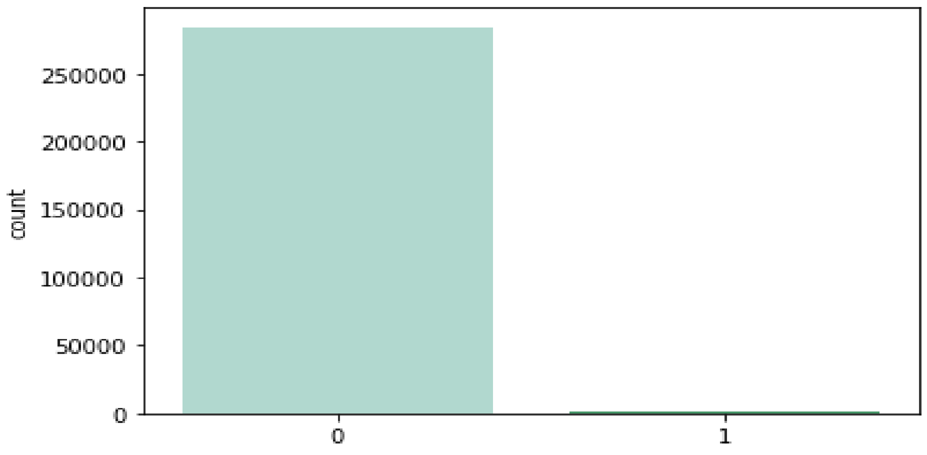
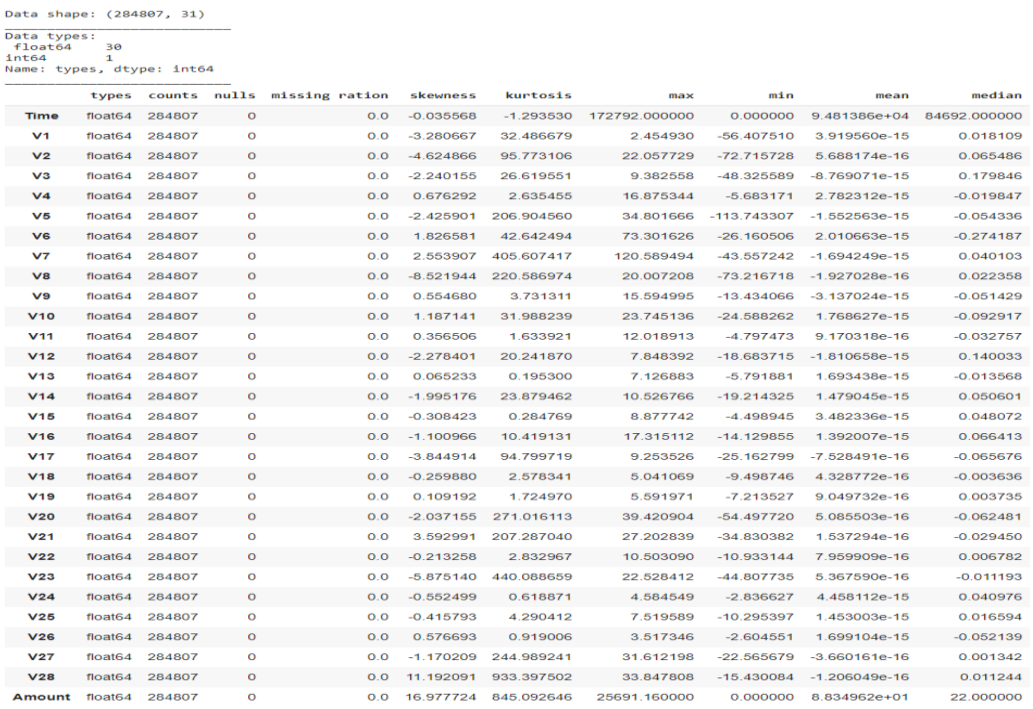


Figure 10 fraud and non-fraud transactions visualization



```
Data shape: (284807, 31)
______________________________
Data types:
 float64    30
int64       1
Name: types, dtype: int64
______________________________
          types  counts  nulls  missing ration  skewness   kurtosis          max         min          mean         median
Time    float64  284807      0             0.0 -0.035568  -1.293530 172792.000000    0.000000  9.481386e+04  84692.000000
V1      float64  284807      0             0.0 -3.280667  32.486679      2.454930  -56.407510  3.919560e-15      0.018109
V2      float64  284807      0             0.0 -4.624866  95.773106     22.057729  -72.715728  5.688174e-16      0.065486
V3      float64  284807      0             0.0 -2.240155  26.619551      9.382558  -48.325589 -8.769071e-15      0.179846
V4      float64  284807      0             0.0  0.676292   2.635455     16.875344   -5.683171  2.782312e-15     -0.019847
V5      float64  284807      0             0.0 -2.425901 206.904560     34.801666 -113.743307 -1.552563e-15     -0.054336
V6      float64  284807      0             0.0  1.826581  42.642494     73.301626  -26.160506  2.010663e-15     -0.274187
V7      float64  284807      0             0.0  2.553907 405.607417    120.589494  -43.557242 -1.694249e-15      0.040103
V8      float64  284807      0             0.0 -8.521944 220.586974     20.007208  -73.216718 -1.927028e-16      0.022358
V9      float64  284807      0             0.0  0.554680   3.731311     15.594995  -13.434066 -3.137024e-15     -0.051429
V10     float64  284807      0             0.0  1.187141  31.988239     23.745136  -24.588262  1.768627e-15     -0.092917
V11     float64  284807      0             0.0  0.356506   1.633921     12.018913   -4.797473  9.170318e-16     -0.032757
V12     float64  284807      0             0.0 -2.278401  20.241870      7.848392  -18.683715 -1.810658e-15      0.140033
V13     float64  284807      0             0.0  0.065233   0.195300      7.126883   -5.791881  1.693438e-15     -0.013568
V14     float64  284807      0             0.0 -1.995176  23.879462     10.526766  -19.214325  1.479045e-15      0.050601
V15     float64  284807      0             0.0 -0.308423   0.284769      8.877742   -4.498945  3.482336e-15      0.048072
V16     float64  284807      0             0.0 -1.100966  10.419131     17.315112  -14.129855  1.392007e-15      0.066413
V17     float64  284807      0             0.0 -3.844914  94.799719      9.253526  -25.162799 -7.528491e-16     -0.065676
V18     float64  284807      0             0.0 -0.259880   2.578341      5.041069   -9.498746  4.328772e-16     -0.003636
V19     float64  284807      0             0.0  0.109192   1.724970      5.591971   -7.213527  9.049732e-16      0.003735
V20     float64  284807      0             0.0 -2.037155 271.016113     39.420904  -54.497720  5.085503e-16     -0.062481
V21     float64  284807      0             0.0  3.592991 207.287040     27.202839  -34.830382  1.537294e-16     -0.029450
V22     float64  284807      0             0.0 -0.213258   2.832967     10.503090  -10.933144  7.959909e-16      0.006782
V23     float64  284807      0             0.0 -5.875140 440.088659     22.528412  -44.807735  5.367590e-16     -0.011193
V24     float64  284807      0             0.0 -0.552499   0.618871      4.584549   -2.836627  4.458112e-15      0.040976
V25     float64  284807      0             0.0 -0.415793   4.290412      7.519589  -10.295397  1.453003e-15      0.016594
V26     float64  284807      0             0.0  0.576693   0.919006      3.517346   -2.604551  1.699104e-15     -0.052139
V27     float64  284807      0             0.0 -1.170209 244.989241     31.612198  -22.565679 -3.660161e-16      0.001342
V28     float64  284807      0             0.0 11.192091 933.397502     33.847808  -15.430084 -1.206049e-16      0.011244
Amount  float64  284807      0             0.0 16.977724 845.092646  25691.160000    0.000000  8.834962e+01     22.000000
```

Figure 11 summary of the different features in the Fraud dataset

The figure 12 support the result found from the statistic description and we can clearly see that the Amount distribution is highly skewed, and some outliers can be visualized from the boxplot, However the Time features seems to be quite symmetric. Another thing that can be viewed from the distribution is that both time and Amount do not represent a separate distribution for both classes and thus let us make assumption that those features will not be helpful in disguising between fraudulent and no fraudulent classes.

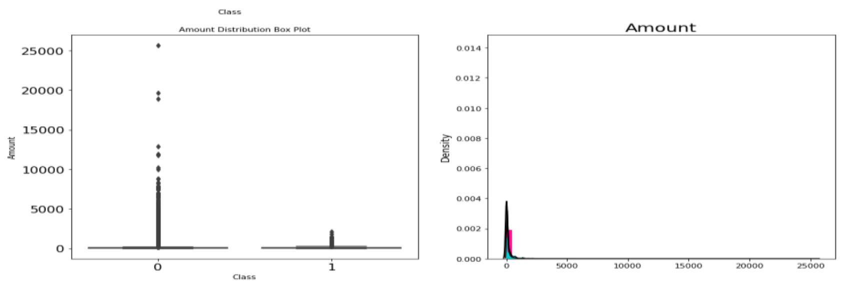


Figure 12: boxplot and distribution visualization for amount features

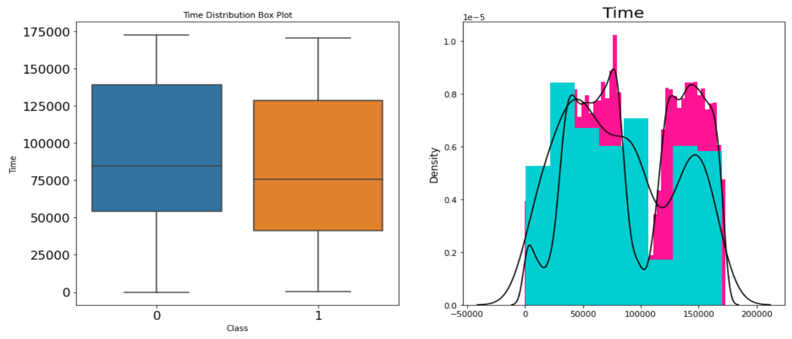


Figure 13: boxplot and distribution visualization for Time feature

Another interesting thing that has been explored, is visualizing the distribution of the pca features. The fig 5 shows some of the most relevant features that do display a separated distribution for the fraudulent and no fraudulent classes (V4, V11, V18, V14, V12), this can be helpful in the feature's selection part. (other visualization can be seen on the notebook

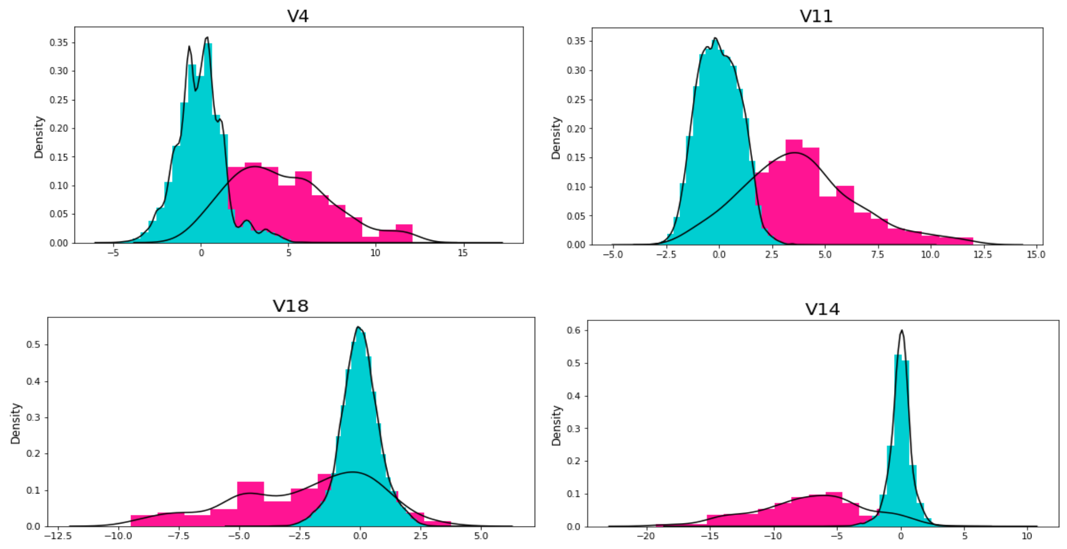
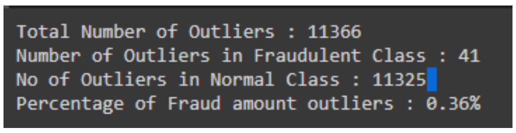


Figure 14: Most Relevant pca features

Data preparation

For the data preparation we started by removing outliers that has been located on the Amount feature. We used **an** Interquartile based method. However, as we are working with an imbalanced dataset, we use a high threshold of 5 to reduce the outlier removal from the fraudulent class.



```
Total Number of Outliers : 11366
Number of Outliers in Fraudulent Class : 41
No of Outliers in Normal Class : 11325
Percentage of Fraud amount outliers : 0.36%
```

Figure 15: Outlier removal

After removing outliers, we applied the log transform on both time and Amount to reduce the range of the values (variance), and to reduce the skewness.

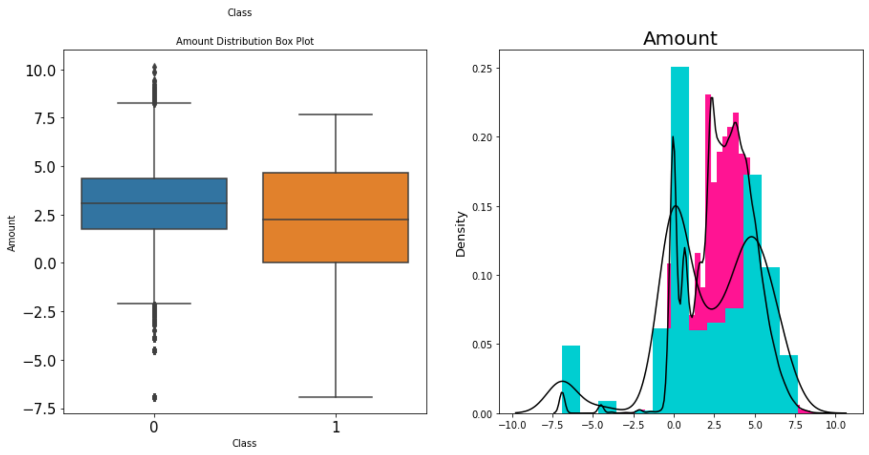


Figure 16: boxplot and distribution of Amount feature after Outlier removal +log transform

We then divided the dataset into train, validation and test set, applying the robust scaling on both amount and time, this scaling method has been chosen after several trials among different scaling method. We did not scale the PCA feature since they're already scaled.

Finally, we extracted the most relevant features by looking at features correlation to target and results are shown on the fig 17, we can see that correlation with target is low, this can indicate the presence of nonlinear relation rather than linear (since correlation is linear one). However, we can see that assumptions that has been made from the distribution visualization are true and dropped columns are those whom distribution are not separated for both classes.

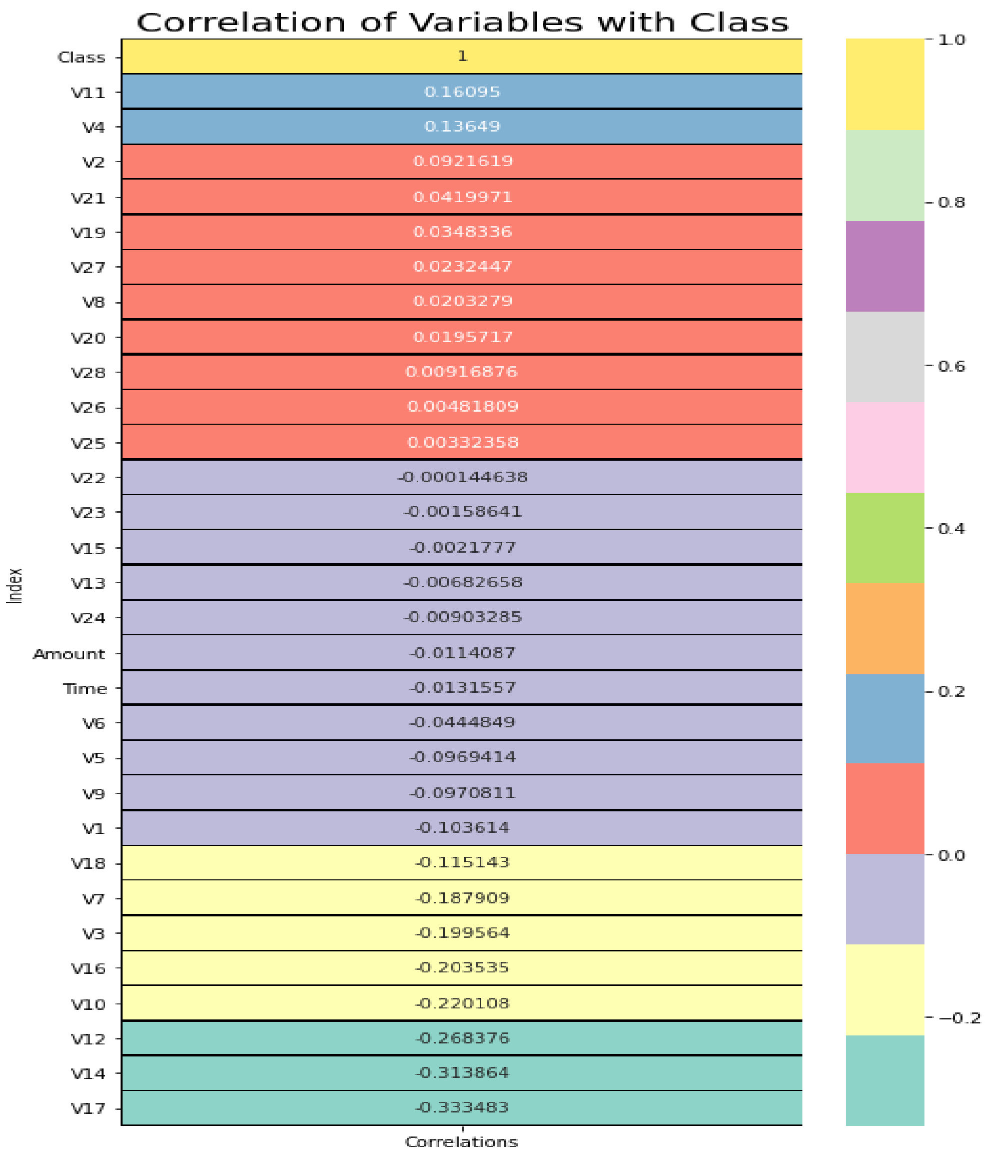


Figure 17: Correlation plot

```
['V21', 'V19', 'V27', 'V8', 'V20', 'V28', 'V26', 'V25', 'V22', 'V23', 'V15', 'V13', 'V24', 'Amount', 'Time', 'V6']
```

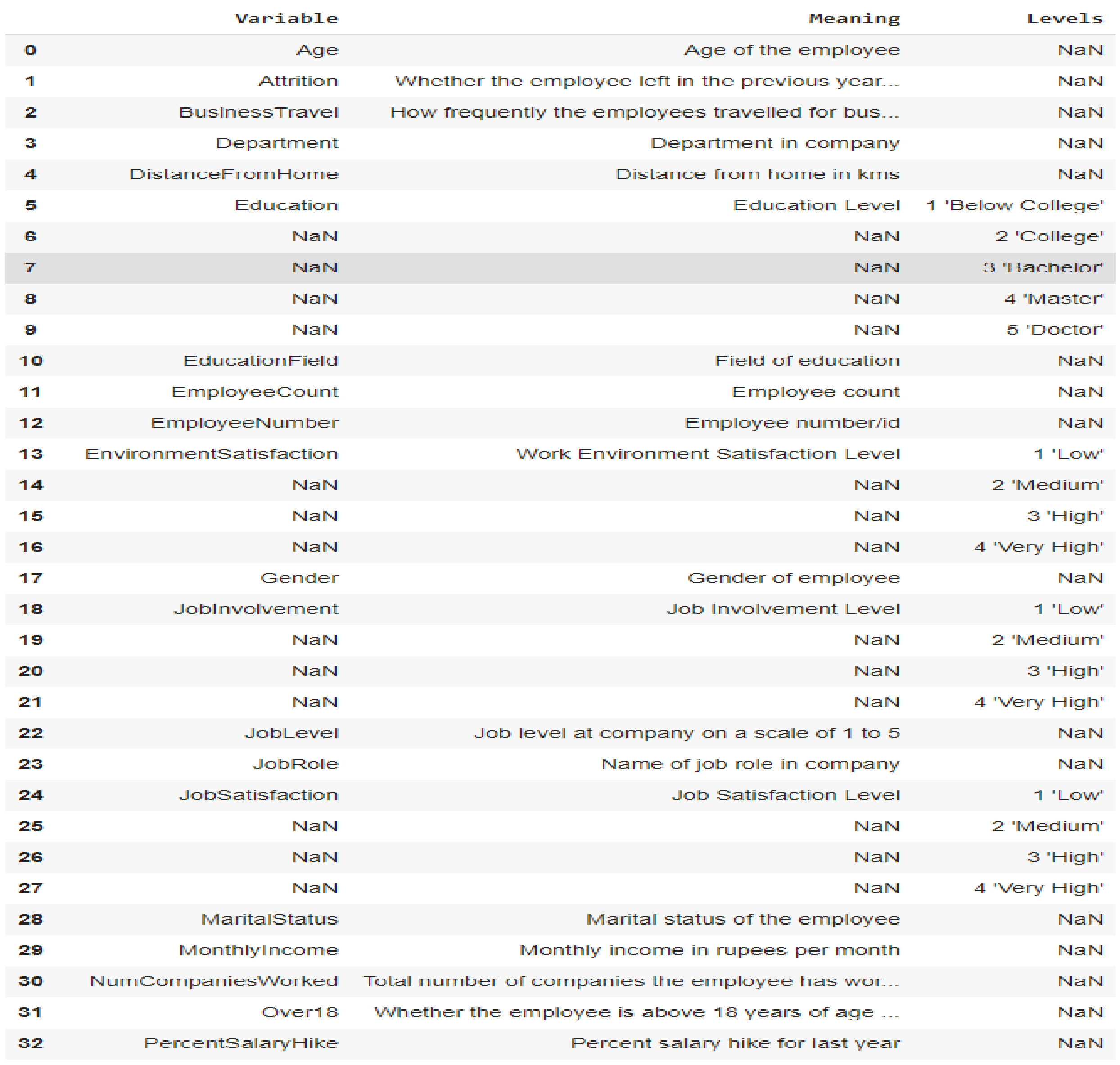
Figure 18: dropped columns

Employee Attrition Dataset:

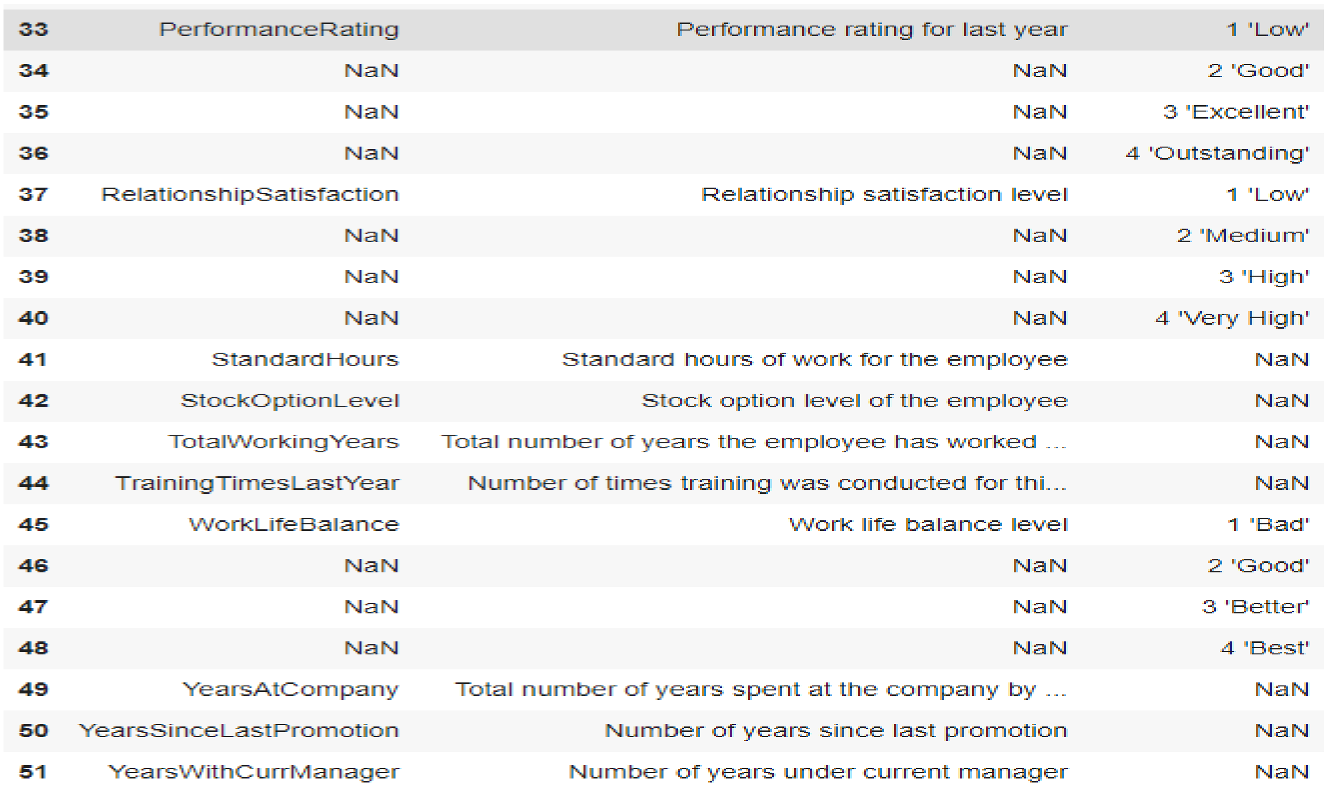
Data description

The employee attrition dataset is composed of characteristic from different employees, The main goal of this dataset is studying the reasons that leads to employees attrition, attrition meaning employees who left the organization.

The dataset is composed of different features, both categorical and continuous and represented in the table below.



	Variable	Meaning	Levels
0	Age	Age of the employee	NaN
1	Attrition	Whether the employee left in the previous year...	NaN
2	BusinessTravel	How frequently the employees travelled for bus...	NaN
3	Department	Department in company	NaN
4	DistanceFromHome	Distance from home in kms	NaN
5	Education	Education Level	1 'Below College'
6	NaN	NaN	2 'College'
7	NaN	NaN	3 'Bachelor'
8	NaN	NaN	4 'Master'
9	NaN	NaN	5 'Doctor'
10	EducationField	Field of education	NaN
11	EmployeeCount	Employee count	NaN
12	EmployeeNumber	Employee number/id	NaN
13	EnvironmentSatisfaction	Work Environment Satisfaction Level	1 'Low'
14	NaN	NaN	2 'Medium'
15	NaN	NaN	3 'High'
16	NaN	NaN	4 'Very High'
17	Gender	Gender of employee	NaN
18	JobInvolvement	Job Involvement Level	1 'Low'
19	NaN	NaN	2 'Medium'
20	NaN	NaN	3 'High'
21	NaN	NaN	4 'Very High'
22	JobLevel	Job level at company on a scale of 1 to 5	NaN
23	JobRole	Name of job role in company	NaN
24	JobSatisfaction	Job Satisfaction Level	1 'Low'
25	NaN	NaN	2 'Medium'
26	NaN	NaN	3 'High'
27	NaN	NaN	4 'Very High'
28	MaritalStatus	Marital status of the employee	NaN
29	MonthlyIncome	Monthly income in rupees per month	NaN
30	NumCompaniesWorked	Total number of companies the employee has wor...	NaN
31	Over18	Whether the employee is above 18 years of age ...	NaN
32	PercentSalaryHike	Percent salary hike for last year	NaN



33	PerformanceRating	Performance rating for last year	1 'Low'
34	NaN	NaN	2 'Good'
35	NaN	NaN	3 'Excellent'
36	NaN	NaN	4 'Outstanding'
37	RelationshipSatisfaction	Relationship satisfaction level	1 'Low'
38	NaN	NaN	2 'Medium'
39	NaN	NaN	3 'High'
40	NaN	NaN	4 'Very High'
41	StandardHours	Standard hours of work for the employee	NaN
42	StockOptionLevel	Stock option level of the employee	NaN
43	TotalWorkingYears	Total number of years the employee has worked ...	NaN
44	TrainingTimesLastYear	Number of times training was conducted for thi...	NaN
45	WorkLifeBalance	Work life balance level	1 'Bad'
46	NaN	NaN	2 'Good'
47	NaN	NaN	3 'Better'
48	NaN	NaN	4 'Best'
49	YearsAtCompany	Total number of years spent at the company by ...	NaN
50	YearsSinceLastPromotion	Number of years since last promotion	NaN
51	YearsWithCurrManager	Number of years under current manager	NaN

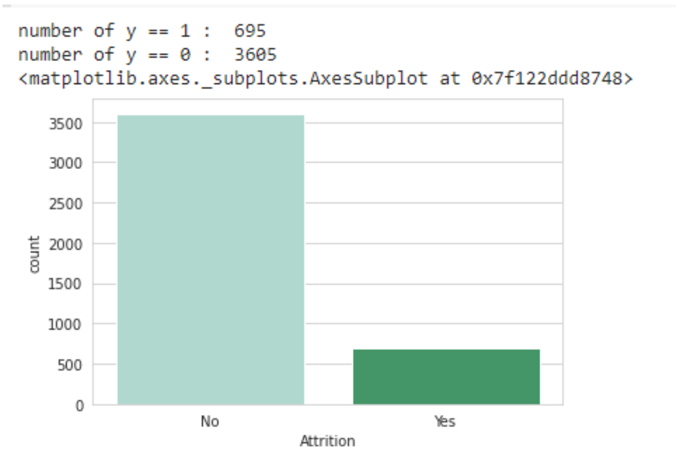
Figure 19: table representing the different features and their meaning

Each sample from the dataset is either an Attrition-yes or Attrition-No employee.

Data exploration

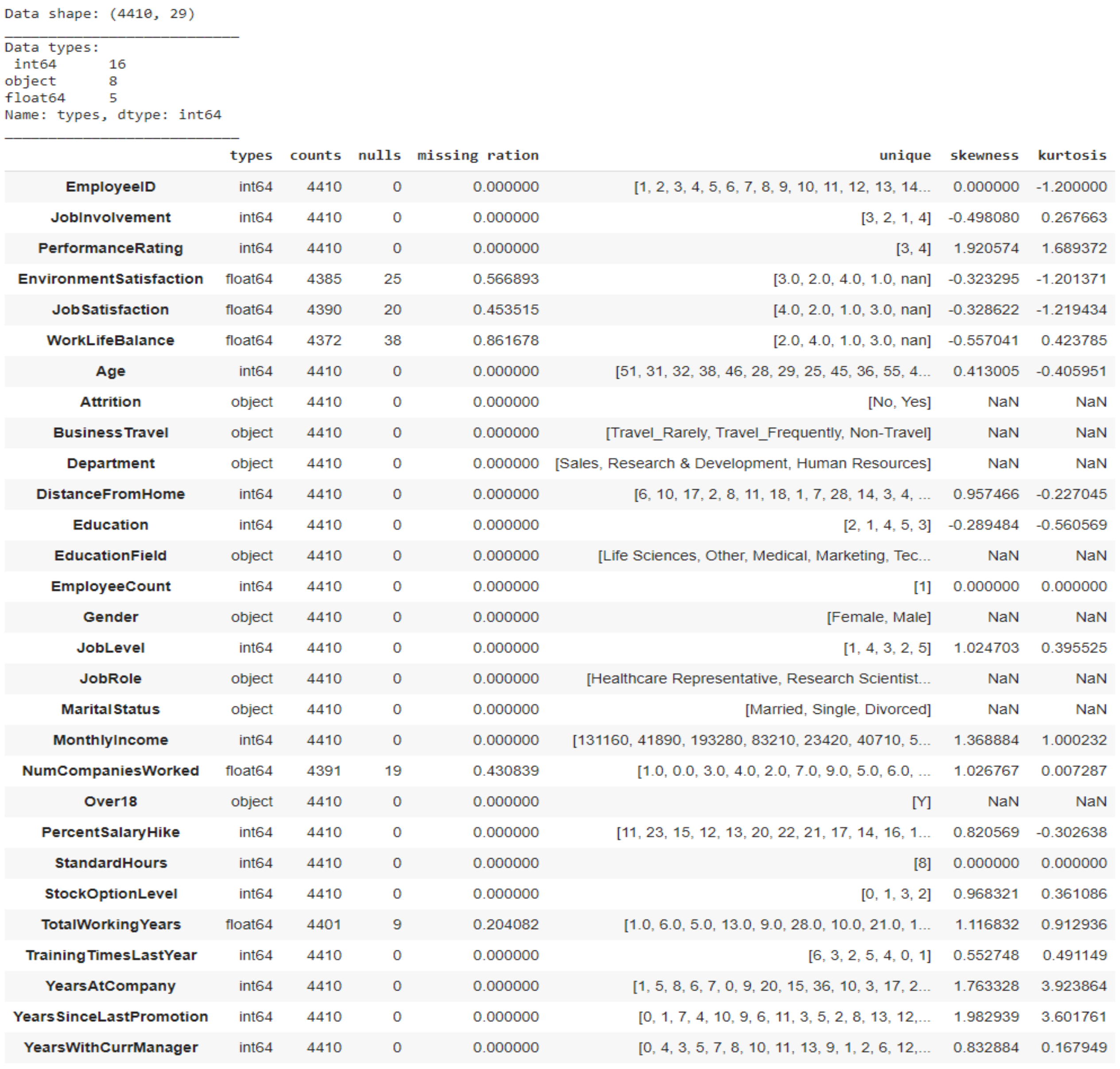
As we can see in Figure 21, the dataset is constituted of 4410 samples, we can also witness the presence of some missing values among our features: EnvironmentSatisfaction, JobSatisfaction, WorkLifeBalance, NumCompaniesWorked, since they just represent a small proportion of our data, we decided to drop those rows. We can also observe presence of some useless featuresthat contains only one modality: EmployeeCount, Over18, EmployeeID, StandardHours, which also will be dropped.

Also, for the feature PerformanceRating, we only found 2 modalities present in the dataset (excellent and outstanding).



```
number of y == 1 :  695
number of y == 0 :  3605
<matplotlib.axes._subplots.AxesSubplot at 0x7f122ddd8748>
```

Figure 20 Employées dataset distribution



```
Data shape: (4410, 29)
___________________________
Data types:
 int64      16
object      8
float64     5
Name: types, dtype: int64
___________________________
```

	types	counts	nulls	missing ration	unique	skewness	kurtosis
EmployeeID	int64	4410	0	0.000000	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	0.000000	-1.200000
JobInvolvement	int64	4410	0	0.000000	[3, 2, 1, 4]	-0.498080	0.267663
PerformanceRating	int64	4410	0	0.000000	[3, 4]	1.920574	1.689372
EnvironmentSatisfaction	float64	4385	25	0.566893	[3.0, 2.0, 4.0, 1.0, nan]	-0.323295	-1.201371
JobSatisfaction	float64	4390	20	0.453515	[4.0, 2.0, 1.0, 3.0, nan]	-0.328622	-1.219434
WorkLifeBalance	float64	4372	38	0.861678	[2.0, 4.0, 1.0, 3.0, nan]	-0.557041	0.423785
Age	int64	4410	0	0.000000	[51, 31, 32, 38, 46, 28, 29, 25, 45, 36, 55, 4...	0.413005	-0.405951
Attrition	object	4410	0	0.000000	[No, Yes]	NaN	NaN
BusinessTravel	object	4410	0	0.000000	[Travel_Rarely, Travel_Frequently, Non-Travel]	NaN	NaN
Department	object	4410	0	0.000000	[Sales, Research & Development, Human Resources]	NaN	NaN
DistanceFromHome	int64	4410	0	0.000000	[6, 10, 17, 2, 8, 11, 18, 1, 7, 28, 14, 3, 4, ...	0.957466	-0.227045
Education	int64	4410	0	0.000000	[2, 1, 4, 5, 3]	-0.289484	-0.560569
EducationField	object	4410	0	0.000000	[Life Sciences, Other, Medical, Marketing, Tec...	NaN	NaN
EmployeeCount	int64	4410	0	0.000000	[1]	0.000000	0.000000
Gender	object	4410	0	0.000000	[Female, Male]	NaN	NaN
JobLevel	int64	4410	0	0.000000	[1, 4, 3, 2, 5]	1.024703	0.395525
JobRole	object	4410	0	0.000000	[Healthcare Representative, Research Scientist...	NaN	NaN
MaritalStatus	object	4410	0	0.000000	[Married, Single, Divorced]	NaN	NaN
MonthlyIncome	int64	4410	0	0.000000	[131160, 41890, 193280, 83210, 23420, 40710, 5...	1.368884	1.000232
NumCompaniesWorked	float64	4391	19	0.430839	[1.0, 0.0, 3.0, 4.0, 2.0, 7.0, 9.0, 5.0, 6.0, ...	1.026767	0.007287
Over18	object	4410	0	0.000000	[Y]	NaN	NaN
PercentSalaryHike	int64	4410	0	0.000000	[11, 23, 15, 12, 13, 20, 22, 21, 17, 14, 16, 1...	0.820569	-0.302638
StandardHours	int64	4410	0	0.000000	[8]	0.000000	0.000000
StockOptionLevel	int64	4410	0	0.000000	[0, 1, 3, 2]	0.968321	0.361086
TotalWorkingYears	float64	4401	9	0.204082	[1.0, 6.0, 5.0, 13.0, 9.0, 28.0, 10.0, 21.0, 1...	1.116832	0.912936
TrainingTimesLastYear	int64	4410	0	0.000000	[6, 3, 2, 5, 4, 0, 1]	0.552748	0.491149
YearsAtCompany	int64	4410	0	0.000000	[1, 5, 8, 6, 7, 0, 9, 20, 15, 36, 10, 3, 17, 2...	1.763328	3.923864
YearsSinceLastPromotion	int64	4410	0	0.000000	[0, 1, 7, 4, 10, 9, 6, 11, 3, 5, 2, 8, 13, 12,...	1.982939	3.601761
YearsWithCurrManager	int64	4410	0	0.000000	[0, 4, 3, 5, 7, 8, 10, 11, 13, 9, 1, 2, 6, 12,...	0.832884	0.167949

Figure 21: Attrition dataset features analyses

We tried visualizing the proportion and density of different categorical features and their modalities among the attrition classes, the fig 14 represent some of the interesting features, and we tried to make some hypotheses:

- Environmental satisfaction can play a role in classification since we can see that the more satisfied the employee is with the environment, the lower the risk of quitting.

- Same thing can be said for the job satisfaction, we can see that those who are not satisfied have higher probability of resigning, and in contrast employees that are very highly satisfied have less probability. However, the modalities medium and high satisfied seems to have equal proportion and thus will not contribute in the classification.
- For the marital status we can clearly see that single employees are more likely to quit their jobs. Both married and divorced show high probability of keeping their job

- For the travel features, we can notice that those who travel frequently are more prone to quit their jobs. In contrast, employees that never travel or do it rarely are more inclined to stay.

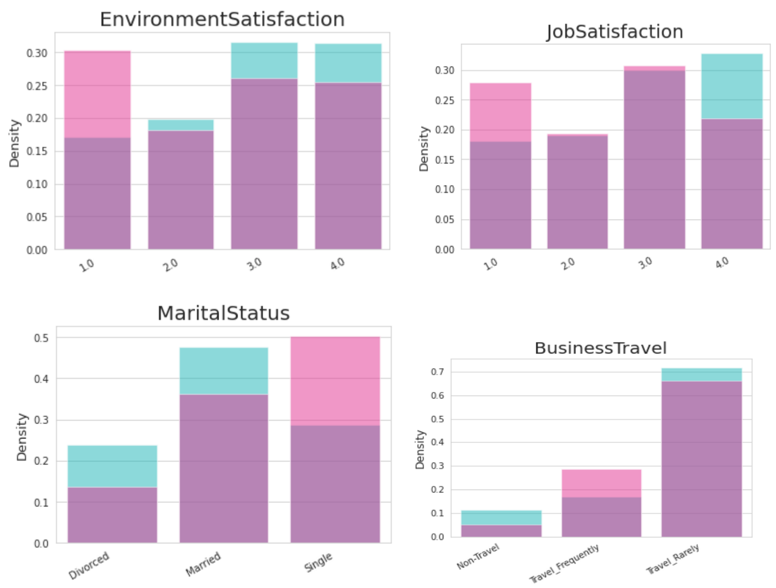


Figure 22: some categorical features and their modalities distribution among attrition class

Quantitative features show a highly overlapped distribution over the Attrition classes, However fig 15 shows some interesting results for Age and Monthly Income Distribution:

- For the age features we can notice the existence of 2 age categories, first the youngest employees from 18 to 35, who are more likely to keep their job. other employees over the age of 35 are more likely to leave their job.

- For the monthly income, it is harder to interpret the distribution of features within the attrition class. However, the boxplot seems to show some outliers we will have to deal with.

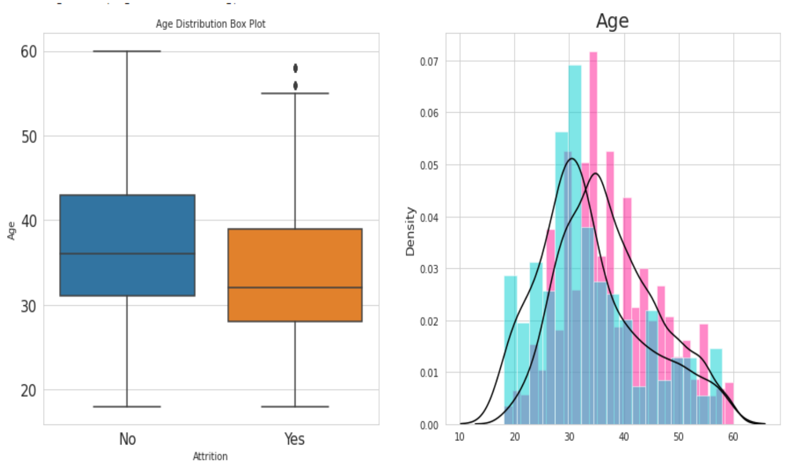


Figure 23 Age feature boxplot and distribution for both classes

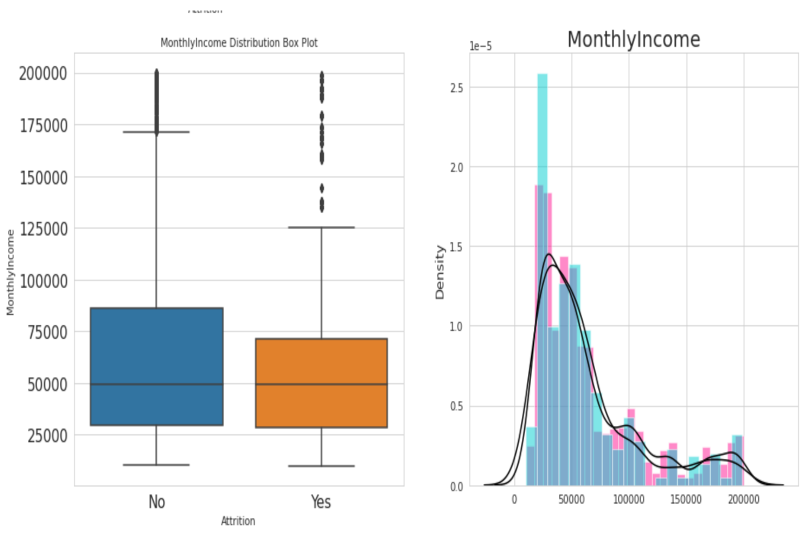
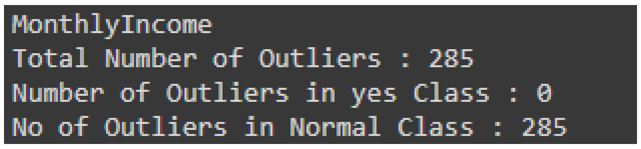


Figure 24: monthlyincome feature boxplot and distribution for both classes

Data preparation

We first started by converting our categorical features into one hot encodings, then we processed our data by removing outliers that has been located on different continuous features. We used Interquartile based method with two different thresholds.

After removing outliers, we applied the log transform on some continuous features to reduce the range of the values and get closer to a normal shape. In addition, we scaled our features using std scaling to keep all of our features in the same range.


```
MonthlyIncome
Total Number of Outliers : 285
Number of Outliers in yes Class : 0
No of Outliers in Normal Class : 285
```
Figure 25 Number of outliers

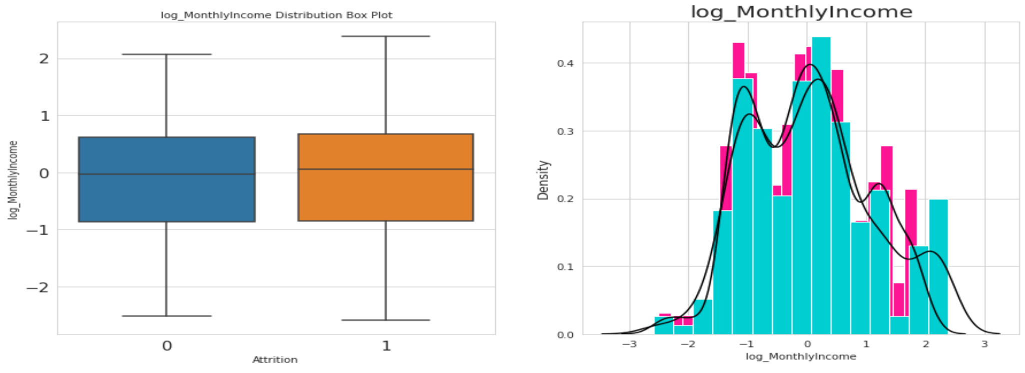


Figure 26: Monthly income after outlier removal and log transformation

Final step, is features selection. For the continuous variables, we checked our features correlation to the target variable (Figure 27). Similar to the fraud dataset we can observe a really low correlation with the target, this can indicate the presence of nonlinear relation rather than linear.

For our categorical variables, we did a Chi2 to find which modalities have a dependence relation with the target variable. Figure 28 display the modalities that has been dropped and we can notice the presence of some of the visual interpretation, we can see that the low satisfaction environment and job has been kept and both high and medium job satisfaction has been dropped.

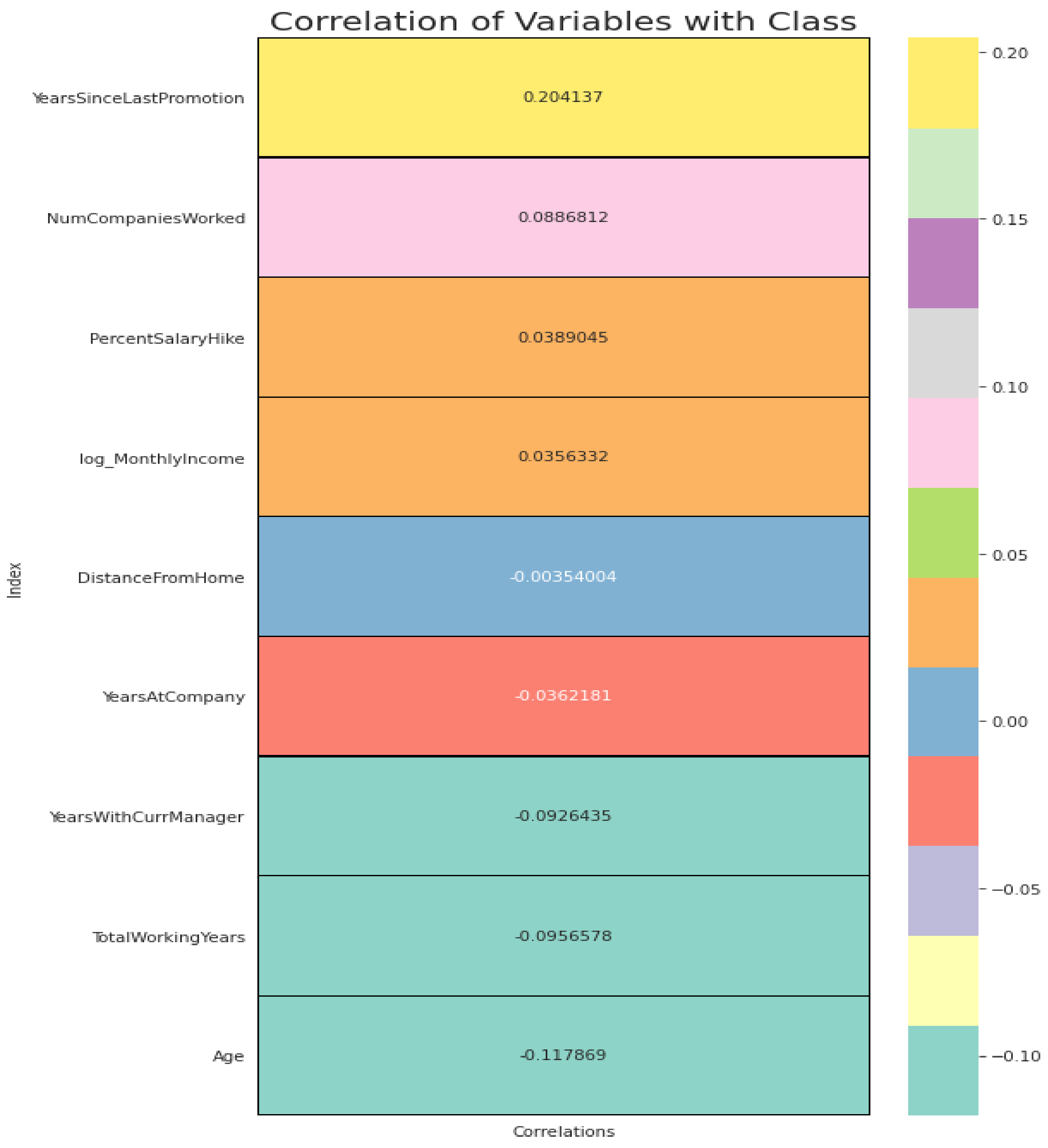
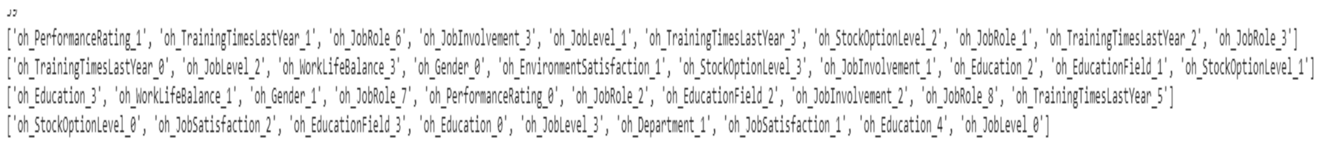


Figure 27: contunious features correlation with attrition



```
['oh_PerformanceRating_1', 'oh_TrainingTimesLastYear_1', 'oh_JobRole_6', 'oh_JobInvolvement_3', 'oh_JobLevel_1', 'oh_TrainingTimesLastYear_3', 'oh_StockOptionLevel_2', 'oh_JobRole_1', 'oh_TrainingTimesLastYear_2', 'oh_JobRole_3']
['oh_TrainingTimesLastYear_0', 'oh_JobLevel_2', 'oh_WorkLifeBalance_3', 'oh_Gender_0', 'oh_EnvironmentSatisfaction_1', 'oh_StockOptionLevel_3', 'oh_JobInvolvement_1', 'oh_Education_2', 'oh_EducationField_1', 'oh_StockOptionLevel_1']
['oh_Education_3', 'oh_WorkLifeBalance_1', 'oh_Gender_1', 'oh_JobRole_7', 'oh_PerformanceRating_0', 'oh_JobRole_2', 'oh_EducationField_2', 'oh_JobInvolvement_2', 'oh_JobRole_8', 'oh_TrainingTimesLastYear_5']
['oh_StockOptionLevel_0', 'oh_JobSatisfaction_2', 'oh_EducationField_3', 'oh_Education_0', 'oh_JobLevel_3', 'oh_Department_1', 'oh_JobSatisfaction_1', 'oh_Education_4', 'oh_JobLevel_0']
```

Figure 28: categorial modalities that has been dropped

Resampling methods

An imbalanced classification problem is an example of a classification problems where the distribution of examples across the known classes is biased or skewed. The distribution can vary from a slight bias to a severe imbalance where there is one example in the minority class for hundreds, thousands, or millions of examples in the majority class or classes.

Building application and optimization of learning algorithms on imbalance datasets can and will lead to semantically incorrect models.

Several different techniques exist in the practice for dealing with imbalanced dataset. In this report will highlight some of the important resampling methods that can be used.

Over sampling methods

Random oversampling is a naïve approach that consist on duplicating examples from the minority class, the duplication is done by randomly selecting samples from the rare class. As shown in the below Figure 29 we can see that the plot didn't change, this is due to the fact that we are only duplicating the original samples and thus the point are superposed on each other.

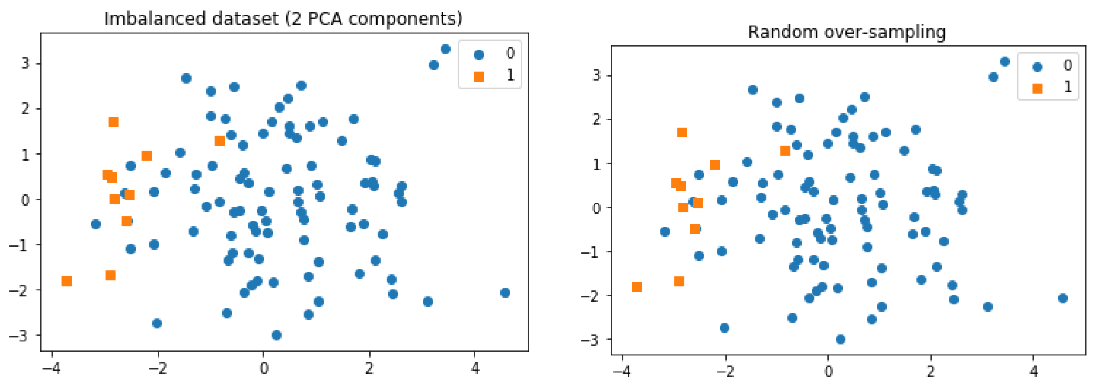


Figure 29: left original plot, right using oversampling

SMOTE (Synthetic Minority Oversampling Technique) also consists creating newer element from the minority class, based on those that already exist. It works by randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors.

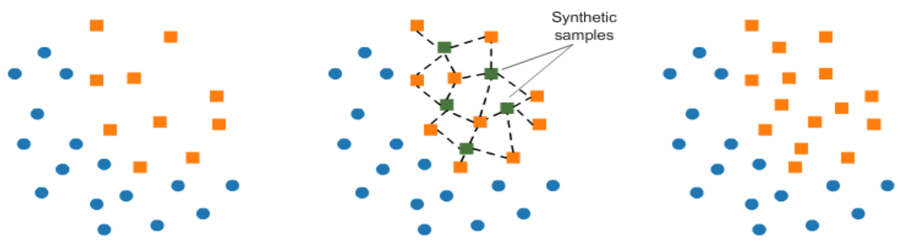


Figure 30: oversampling usin SMOTE

ADASYN (Adaptive Synthetic) is also an algorithm that generates synthetic data, it builds on the methodology of SMOTE, by shifting the importance of the classification boundary to those minority classes which are difficult. its strength is that it is not copying the same minority data but rather generates data for minority class examples that are harder to learn.

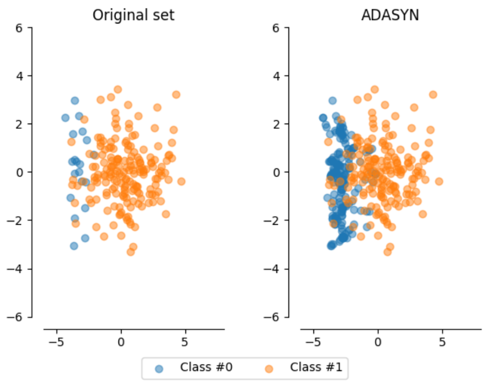


Figure 31: ADASYN example

Under sampling methods

Random under sampling is another naïve method that aims to reduce samples from the majority class by randomly selecting examples from the majority class and deleting them from the training dataset. Doing this can result into a loss in valuable information's since and thus bias our model.

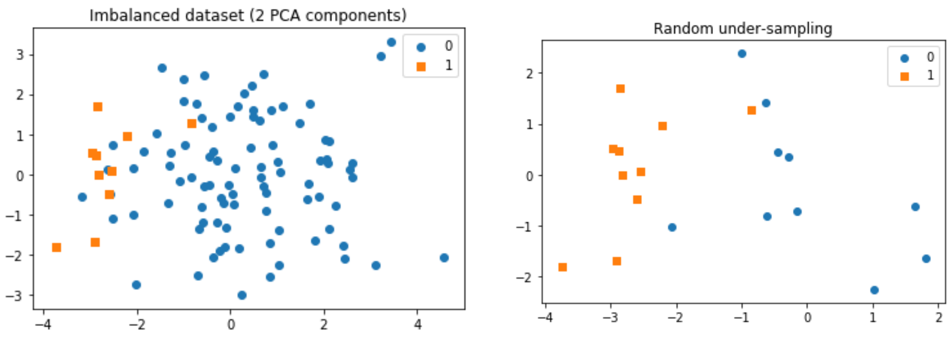


Figure 32: using random undersampling

Tomek links on the other hand is a method aims into removing the overlap between classes by removing samples from the majority class, this will be helpful since doing this will increase the space between the classes and thus facilitate the classification process.

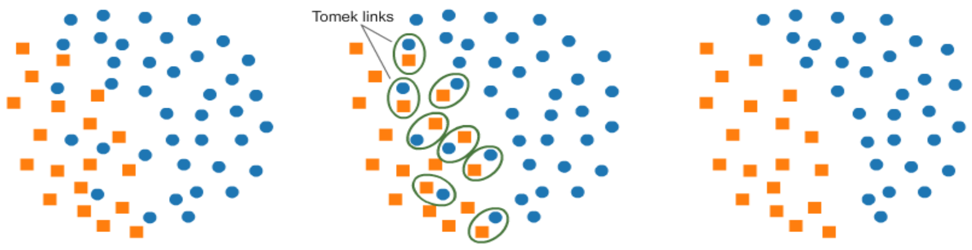


Figure 33: Tomek Links method

Hybrid methods

Hybrid methods aims to reduce the negative effect of applying oversampling and undersampling method in isolation, and thus decrease the negative effect of those methods which are: overfitting and information loss.

- **Random oversampling + Tomek links:** this methods aims to reduce the effect of overfitting induced by the random oversampling by applying a reduced oversampling (by changing the oversampling ratio) followed by undersampling focusing on reducing the overlap between the classes by using tomek links.

- **SMOTE + Tomek links:** Although SMOTE is and interesting oversampling method it can fails to take into account neighboring examples that may come from other classes and thus increase the probability of class overlap and introduce additional noise. To decrease this impact, we can rely on applying tomek links to reduce the overlap that can be added to the original data.

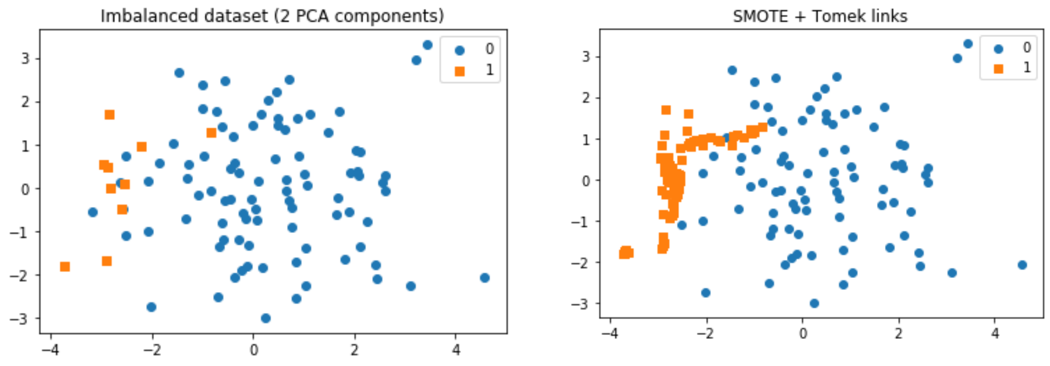


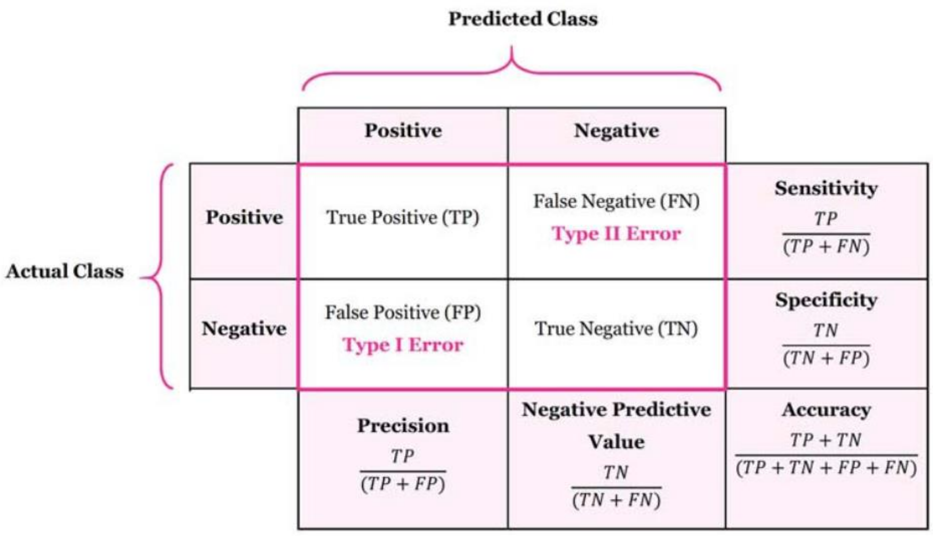
Figure 34: applying smote + tomek

Class weights

Another method to deal with imbalanced dataset is modifying our neural network model parameters to take into account the skewed distribution of our classes. This can be done by giving different weights to both the majority and minority classes. The whole purpose is to penalize the misclassification made by the minority class by setting a higher class weight and reducing at the same time weights corresponding to the majority class.

Evaluation Metrics

To evaluate our models and methods, different metrics were used as a comparison, some couldn't be trusted as a relevant metric of performance in the case of imbalance.



Predicted Class

		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Figure 35 Relatios between actual and predicted classes

Accuracy

Accuracy is the simplest way to evaluate a model as it simply consists of calculating the ratio of correct predictions (% of correct prediction).

When classes are balanced, it is an effective solution to evaluate a classification model, but in our case where data is strongly imbalanced, we cannot trust such metric. As an example, in a dataset where 99% of our labels correspond to the same class, a model just have to predict such class every time to classify correctly 99% of the time, but in cases like fraud detection, where we are much more interested in classifying the minority classes, this metric isn't adapted.

F1score / Precision / Recall

Precision and recall share a complementary relationship, Precision being the ratio of correctly predicted positive observations to the total predicted positive observations, recall on the other hand, is the ratio of correctly predicted positive observations to the all observations in actual class. Finally, F1-score is simply a weighted average of those two.

In case of balanced data, those are even greater performance metrics and usually more trusted than accuracy, but in the case of imbalanced data where we are more concerned about the minority class (like fraud detection), we usually care more about the recall of the minority class without giving too much attention to precision.

ROC and AUC

AUC, or Area under curve, is the area under the ROC curve.

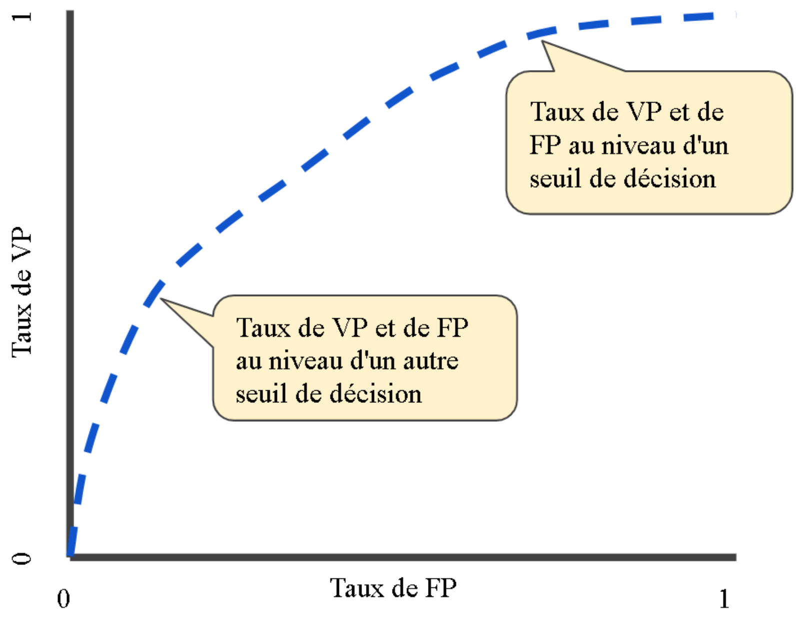


Figure 36 ROC curve

The ROC is a curve which is plotted between the False positive rate and True positive rate, AUC and ROC are performance measurement for the classification problems at various threshold settings. It tells how much the model is capable of distinguishing between classes, Higher the AUC, the better the model is at predicting true classes.

For the case of imbalanced data, those are one of the most important metrics which perfectly balance the importance of getting a good true positive ratio with the importance of reducing the false positive ratio as much as possible, and will be the metric we will use to track the performance of our models, as it perfectly fit our case.

Models

We tried our different methods using two different models and compared their performance on each, one simple base model and a more complex CNN 1D, we can't make a model too big due to lack of data's which could result in overfitting.

For each model, we use a sigmoid activation function on the last layer with a binary cross entropy loss and an Adam optimization function.

The primary metric we will use to track the performance of ou model will be the AUC on the validation dataset, we will train the model to a maximum of 1000 epochs with an early stopping setting that will stop training if no increase in performance is noticed after 30epochs.

Base

The initial base model is a simple one hidden layer model with 16 neurons followed a “relu” activation function and a dropout layer of a 0.5 probability of dropout.

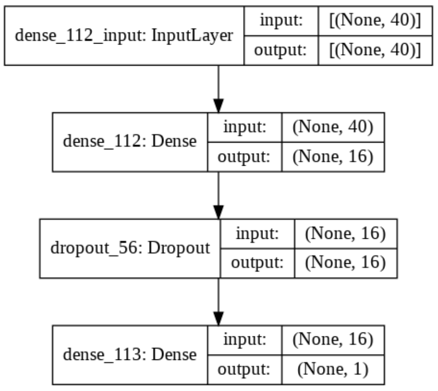


Figure 37 Base model architecture

The model is really simple and doesn’t require much data to be trained, which makes it really efficient and less prone to overfitting.

CNN 1D

Even though convolutional neural networks were originally developed for image classification problems, where the model learn a representation of a two-dimensional input, it turned out really useful in other data types like one-dimensional sequence of data, and such model are able to extract features from sequences of observations and map its feature to its corresponding class.

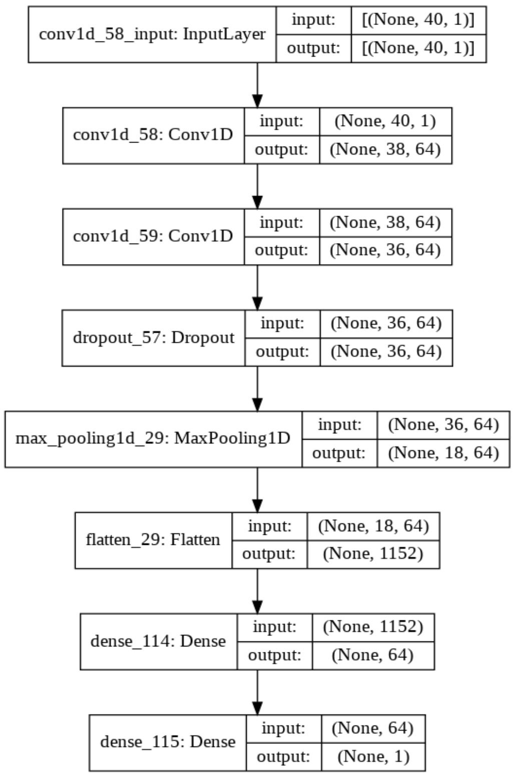


Figure 38 CNN 1D model architecture

In our case, the model we trained is composed of:

- 2 CNN layers containing both 32 filters with a kernel size of 3, each followed by a relu activation function
- We follow our two CNN layers by a dropout layer of probability 0.5, followed by a MaxPooling layer with a pool size of 2.
- The last layer is also a CNN layer similar to the previous ones with the exception that we will use 64 filters this time.

This model is much more complex and deeper than the first one, its architecture makes it more efficient but also more prone to overfitting, a clear case of overfitting can be seen below for the Adasyn resampling method on the bank dataset:

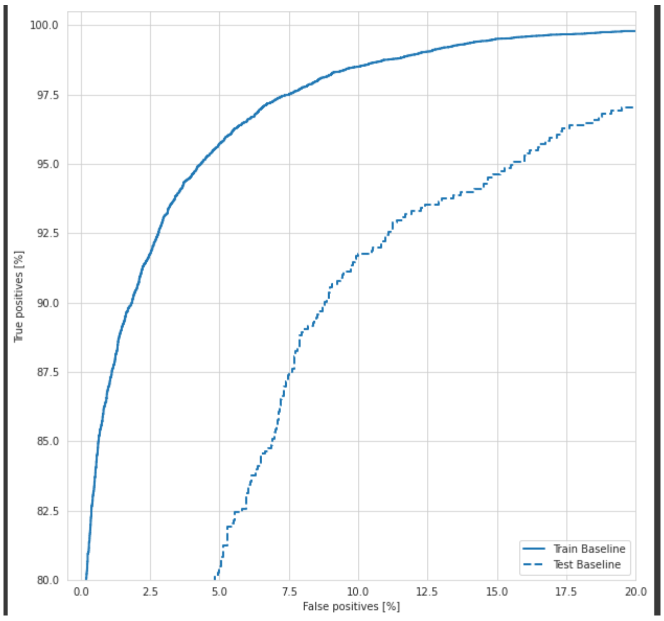


Figure 39 CNN overfitting on data example

even though the CNN model got slightly better results than the Base model (+0.3% AUC), we can clearly improve those result and further test should be done to improve our results.

Results discussion

For our results, we will primarily track and compare the performance of our models based on the AUC metric on the test set, as it perfectly fit the case of imbalanced data, and other results (accuracy / Recall etc..) depends on it.

We will also plot the ROC curve of all our models for each dataset at the end to make an overall result comparison of each method.

Simple training (Without resampling)

	Bank		Fraud		Employee	
Model	BASE	CNN	BASE	CNN	BASE	CNN
Simple training	96.0	**97.0**	87.1	**92.9**	80.0	**84.4**

Figure 40 initial results (AUC on test set)

First, starting by comparing our results on each dataset for both our models without resampling or adding weights, we can clearly see that the CNN model outperform the Base model on every dataset. This is probably due to the fact that CNN is able to extract features and find patterns that couldn't be interpreted by the BASE model.

However, with proper preprocessing, we can see that they both give pretty good results.

Finally, we can note that the "Employee attrition" dataset is the one giving the worst results in this case, this may be due to a lack of data as it is the smallest dataset of the 3.

Training with weights

	Bank		Fraud		Employee	
Model	BASE	CNN	BASE	CNN	BASE	CNN
Weighted	96.5	**97.0**	98.6	98.6	**83.5**	82.9

Figure 41 results with class weights (AUC on test set)

We can clearly see that adding weights to our models had a positive impact on the overall results, especially in the case of Fraud, which is the dataset with the smallest minority class ratio (0.17%) and the one getting the highest increase in AUC score for both models (+11.5% for BASE), the inverse observation can be made looking at the "employee attrition" dataset, which even got worse result then without adding weights for the case of the CNN model.

We can thus deduce that adding weights to our model classes has a stronger impact and more benefit when there is a bigger ratio gap between our classes, and more data available for training.

Training with over-sampling

	Bank		Fraud		Employee	
Model	BASE	CNN	BASE	CNN	BASE	CNN
ROS	96.5	**96.9**	**98.9**	97.4	88.1	**95.6**
SMOTE	96.5	**97.0**	**98.4**	97.7	85.8	**94.5**
ADASYN	96.5	**96.8**	**98.1**	97.3	88.5	**93.2**

Figure 42 results with Over sampling methods (AUC on test set)

We can see that both models had an increase in AUC score compared to the training without resampling, also, for the case of the employee dataset, we can clearly see that oversampling is more efficient then adding weights.

If we now compare between each oversampling method, at first glance, we can see that their performance seems pretty much equivalent, with a slightly weaker performance for ADASYN.

Another thing that can easily be noticed if we now compare our models performance on each dataset, is that for Bank and especially employee dataset, CNN outperform BASE results, in contrast with Fraud dataset where BASE had better result than CNN, this can be explained by the fact that the ratio of minority classes for “Fraud” is much smaller than the other two (0.17%), and thus, when oversampling, since we have too few data for the minority data’s example, it leads to overfitting when a model has too much weights.

Training with under-sampling

	Bank		Fraud		Employee	
Model	BASE	CNN	BASE	CNN	BASE	CNN
RUS	96.2	**96.8**	97.4	**98.0**	83.8	**91.7**
TOMEKLINKS	96.2	**97.0**	87.3	**94.0**	80.5	**83.5**

Figure 43 results with under sampling methods (AUC on test set)

First thing we can notice comparing to previous results, is that under sampling has smaller positive increase in AUC compared to over sampling.

Another thing that can be noticed is that TomekLinks had little to no impact compared to training without resampling, this can be due to the nature of our data (especially after preprocessing) which may have insignificant overlapping between their classes.

Focusing now on random under sampling, it had way better results than both tomek links and training without resampling, and CNN had way better results than BASE on all datasets.

Training with hybrid methods

	Bank		Fraud		Employee	
Model	BASE	CNN	BASE	CNN	BASE	CNN
SMOTE + TOMEKLINKS	96.5	**96.8**	**98.4**	97.7	85.8	**94.5**
ROS + TOMEKLINKS	96.4	**96.9**	**98.8**	98.6	83.8	**94.5**

Figure 44 results with hybrid resampling methods (AUC on test set)

First of all, we can make the same observation as the over sampling methods for the case of the Fraud dataset, which is that oversampling on highly imbalanced data makes complex models overfit, but in this case we can see that the effect of overfitting is slightly reduced with slightly better results due to the under sampling method that increase the bias in our data.

Now comparing between each method, we can see that both methods are almost equivalent with no noticeable benefit on one compared to the other.

Finally, those two methods don't seem to add any benefit compared to the over sampling methods, this may due to previous observation made on tome links, and further study could produce better results combining oversampling methods with random under sampling.

Overall results comparison

	Bank		Fraud		Employee	
Model	BASE	CNN	BASE	CNN	BASE	CNN
Simple training	96.0	97.0	87.1	92.9	80.0	84.4
Weighted	96.5	97.0	98.6	98.6	83.5	82.9
ROS	96.5	96.9	98.9	97.4	88.1	95.6
SMOTE	96.5	97.0	98.4	97.7	85.8	94.5
ADASYN	96.5	96.8	98.1	97.3	88.5	93.2
RUS	96.2	96.8	97.4	98.0	83.8	91.7
TOMEKLINKS	96.2	97.0	87.3	94.0	80.5	83.5
SMOTE +TOMEKLINKS	96.5	96.8	98.4	97.7	85.8	94.5
ROS +TOMEKLINKS	96.4	96.9	98.8	98.6	83.8	94.5

Figure 45 overall results comparison (AUC on test set)

If we first compare our two models, we can conclude that CNN is more efficient overall but simpler models like BASE may be a viable solution especially using with a good preprocessing and a highly imbalanced dataset.

Now comparing our sampling methods, they are all pretty much equivalent, with both their strength and weaknesses, and it mostly depend on the nature of the data we are working with, and if we look at the ROC curves we can deduce that adding weights is only advisable when dealing with a high minority class ratios, and the most robust methods overall were ROS, RUS and SMOTE, which gave stable results on every dataset.

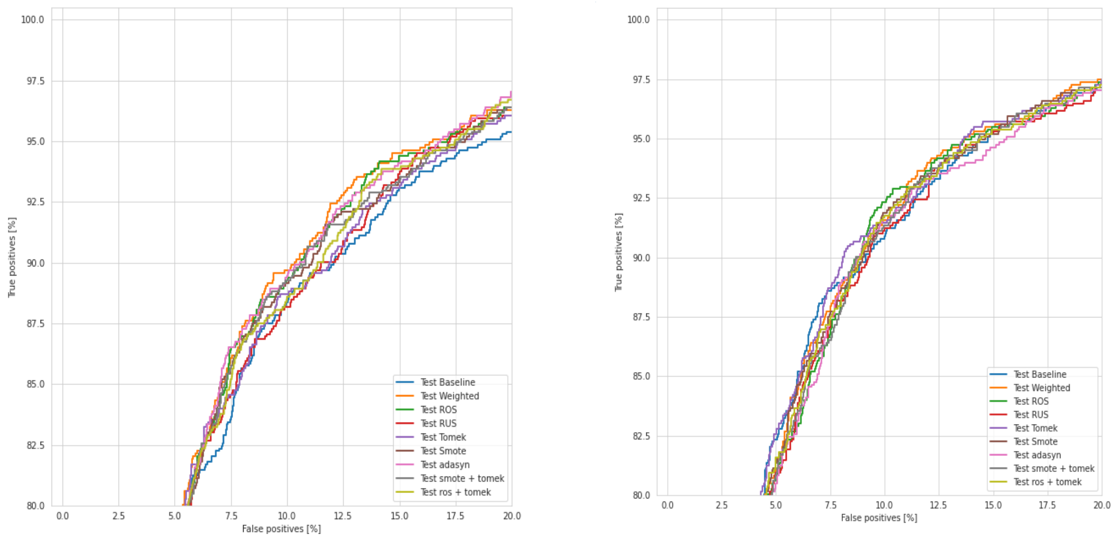


Figure 46 ROC for all methods on BANK dataset (BASE left CNN right)

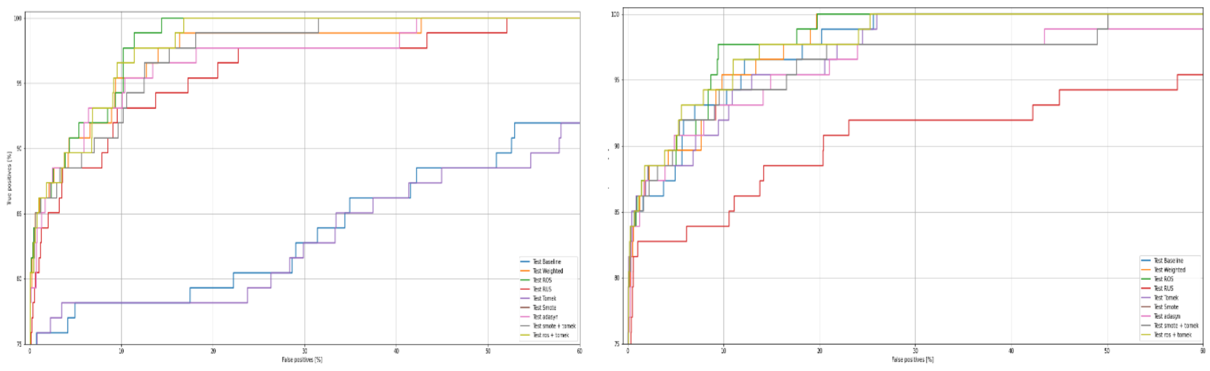


Figure 47 ROC for all methods on FRAUD dataset (BASE left CNN right)

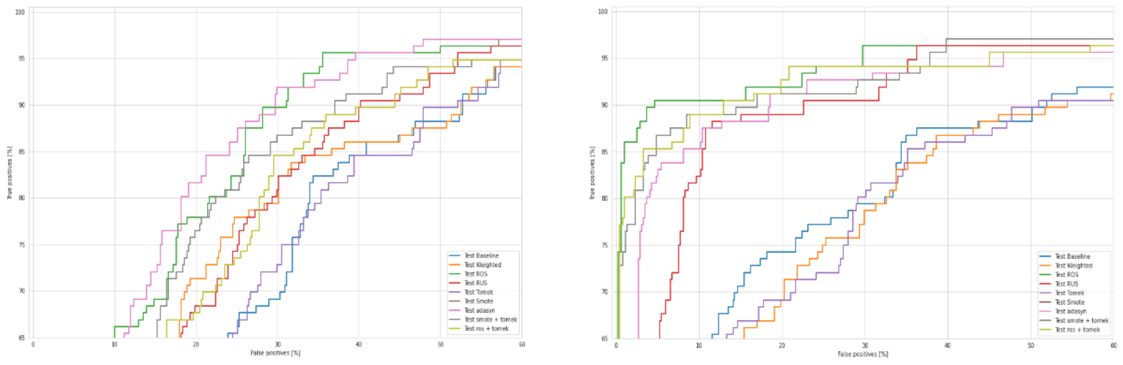


Figure 48 ROC for all methods on EMPLOYEE dataset (BASE left CNN right)

Conclusion

During this project, we had the opportunity to experiment different resampling methods on different imbalanced datasets using both simple and complex deep learning models.

to implement such methods involves understanding the nature of our data using different visualization tools to be able to preprocess it efficiently as a way to extract relevant features.

Also, dealing with such imbalanced data can be done using different sampling methods, each having their strength and weaknesses, however, from our results, we can conclude that choosing the right sampling method mostly depend on the nature of the data we are dealing with and the complexity of the model we are working with.

References

[1] [Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, In press, http://dx.doi.org/10.1016/j.dss.2014.03.001

[2] How to Improve Class Imbalance using Class Weights in Machine Learning, https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/

[3] How to Combine Oversampling and Undersampling for Imbalanced Classification, https://machinelearningmastery.com/combine-oversampling-and-undersampling-for-imbalanced-classification/

[4] Déclarer la guerre aux données déséquilibrées : SMOTE, https://blog.soat.fr/2019/12/techniques-augmentation-dataset-smote/

[5] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015

[6] Dal Pozzolo, Andrea; Caelen, Olivier; Le Borgne, Yann-Ael; Waterschoot, Serge; Bontempi, Gianluca. Learned lessons in credit card fraud detection from a practitioner perspective, Expert systems with applications,41,10,4915-4928,2014, Pergamon

[7] Dal Pozzolo, Andrea; Boracchi, Giacomo; Caelen, Olivier; Alippi, Cesare; Bontempi, Gianluca. Credit card fraud detection: a realistic modeling and a novel learning strategy, IEEE transactions on neural networks and learning systems,29,8,3784-3797,2018,IEEE

[8] Dal Pozzolo, Andrea Adaptive Machine learning for credit card fraud detection ULB MLG PhD thesis (supervised by G. Bontempi)

[9] Carcillo, Fabrizio; Dal Pozzolo, Andrea; Le Borgne, Yann-Aël; Caelen, Olivier; Mazzer, Yannis; Bontempi, Gianluca. Scarff: a scalable framework for streaming credit card fraud detection with Spark, Information fusion,41, 182-194,2018,Elsevier

[10] Carcillo, Fabrizio; Le Borgne, Yann-Aël; Caelen, Olivier; Bontempi, Gianluca. Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization, International Journal of Data Science and Analytics, 5,4,285-300,2018,Springer International Publishing

[11] Bertrand Lebichot, Yann-Aël Le Borgne, Liyun He, Frederic Oblé, Gianluca Bontempi Deep-Learning Domain Adaptation Techniques for Credit Cards Fraud Detection, INNSBDDL 2019: Recent Advances in Big Data and Deep Learning, pp 78-88, 2019

[12] Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, Frederic Oblé, Gianluca Bontempi Combining Unsupervised and Supervised Learning in Credit Card Fraud Detection Information Sciences, 2019