HR Analytics: Job Change of Data Scientists

Machine Learning and Data Mining project

Sara Raccagni

2022/2023

# **1. INTRODUCTION**

A company which is active in Big Data and Data Science wants to hire data scientists among people who successfully pass some courses. A lot of candidates sign up for the training, therefore, the company would know which ones really want to work for them after training or are looking for a new employment. The idea is to avoid, or at least reduce, the waste of time and money in training people who are not really interested in, as well as better manage the course planning.

The purpose of this work is to understand which factors most affect candidates’ decisions, and to predict the probability that a candidate will work for the company or look for a new job. Several machine learning techniques, such as Decision Tree, Random Forest, Extra Trees, XGBoost, SVM and Neural Networks, are applied to a dataset of 19158 records. During the analysis, some plots are used to graph the outcomes obtained, and the evaluation is performed using a complete set of metrics such as Accuracy, Precision, Recall, Specificity, F1-score and AUC.

This report is organized in several sections that describe the main components of the experiment. After a brief presentation of the dataset used, that covers section 2, section 3 is about feature distribution, and section 4 is about data preprocessing. In particular, the latter includes the application of several feature engineering techniques such as the handling of missing values, the data sampling, the feature scaling and encoding, and the feature correlation evaluation. Section 5 shows the training of classification models while section 6 is about Auto-Sklearn. Finally, section 7 compares the results obtained ending with the conclusion.

# 2. DATASET DESCRIPTION

This dataset has been downloaded from Kaggle dataset repository, to whom it was donated in 2021 by Arashnic Mobius. It includes a training set of 19158 records with 14 attributes each, and a test set provided for the Kaggle submission. The latter consists of 2129 records where target values are not included since predicting them is a task of this work.

The input attributes related to a single candidate are:

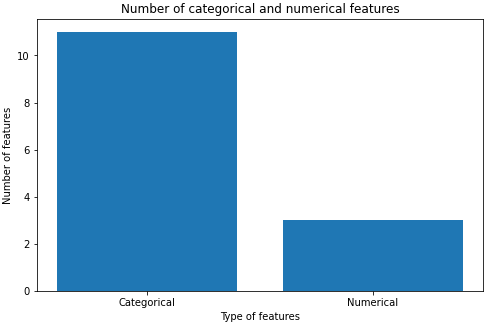
* Unique ID
* City code
* City Development index
* Gender
* Relevant experience
* University enrollment
* Education level
* Education major discipline
* Total experience
* Company size
* Company type
* Last new job
* Training hours

All this information about demographics, education and experiences was collected through the applicants' enrollment form.

The target value is binary:

* Not looking for a new job
* Looking for a new job

Most of the features are categorical (nominal, ordinal and binary), some with high cardinality, while the remaining are numerical.



Categorical features, on the left, are 11 and numerical features, on the right, are 3.

**Categorical features**

| **Feature** | **Description** | **# unique values** | **Values** |
| --- | --- | --- | --- |
| city | city code | 123 | city\_X, where X is a positive integer |
| gender | candidate’s gender | 3 | Male, Female, Other |
| relevent\_experience | candidate’s relevant experience | 2 | Has relevent experience, No relevent experience |
| enrolled\_university | type of university enrollment | 3 | No enrollment, Part time course, Full time course |
| education\_level | candidate’s education level | 5 | Primary school, High school, Graduate, Masters, Phd |
| major\_discipline | candidate’s major education discipline | 6 | STEM, Business Degree, Arts, Humanities, No Major |
| experience | candidate’s total experience in years | 22 | <1, 1, 2, …, 19, 20, >20 |
| company\_size | number of employees in the current employer’s company | 8 | <10, 10-49, 50-99, 100-500, 500-999, 1000-4999, 5000-9999, 10000+ |
| company\_type | type of current employer’s company | 6 | Pvt Ltd, Funded Startup, Early Stage Startup, Public Sector, NGO, Other |
| last\_new\_job | difference in years between previous and current job | 6 | never, 1, 2, 3, 4, >4 |
| target | (not) looking for job change | 2 | 0-Not looking for a new job  1-Looking for a new job |

**Numerical features**

| **Feature** | **Description** | **# unique values** | **Values** |
| --- | --- | --- | --- |
| enrollee\_id | candidate’s ID | 21287 | 1, 2, 3, … 33377, 33379, 33380 |
| city\_development\_index | development index of the city | 93 | scaled value between 0 and 1 |
| training\_hours | training hours completed | 241 | value between 1 and 336 |

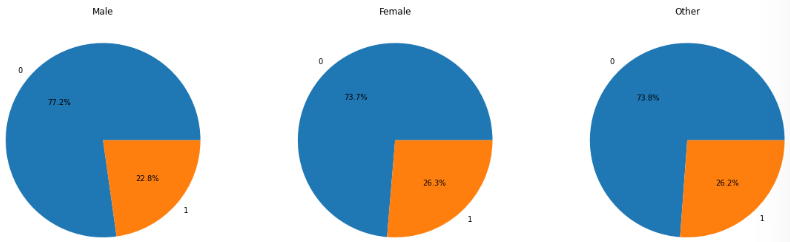
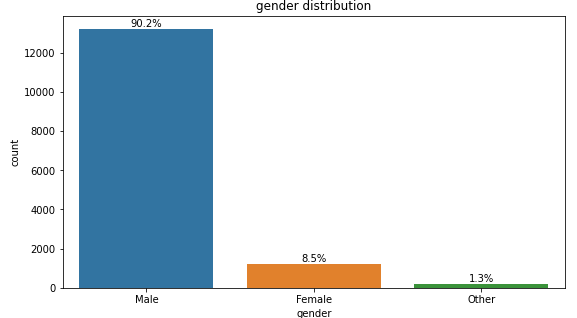
# **3. FEATURE DISTRIBUTION**

In this section several plots are used to analyze the feature distribution. It can be useful to understand how the values of each feature are distributed within the dataset and how they are splitted between the target classes 0 and 1. Moreover, some of this information will be used in the data preprocessing section during the handling of missing values and feature scaling.

Note that during this analysis, ID and city code attributes will not be taken into consideration since they are not so relevant for the task. Later, in the next section, we will go into more detail about their removal from the dataset.

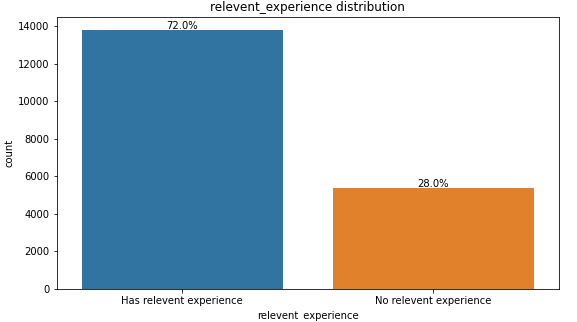
Let’s have a look at categorical features where count plots and pie charts are used for the analysis.

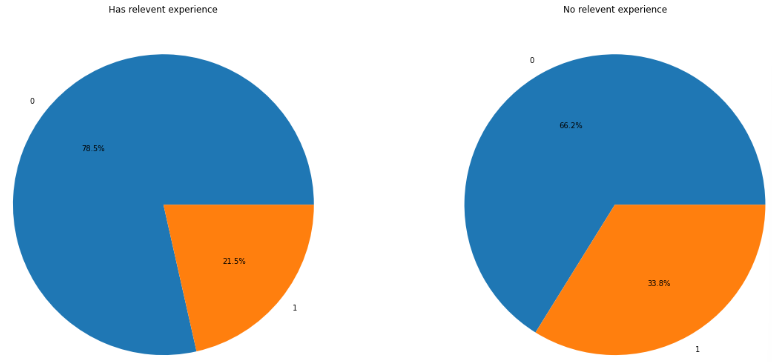
**Gender**

****

The 90% of the dataset consists of men, around 9% of women, while a very small percentage belongs to other categories. In all cases the number of people who will not look for a new job is higher (around 75%).

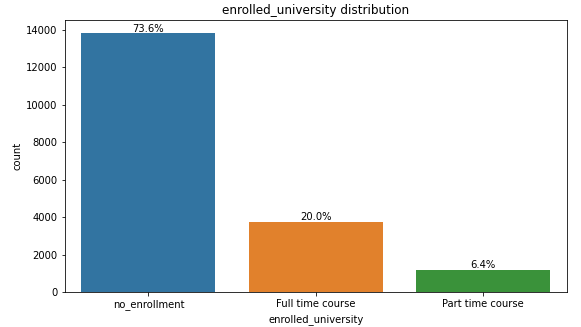
**Relevant Experience**

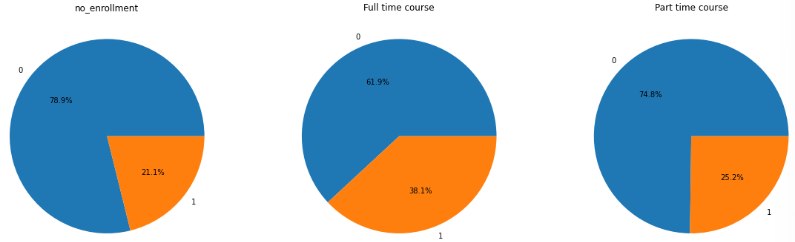
****

****

Dataset consists of more people that have relevant experience in the field (72%). The percentage of those who will not look for a new job is higher among the ones with relevant experience (around 79%) than the ones without (around 66%).

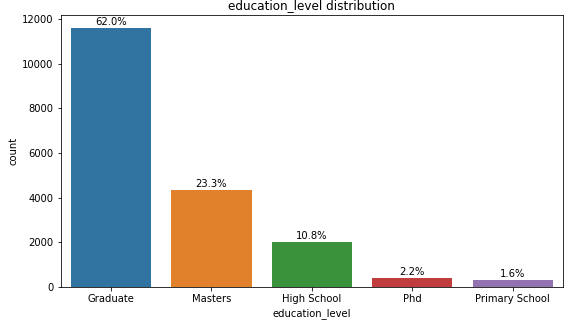
**Enrolled University**

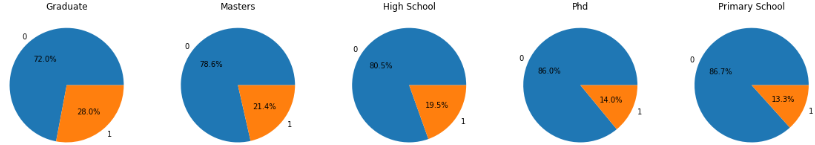




Most of the candidates are not attending the university and the majority of them are really interested in the job. There is a group of people who are enrolled full time and among them the target distribution is a little more balanced: 38% will look for a new job, 62% will not. This trend is understandable since usually a full time student doesn’t have enough time to both study and work or doesn’t want to. There is also a small group of part time students where 75% of them will not look for a new job.

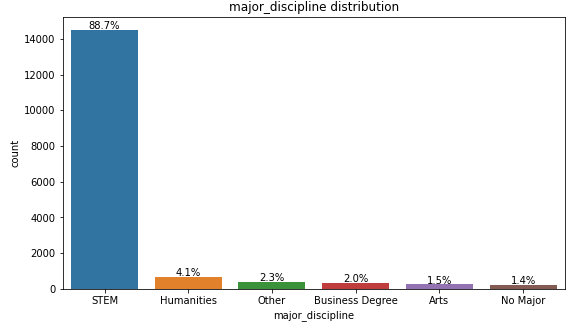
**Education level**

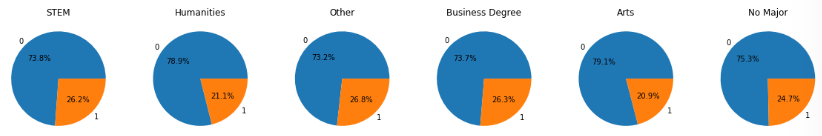




Most of the samples relate to people who have a bachelor or a master degree since they are candidates for a job that requires a high education level. There is a group of people from high school and two really small from phd and primary school. The percentage of the ones who are really interested in the job follows the opposite trend: it’s higher in groups with few samples and lower in groups with many samples.

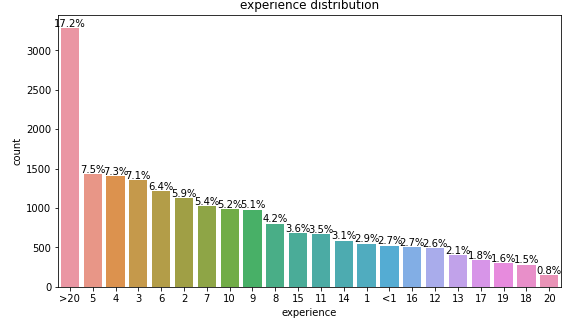
**Major discipline**

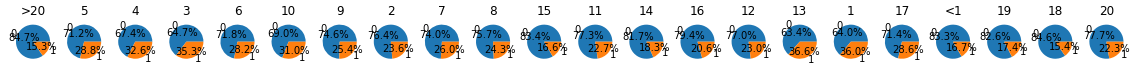




It’s clear that most of the candidates have a STEM discipline (Science, Technology, Engineering and Mathematics) as their major discipline. This result is understandable since the company we are talking about is active in Big Data and Data Science. The percentage of those who will look for a new job remains between 21% and 27% for all cases.

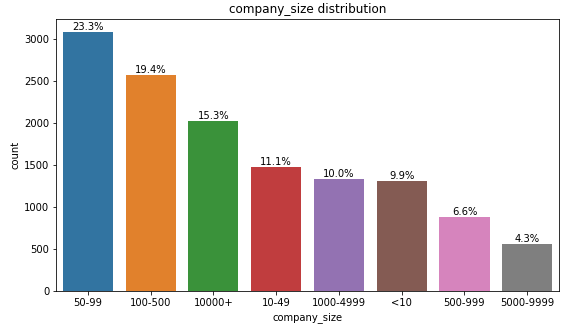
**Experience**

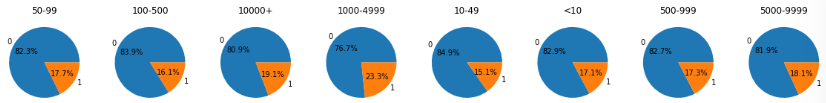




The distribution of candidates' total experience, expressed in the number of years, takes a gradual downward trend, except for the case with more than 20 years of experience. It doesn’t seem possible to deduce a specific order or criterion by which the several values are positioned in the plot. The percentage of those who will look for a new job goes from 15.3% (min value), that is related to people with more than 20 years of experience, to 36.6% (max value), that is related to the ones with 13 years.

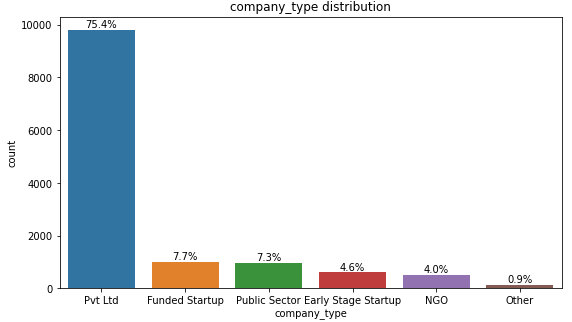
**Company size**

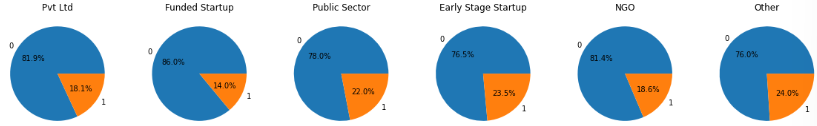




The distribution of company size values takes a gradual downward trend. Here we can not identify a specific value that is really preponderant: no value clearly imposes itself on the others, but the distribution is more homogeneous. The percentages of target values are quite similar for all cases.

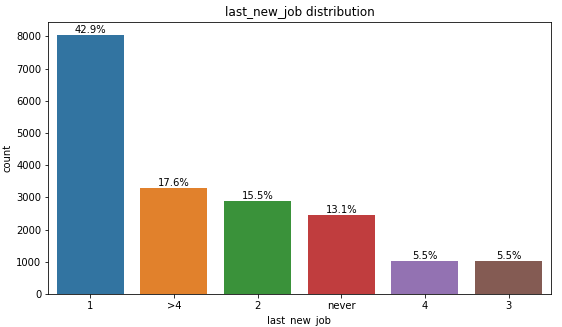
**Company type**

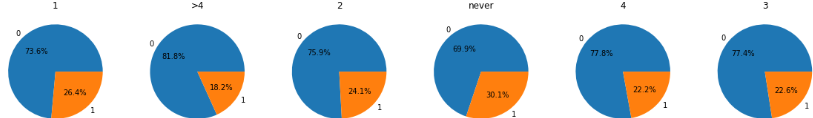




Around 75% of people are currently working in a private company, the remaining are quite equally distributed among the other options.

**Last new job**

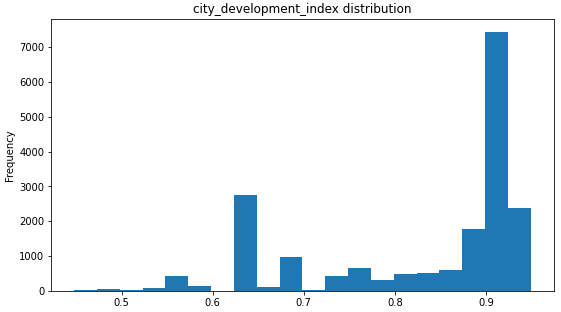
****

****

People who are working in the current company for 1 year cover almost half of the samples (43%). From the pie charts it’s interesting to note that as the difference in years between previous and current job increases, the percentage of those who will look for a new job after the training course decreases.

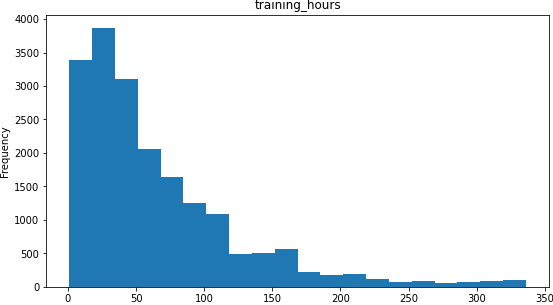
Now, let’s have a look at numerical features where hist plots are used for the analysis.

**City development index**

****

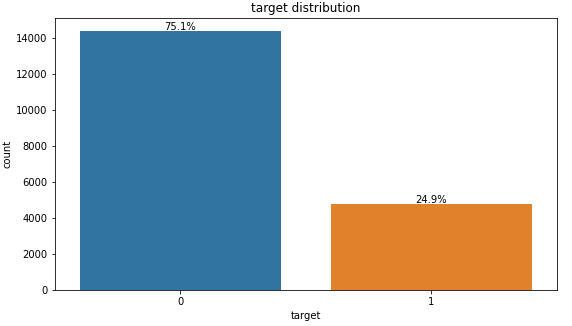
Dataset consists of more samples where the city development index is higher than 0.7

**Training hours**

****

Dataset consists of more samples where the number of training hours already completed is lower than 150.

Finally, let’s have a look at the target class distribution.



In all the previous pie charts the percentage of records with target value equals to 0 is bigger than the number of records with target value equals to 1. Dataset is clearly unbalanced: around 25% of data belong to class 1 and the remaining 75% belong to class 0. This problem will be solved in the data preprocessing section, before the training, in order to have a better prediction.

# **4. DATA PRE-PREPROCESSING**

Data preprocessing is the process of cleaning raw data and making it suitable for a machine learning model. It is an integral step since the quality of data and the useful information that can be derived from it directly affects the ability of the model to learn.

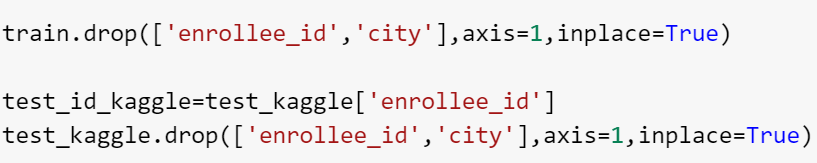
The concepts that will be covered in this section are:

1. feature removal
2. duplicate records
3. handling missing values
4. feature scaling
5. data sampling
6. feature encoding
7. feature correlation

## 4.1 FEATURE REMOVAL

As anticipated in the feature distribution section, two attributes are removed from the dataset since they contain useless or redundant information, becoming irrelevant for our task. These are:

* candidate’s ID
* city code, because the city development index attribute is enough to categorize cities.



## 4.1 DUPLICATE RECORDS

The second step of data cleaning is checking for duplicate records within the dataset:





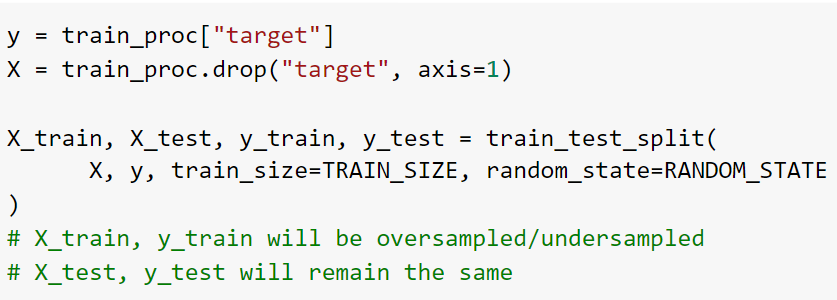
60 records are duplicates, therefore, they are removed from the dataset. In this way, every single remaining record differs from the others: no one of them is a duplicate anymore.

## 4.3 HANDLING MISSING VALUES

Before going into details, the dataset is divided into two subsets:

* training set, that consists of 80% of records and it’s used to train the model
* test set, that consists of the remaining 20% and it’s used to test the model just trained.

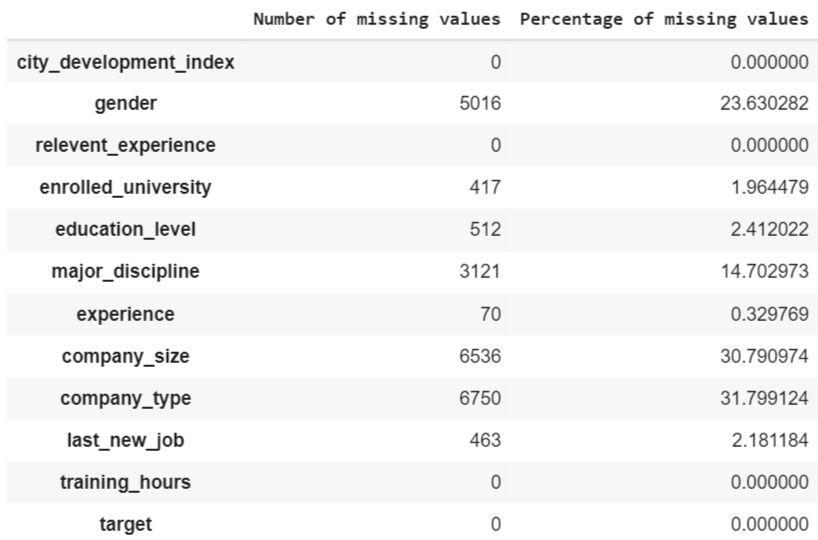
The idea is to apply data processing techniques separately on the training set and test set, and apply data sampling only on the first one while keeping the second unchanged. In this way, the models' performances will be computed over the same samples thus allowing their comparison.



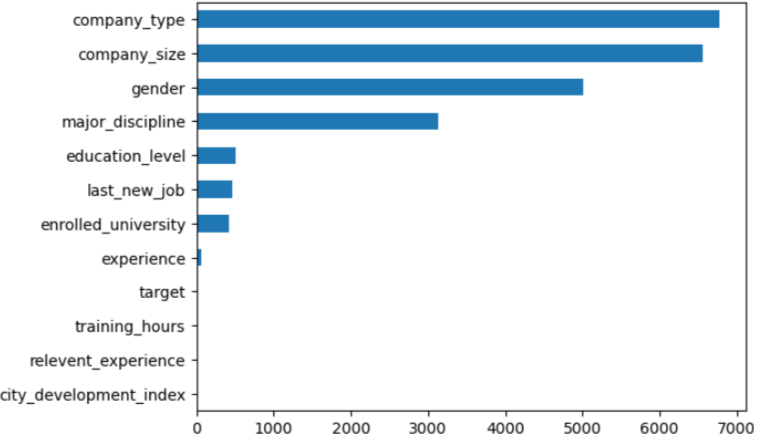
Missing value occurs when no data is stored for a variable in a given dataset. The main reasons for having missing values are:

* information is not collected
* attributes may not be applicable to all cases

Our dataset contains a lot of NaN: their number and percentage for each attribute is shown below.

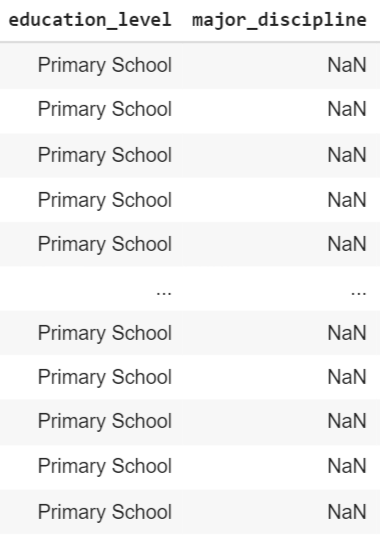


8 out of 14 variables have at least one missing value. They are gender, enrolled university, education level, major discipline, experience, company size, company type and last new job. For a better visualization we can use the following plot where variables are ordered by the higher number of NaN.

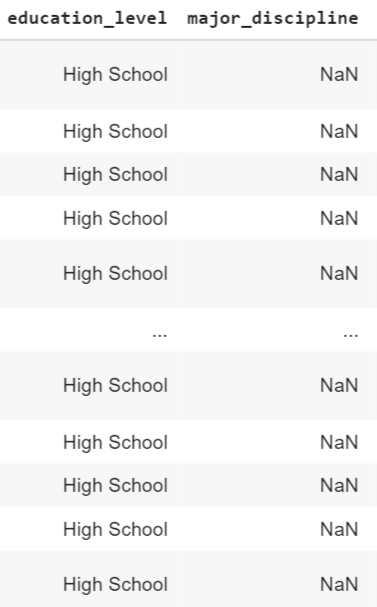


Let’s now check if there is a relation between attributes which forces that some values are not applicable thus generating a missing value. After this analysis it was found that:

1. When the education level attribute is equal to “Primary School”, a NaN is expected for the major discipline feature



1. When the education level attribute is equal to “High School”, a NaN is expected for the major discipline feature.



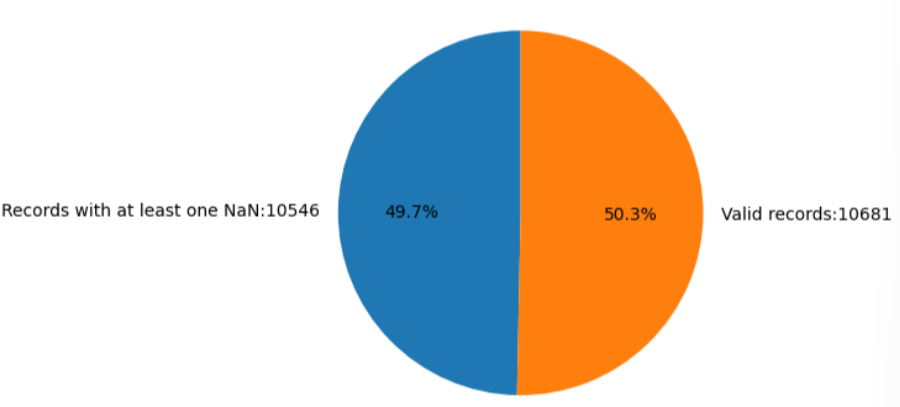
A major discipline is the academic discipline to which an undergraduate student formally commits, therefore it’s not applicable to students with a lower education level. For this reason, all these NaN values are replaced by “No major”. After this first replacement, the number of missing values for the major discipline feature decreased from 3121 to 542.

Furthermore, I tried to understand when candidates don’t fill in information about the current company just because they don’t work at the moment, but it was not possible to understand how to exactly identify them. Assumptions were reached, but nothing for sure.

Now we can proceed to the handling of the remaining missing values. Some techniques to do so are:

* eliminate records which contain at least one NaN
* estimate missing values using an imputer
* replace missing values with all possible values weighted by their probabilities

To understand which technique is best to use, let’s have a look at the distribution of NaN values within the dataset.



Since 50% of records in the dataset have at least one NaN (that is the number of missing values is high), eliminating them is not a good strategy. Instead, estimating them through an imputer or replacing them using probability distributions are better. Going into more details, an imputer is an estimator used to fill the missing values in a dataset. Several imputer exist, but in this work we are only focus on the SingleImputer from Sklearn that uses simple strategies:

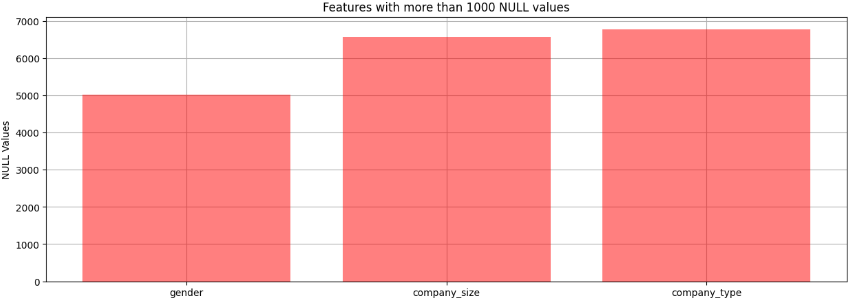
* null values of numerical features can be replaced by the mean, median or most frequent value computed along the corresponding column or using a constant value
* null values of categorical features can be replaced by the most frequent category along the corresponding column or using a constant value, since it’s not possible to compute mean and median.

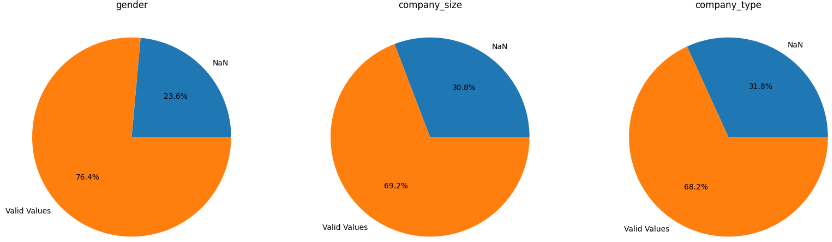
Before applying one of these techniques, two main cases can be identified:

1. features with a high number of missing values but in a limited way compared to the amount of data
2. features with a small number of missing values

*CASE 1)*

Gender, Company size and Company type belong to this group since they have more than 5000 null values.



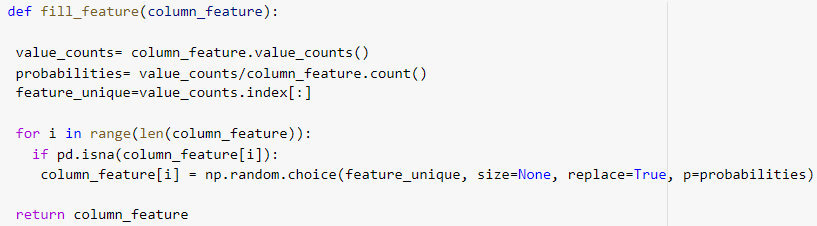


Let’s now try to understand which strategy is better to use. Regarding the gender attribute, in the feature distribution section we saw that 90.2% of the samples are Male, only 8.5% are female and just 1.3% belong to another category. Therefore, in this case replacing all the 5016 NaN with the most frequent value (Male) through the imputer or replacing them with all the possible values weighted by their probabilities is quite similar: in the first case we are sure that all nulls will be replaced by Male, in the second one we know that nulls will be replaced by Male with a probability of 90%, by Female with a probability of 8.5% and so on. Regarding the company type attribute, in the feature distribution section we saw that around 75% of the candidates currently work in a private company and the remaining are distributed among the other 5 options. Therefore, in this case choosing between the two techniques is quite different, since the number of missing values is very high (6750). Furthermore, if we consider the company size attribute, the difference becomes higher since records are more equally distributed among the several options:

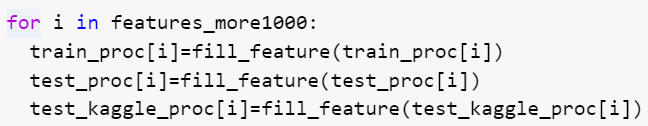
* 23.3% for 50-90
* 19.4% for 100-500
* 15.3% for 100000+
* 11.1% for 10-49
* 10% for 1000-4999
* 9.9% for <10
* 6.6% for 500-999
* 4.3% for 5000-9999

Thus, replacing all the 6536 NaN with the most frequent value (50-90) or replacing them with all possible values is very different. The first method can lead to a heavy imbalance of its feature distribution.

After all these considerations, I decided to replace these NaN with all the possible values weighted by their probabilities. To do so, a function that fills missing values through an extraction with reinsertion based on the probability distribution of the several values was defined.

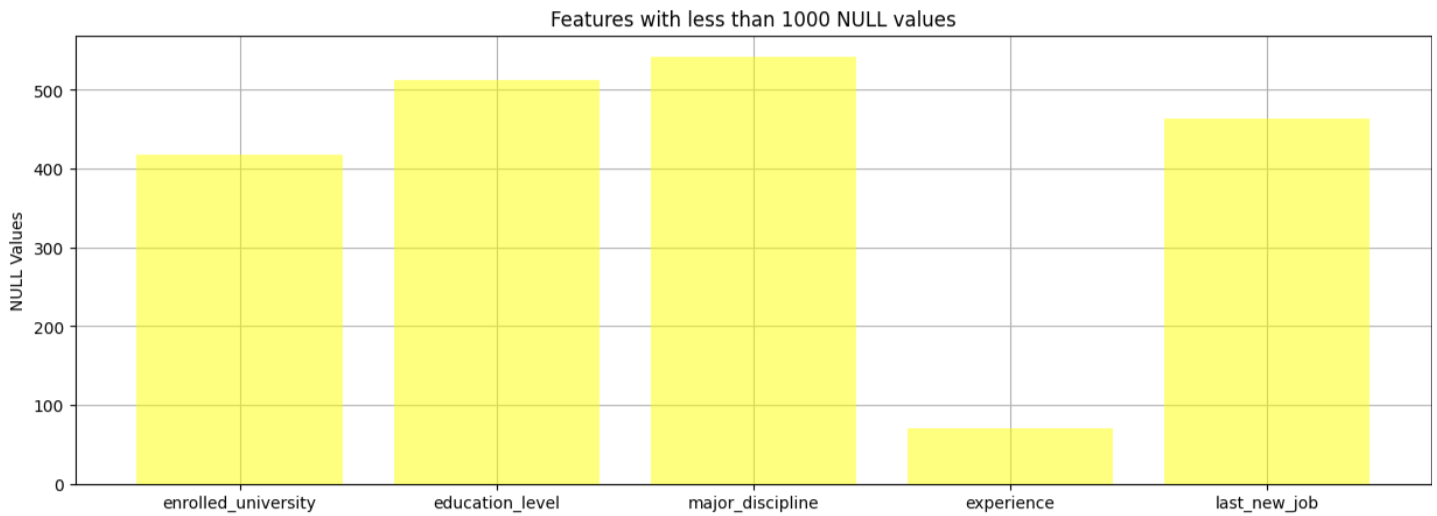


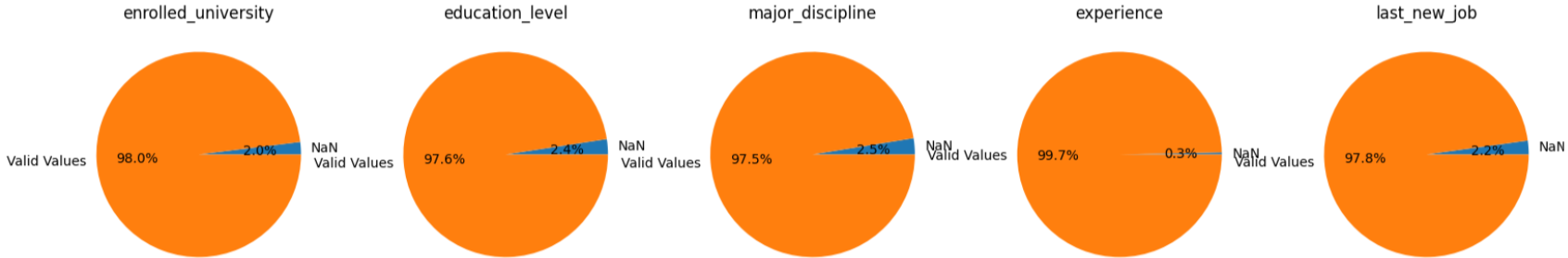
Then, this function was applied to the features with a high number of missing values we were talking about before.



*CASE 2)*

Enrolled University, Education Level, Major Discipline, Experience and Last New Job belong to this group since they have less than 600 null values.

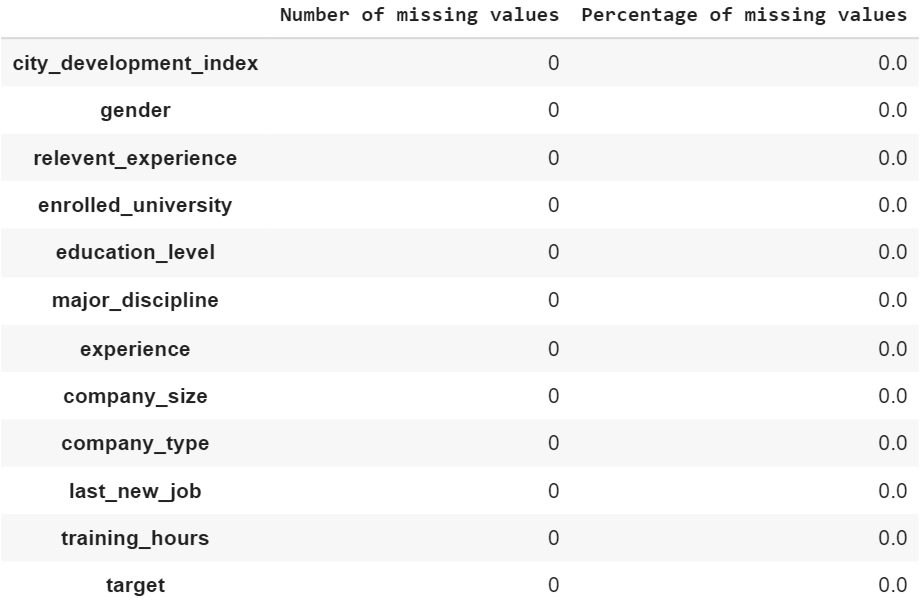




In this case the number of missing values is very small compared to the entire amount of data. Therefore, we can skip the consideration made before and directly apply the imputer, replacing the nulls with the most frequent value in each column.



Once the necessary treatments have been applied, let’s recalculate that the number of missing values for each feature is equal to 0, in order to verify they are no longer present in the dataset.

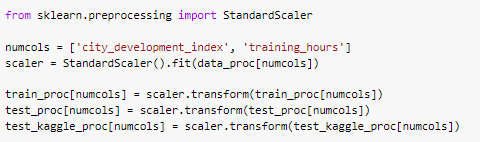


## 

## 4.4 FEATURE SCALING

Feature scaling is a method to transform numeric features to a standard range so that the performance of machine learning algorithms improves. The chosen scale is not important but each of the features should be on the same scale.

In the feature distribution section we saw that the city development index ranges from 0.448 to 0.949, whereas the training hours attribute is from 1 to 336. The machine learning algorithm thinks that the feature with higher range values is most important while predicting the output and tends to ignore the feature with small range values. To avoid it, the range of all features are scaled so that each feature contributes proportionately and model performance improves. Therefore, a StandardScaler is applied to these numerical attributes.



Rescaling the distribution does not affect the shape.

## 4.5 DATA SAMPLING

As already anticipated during the feature distribution analysis, the dataset is unbalanced: both the training set and testing have more observations on one specific class than the other. In particular, 75% of records belong to class 0, which is called the majority class, and 25% of records belong to class 1, which is called the minority class.



The main problem with an imbalanced dataset is that classes are not represented equally and the classifier may get biased towards prediction: it could not be accurate enough to predict both the majority and minority class. The model should not be biased to detect only the majority class but should give equal weight or importance towards the minority too.

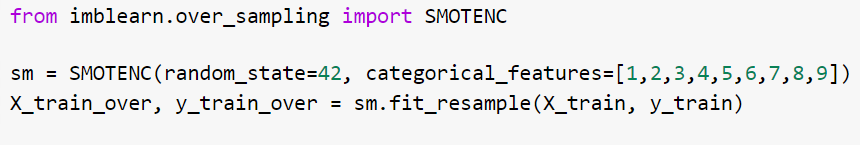
Among the techniques which can deal with this problem are the oversampling and undersampling. In general, undersampling can be implemented when the dataset is sufficient, however, in most cases oversampling is preferred as the first can undergo loss of important data. Since the two techniques are compatible with our dataset, I decided to implement both, to understand which one leads to the best results. To do so, we can easily benefit from the imblearn package in python to resample.

4.5.1 OVERSAMPLING

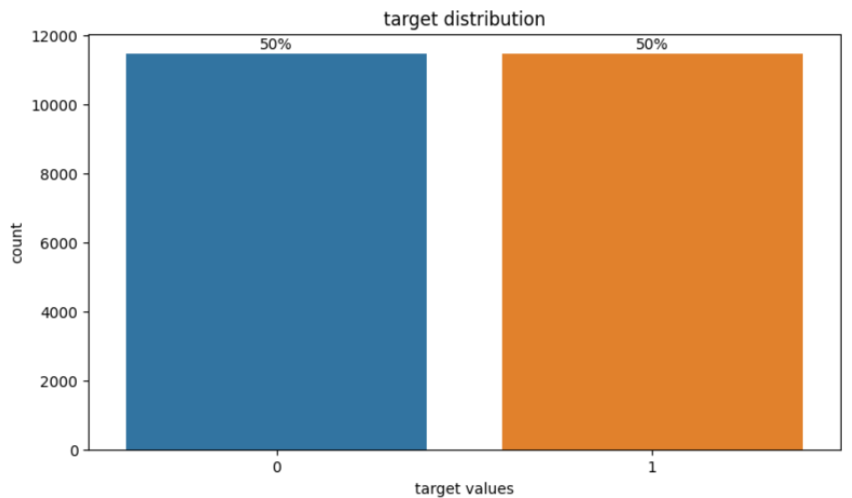
This technique increases the minority class adding new minority elements to the dataset. There are two ways to implement it:

1. RandomOverSampler, that generates new samples by randomly sampling with replacement the current available instances;
2. SMOTE (Synthetic Minority Oversampling Technique), that creates new synthetic samples based on the original instances, increasing both the size of the training set and the variety of examples.

It’s interesting to notice that there are situations where oversampling with SMOTE is better not to apply as more attention is needed, such as with dataset about human health issues. In our dataset this problem doesn’t sussist and I decided to use this approach since it leads to better performances. When the dataset contains both numerical and categorical features SMOTENC is used.

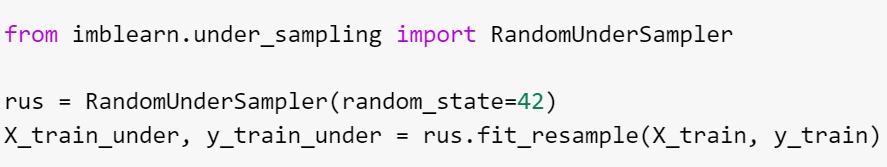


Thanks to the oversampling, the number of records with target value equal to 1 has increased from 3800 to 11478, now balancing the number of samples with the other value.

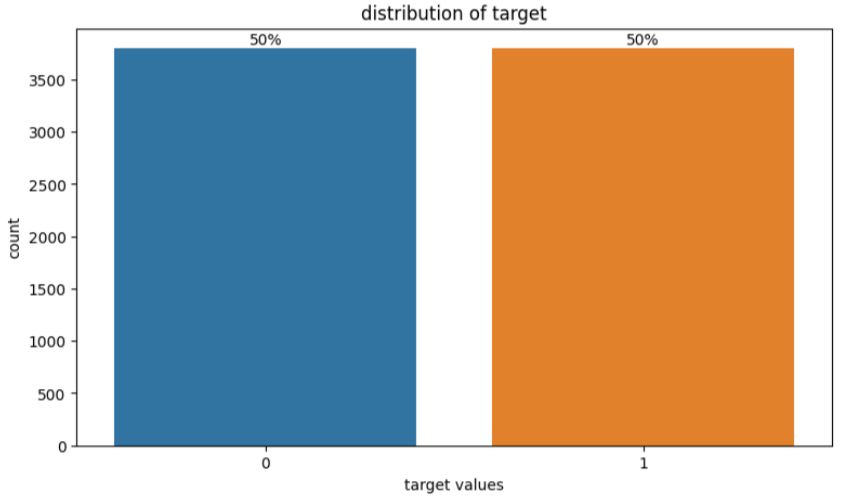


4.5.2 UNDERSAMPLING

Instead of increasing the minority class, this method reduces the majority one removing some instances from this group. Among the several available in imblearn.under\_sampling, I implemented the RandomUnderSampler approach. It provides a fast and easy way to balance the data by randomly selecting a subset of data for the target classes.



Thanks to the undersampling, the number of records with target value equal to 0 has decreased from 11478 to 3800, now balancing the number of samples with the other value.



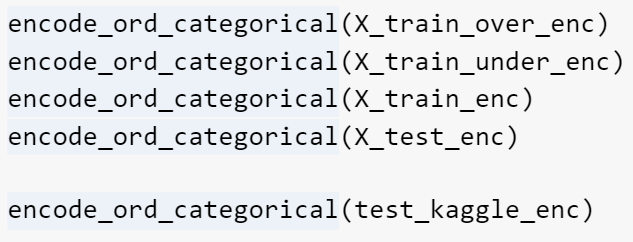
## 4.6 FEATURE ENCODING

Most machine learning models can only work with numerical values. For this reason, it is necessary to transform the categoricals into numericals. Label encoding is one of the most basic types of categorical feature encoding methods since it simply assigns a number to each unique value in a feature. This technique was applied differently depending on the type of categorical feature.

4.6.1 ORDINAL FEATURES

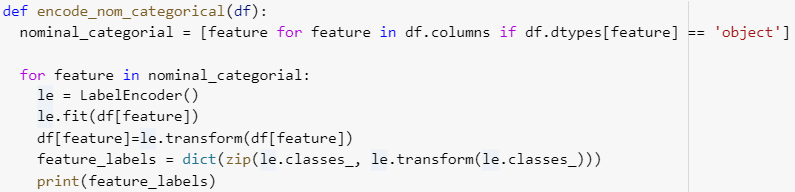
Ordinal features are categorical features characterized by a clear ordering. For example, the education level attribute belongs to this group since there is a progressive order between its several values: primary school, high school, graduate, master and phd. To maintain this relationship, the function ‘encode\_ord\_categorical’ was set up to assign the lower number to the lower education level and the higher number to the higher education level, using a dictionary mapping. The same reasoning has been applied to all ordinal features both in the undersampled and oversampled datasets, as well as in the test set for the kaggle submission.

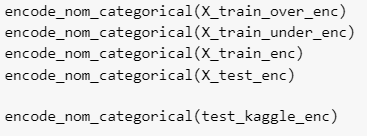




4.6.2 NOMINAL FEATURES

Nominal variables are categorical features without an intrinsic ordering to its categories. For example the gender attribute belongs to this group since there isn’t an order between its three values: Male, Female and Other. Therefore, the numbers assigned for the options are random and their labels have no actual meaning. They are simple to deal with. To do so, the function ‘encode\_nom\_categorial’ was set up using the LabelEncoder from sklearn.preprocessing.





## 4.7 FEATURE CORRELATION

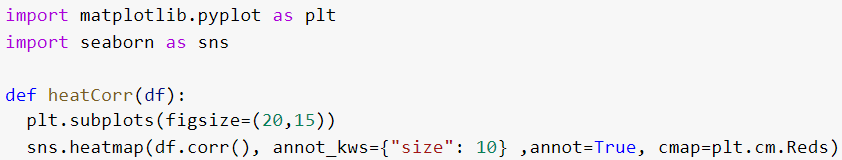
Feature correlation is a measure of the dependence degree between two random variables. It can be positive if the increase of one variable is related to the increase of another one, or negative if the increase of one variable is related to the decrease of the other one.

In this section the idea is to analyze the feature correlation both before and after the application of all data preprocessing techniques, to verify that correlations remain almost the same. In particular we want to check that the handling of missing values and data sampling have not generated correlations too strong or weak. To do so, correlation heatmaps are used to understand which variables are related to each other and the strength of their relationship, that is indicated by the values in the cells. These values can be interpreted as follows:

* Correlation coefficients whose magnitudes are between 0.9 and 1 indicate variables which can be considered very strongly or perfectly correlated;
* Correlation coefficients whose magnitudes are between 0.7 and 0.9 indicate variables which can be considered strongly correlated;
* Correlation coefficients whose magnitudes are between 0.5 and 0.7 indicate variables which can be considered moderately correlated;
* Correlation coefficients whose magnitudes are between 0.3 and 0.5 indicate variables which can be considered weakly correlated;
* Correlation coefficients whose magnitudes are between 0 and 0.3 indicate variables which can be considered very weakly correlated or without correlation.

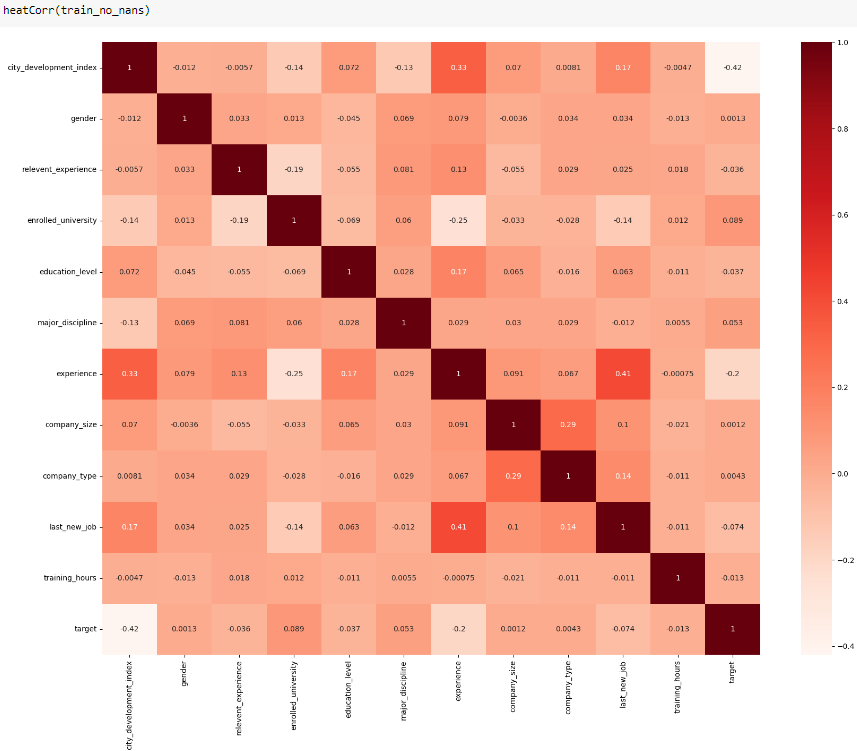
The feature correlation analysis is also important to identify features with a coefficient bigger than 0.7: when it happens the two features are strongly dependent and one of them should be removed from the dataset because it is redundant.

In Python, the Seaborn library allows us to create a basic correlation matrix. Since Seaborn had been built on the Matplotlib data visualization library and it is often easier to use the two in combination, the import of Matplotlib.pyplot is necessary too.



4.6.1 INITIAL CORRELATION

To analyze the feature correlation before the application of data preprocessing techniques, all records of the original dataset containing missing values are removed using the dropna() function, and after that, categorical features are encoded (see Section 4.5).

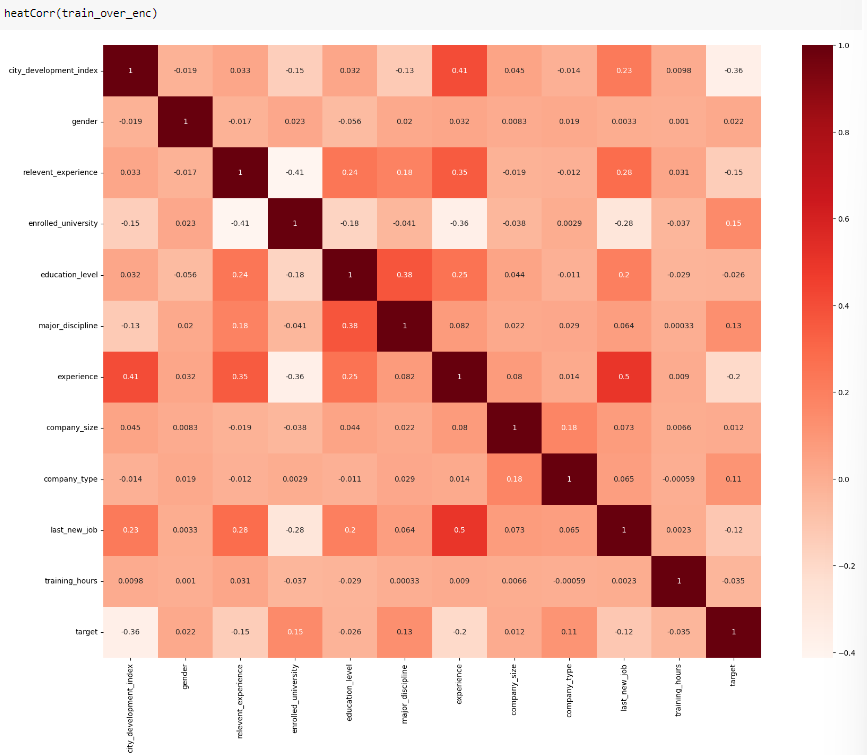


The initial correlation matrix shows that:

* Correlations between target class and the other features are very weak, except for the couple (city\_development\_index, target) where the coefficient is equal to -0.42 that indicates a weak/moderate correlation. The negative value means that if the city development index increases the target value tends to decrease and viceversa;
* Almost all correlations between input attributes are very weak, except for the couples (city\_development\_index, experience) and (last\_new\_job, experience) where the coefficients are respectively equal to 0.33 and 0.41;
* There is no couple with a coefficient bigger than 0.7 meaning a strong correlation.

4.6.2 CORRELATION AFTER DATA PREPROCESSING

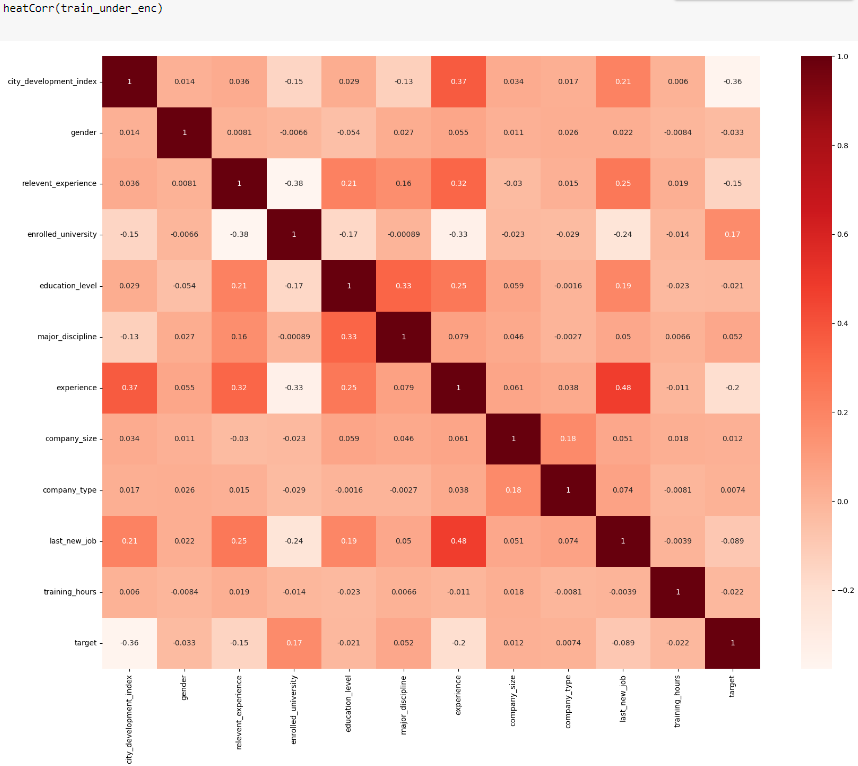
*CASE 1) Oversampling*



The correlation matrix after the handling of missing values and the oversampling shows that:

* Correlations between target class and the other features are very weak, except for the couple (city\_development\_index, target) where the coefficient is equal to -0.36 that indicates a weak correlation;
* Almost all correlations between input attributes are very weak, but in this case there are six couples with a coefficient between 0.3 and 0.5. Among them the highest value is 0.5 associated to the couple (last\_new\_job, experience) that indicates a moderate correlation;
* There is no couple with a coefficient bigger than 0.7.

*CASE 2) Undersampling*



The correlation matrix after the handling of missing values and the undersampling shows that:

* Correlations between target class and the other features are very weak, except for the couple (city\_development\_index, target) where the coefficient is equal to -0.36 that indicates a weak correlation;
* Almost all correlations between input attributes are very weak, but in this case there are six couples with a coefficient between 0.3 and 0.5. Among them the highest value is 0.48 associated to the couple (last\_new\_job, experience) that indicates a weak/moderate correlation;
* There is no couple with a coefficient bigger than 0.7.

In conclusion, after the handling of missing values and the implementation of a data sampling technique, the correlation coefficients between some features became slightly heavier. However, no correlation with a magnitude higher than 0.7 was introduced, thus, no significant difference can be identified between the before and after cases. Most features remain weakly or very weakly correlated.

# **5. TRAINING**

In machine learning, training is a process in which a machine learning algorithm is fed by training data to learn how to make predictions for a given task. In particular, in this work we want a model that can predict whether a candidate really wants to work for the company after the training courses or will look for a new job.

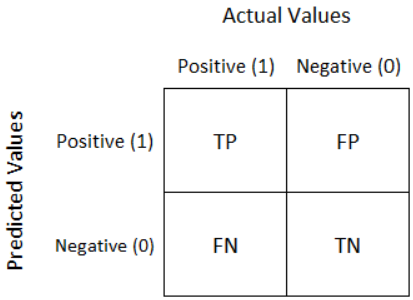
Data available will be used to train and test several machine learning models, each characterized by a set of parameters that can be tuned to optimize the classification. The traditional way of performing parameter tuning is the Grid Search that basically explores a range of parameters and finds the best combination. During this process the cross validation technique is used to prevent the overfitting. It makes use of stratified folds to ensure that splits will not be completely random but the ratio between target classes will be the same in each fold as it is in the full dataset.



Once the best estimator configuration is found through the parameter tuning, the fit() method will be used to fit the model to the input training instances while the predict() method to perform predictions on the testing instances, based on the parameters just learned. Then, each model will be evaluated on a complete set of metrics.

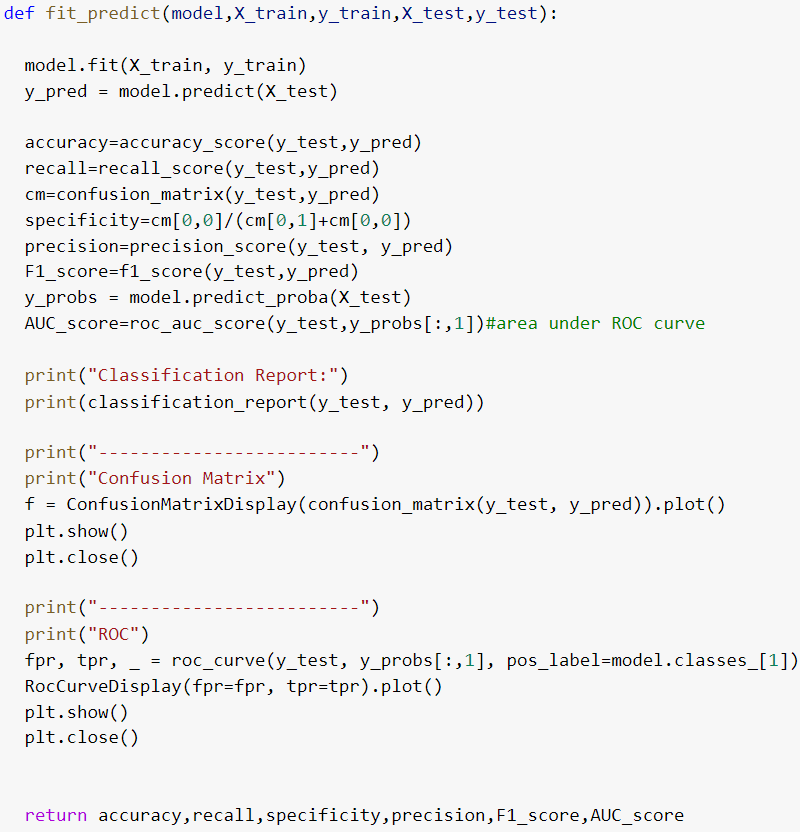
* *Accuracy = (TP+TN)/N,* that is the number of correct predictions out of the number of total predictions
* *Recall = TP/(TP+FN),* that summarizes how well the positive class is predicted;
* *Specificity = TN/(TN+FP),* that summarizes how well the negative class is predicted;
* *Precision = TP/(TP+FP),* that summarizes the fraction of examples assigned to the positive class that belongs to the positive class;
* *F1 Score = (2\*precision\*recall)/(precision+recall) = 2\*TP/(2\*TP+FN+FP),* that keeps the balance between precision and recall;
* *AUC Score = area under ROC curve.*

In the previous formulas, TP is the number of true positives, TN of true negatives, FP of false positives and FN of false negatives. These values are obtained by confusion matrices.



Although widely used, accuracy can be inappropriate for imbalanced classification because a high value is achievable by a model that predicts well only the majority class. Also ROC Curve and ROC AUC are generally effective, but can be optimistic under a severe class imbalance, especially when the number of examples in the minority class is small. Instead, other metrics such as recall, specificity and precision may be more useful as they focus on one class.

All the previous computation are executed inside a prediction function, thus defined:



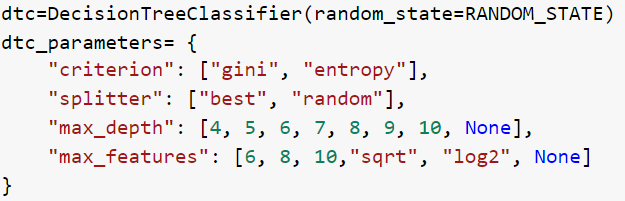
Several machine learning models such as Decision Tree, Random Forest, Extra Trees, XGBoost, SVM and Neural Networks were selected for the training, to find out which gives best results.

## 5.1 DECISION TREE

After reading the documentation provided by Sklearn, the set of parameters chosen for the parameter tuning process is:

* criterion, that is the function used to measure the quality of a split;
* splitter, that is the strategy used to choose the split at each node;
* max\_depth, that is the maximum depth of the tree;
* max\_features, that is the number of features to consider when looking for the best split.

Instead, the others are set by default.



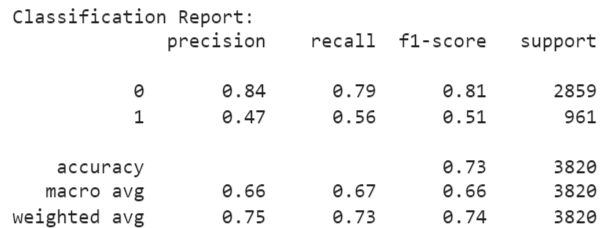
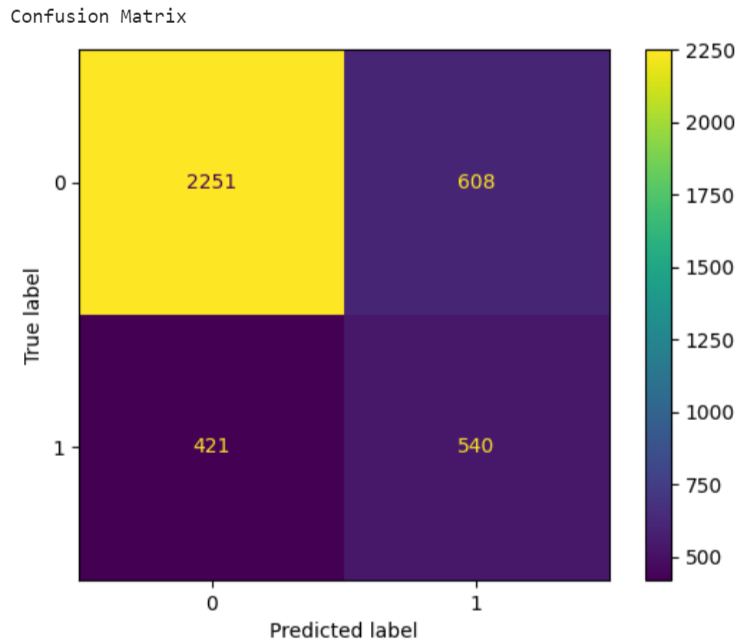
The set of values for some parameters was chosen based on the structure of the dataset, in particular, on the number of features.

## 5.1.1 OVERSAMPLED DATASET

The best score is given by this combination of parameters:

* criterion= gini
* splitter= best
* max\_depth= 10
* max\_features= None

The DT classifier with this parameter configuration was trained, obtaining the following performances on the test set:

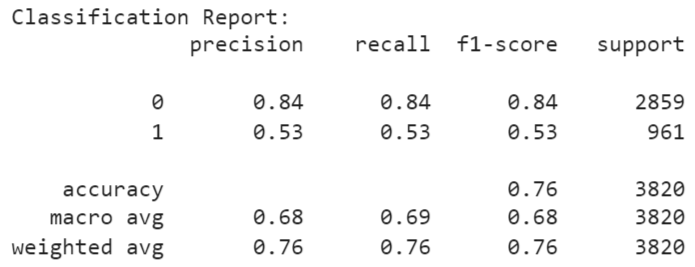
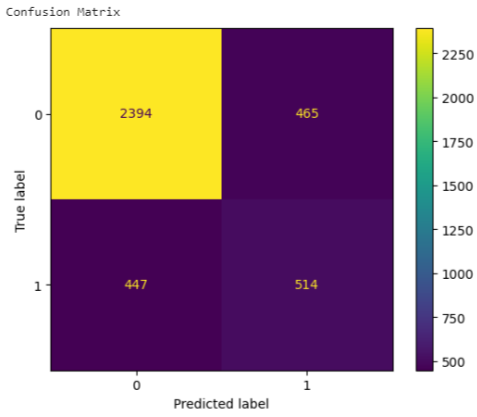


## 5.1.2 UNDERSAMPLED DATASET

The best score is given by this combination of parameters:

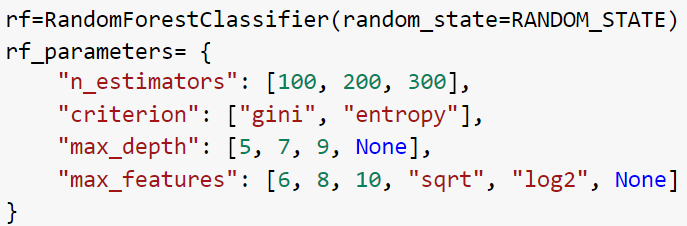
* criterion= entropy
* splitter= best
* max\_depth= 4
* max\_features= 6

The DT classifier with this parameter configuration was trained, obtaining the following performances on the test set:



## 5.2 RANDOM FOREST

After reading the Sklearn documentation, the set of parameters chosen for the parameter tuning process is similar to the one chosen for the Decision Tree. This is because Random Forest is a meta estimator that fits a number of Decision Tree classifiers on various subsamples of the dataset and uses averaging to improve the predictive accuracy and control overfitting. The only new parameter introduced here is n\_estimators that represent the number of trees in the forest.



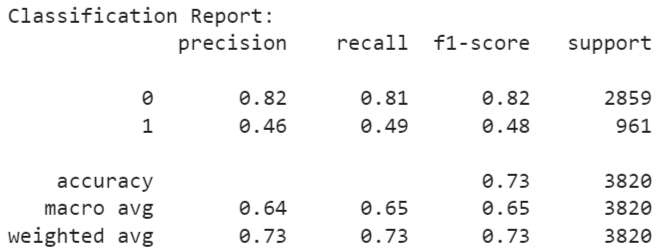
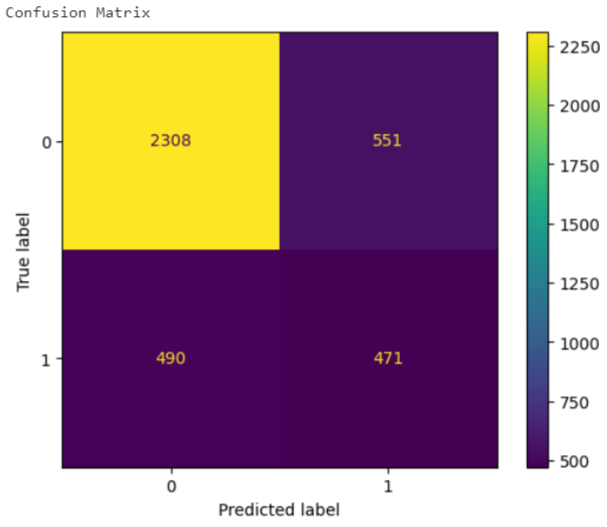
Instead, the others are set by default.

## 5.2.1 OVERSAMPLED DATASET

The best score is given by the following combination of parameters:

* n\_estimator= 200
* criterion= entropy
* max\_depth= None
* max\_features= 6

The RF classifier with this parameter configuration was trained, obtaining the following performances on the test set:

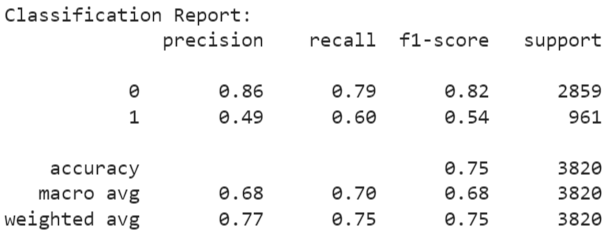
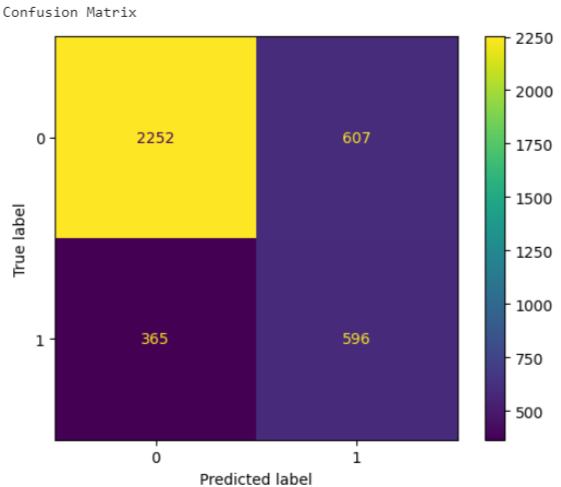


## 5.2.2 UNDERSAMPLED DATASET

The best score is given by the following combination of parameters:

* n\_estimator= 300
* criterion= gini
* max\_depth= 9
* max\_features= sqrt

The RF classifier with this parameter configuration was trained, obtaining the following performances on the test set:



## 5.3 EXTRA TREES

After reading the Sklearn documentation, the set of parameters chosen for the parameter tuning process is equal to the one chosen for the Random Forest. This is because Extra Trees fits a number of randomized decision tree classifiers on various subsamples of the dataset and uses averaging, like random forest does, but it introduces two main differences:

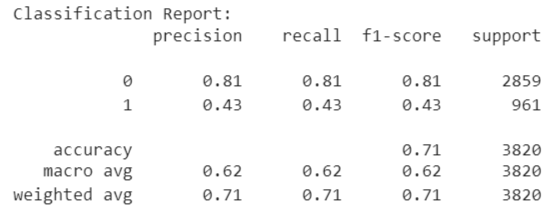
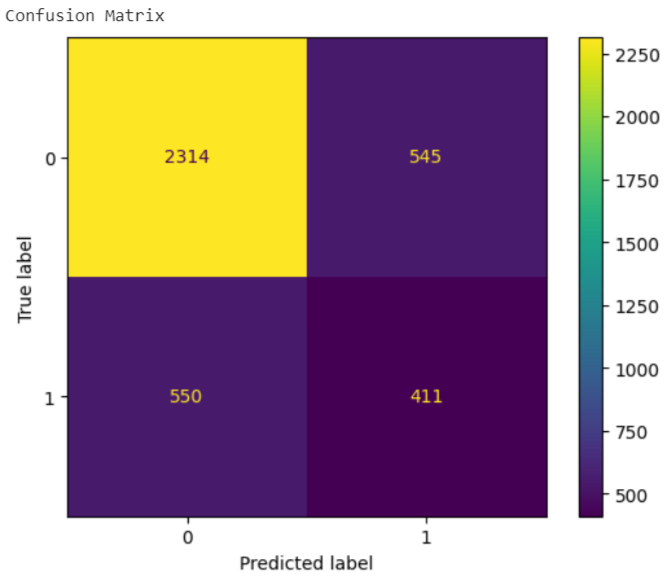
1. Bootstrap parameter is equal to False by default, thus it performs a sampling without reinsertion (instead in RF in True by default);
2. Split is random (instead, RF chooses the best split).

## 5.3.1 OVERSAMPLED DATASET

The best score is given by the following combination of parameters:

* n\_estimator= 200
* criterion= entorpy
* max\_depth= None
* max\_features= 8

The ET classifier with this parameter configuration was trained, obtaining the following performances on the test set:

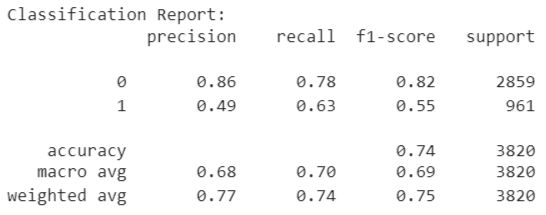
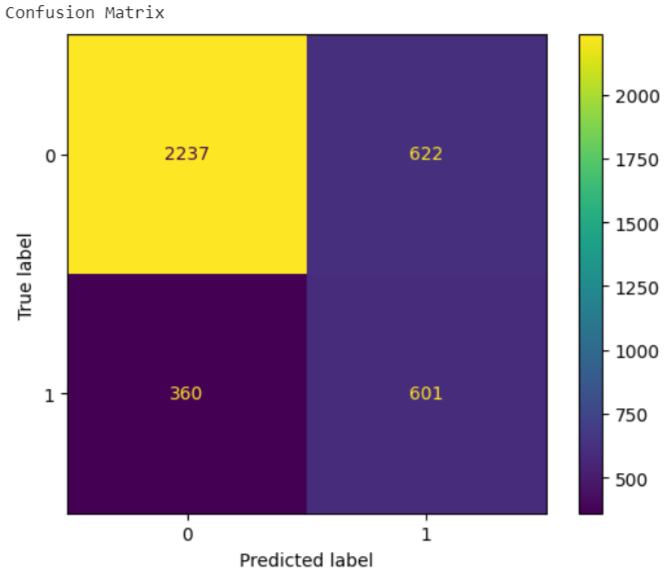


## 5.3.2 UNDERSAMPLED DATASET

The best score is given by the following combination of parameters:

* n\_estimator= 200
* criterion= gini
* max\_depth= 9
* max\_features= 10

The ET classifier with this parameter configuration has been trained, obtaining the following performances on the test set:

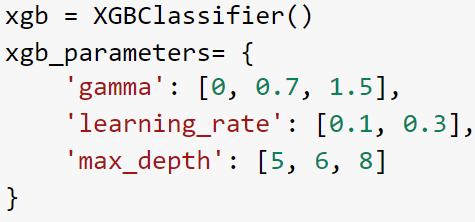


## 5.4 XGBOOST

The Extreme Gradient Boosting is a decision tree based machine learning algorithm which uses a process called boosting to help improve performances. Since its introduction, it has become one of the most effective models and regularly produces results that outperform most other classifiers. After reading the documentation provided by the xgboost library, the set of parameters chosen for the parameter tuning process is:

* gamma, that is the minimum loss reduction required to make a further partition on a leaf node of the tree;
* learning\_rate, also called eta, that is the step size shrinkage used in update to prevent overfitting ;
* max\_depth.

Instead, the others are set by default.

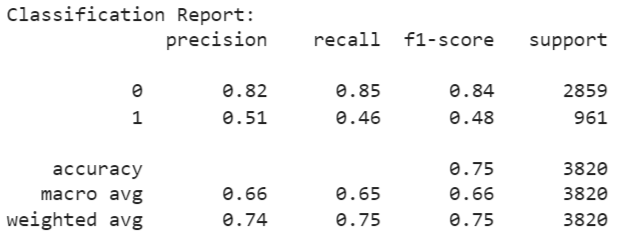
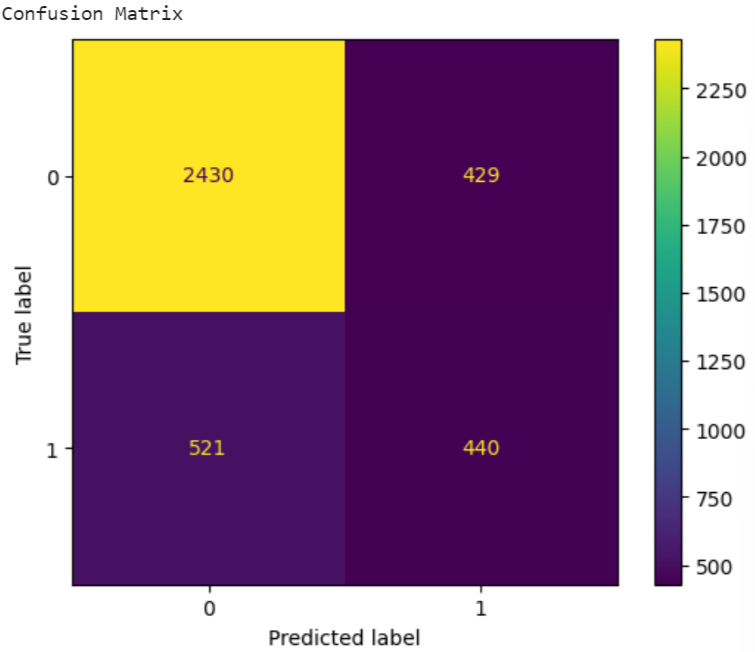


## 5.4.1 OVERSAMPLED DATASET

The best score is given by the following combination of parameters:

* gamma= 0
* learning\_rate= 0.3
* max\_depth= 8

The XGB classifier with this parameter configuration was trained, obtaining the following performances on the test set:

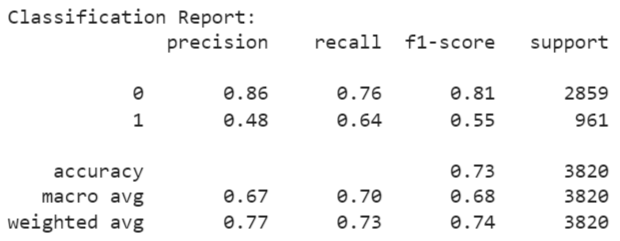
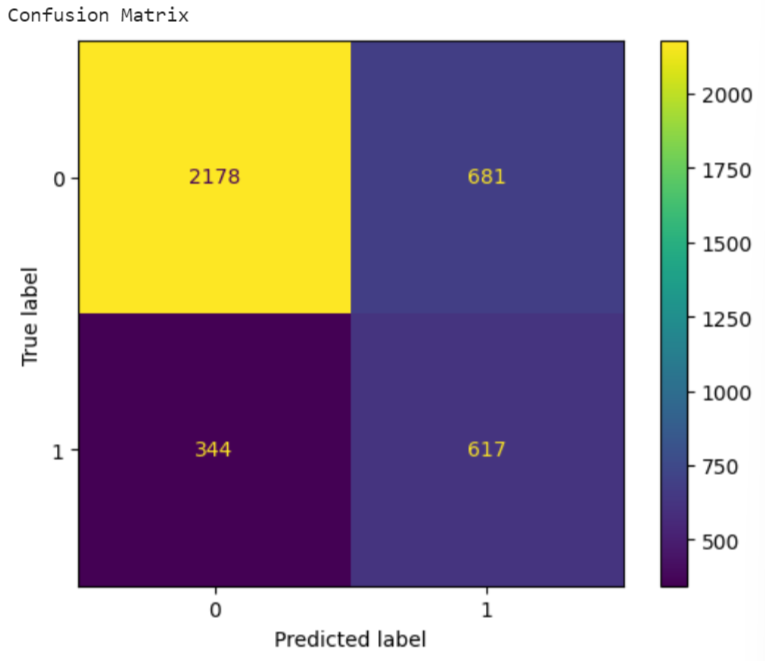


## 5.4.2 UNDERSAMPLED DATASET

The best score is given by the following combination of parameters:

* gamma= 1.5
* learning\_rate= 0.1
* max\_depth= 5

The XGB classifier with this parameter configuration was trained, obtaining the following performances on the test set:

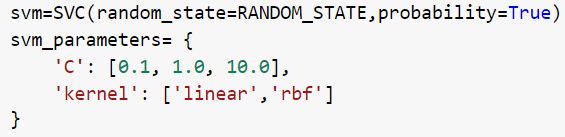


## 5.5 SVM

Support Vector Classifier is a supervised machine learning algorithm which works by mapping data points to a high-dimensional space and then finding the optimal hyperplane that divides data into two classes. Kernels are used to make non-separable data into separable data. After reading the Sklearn documentation, the set of parameters chosen for the parameter tuning process is:

* C, that is the regularization parameter;
* kernel, that specifies the kernel type to be used in the algorithm.

Instead, the others are set by default.



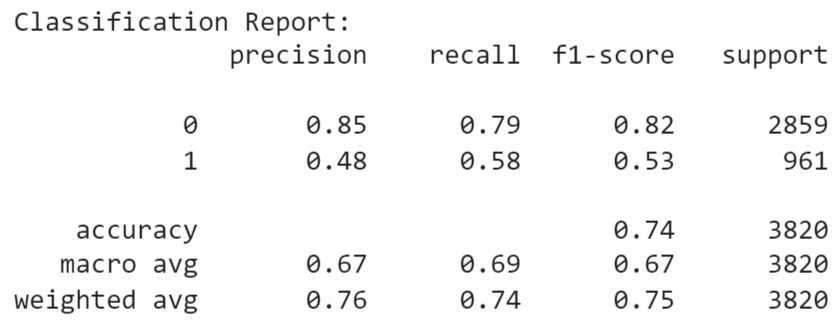
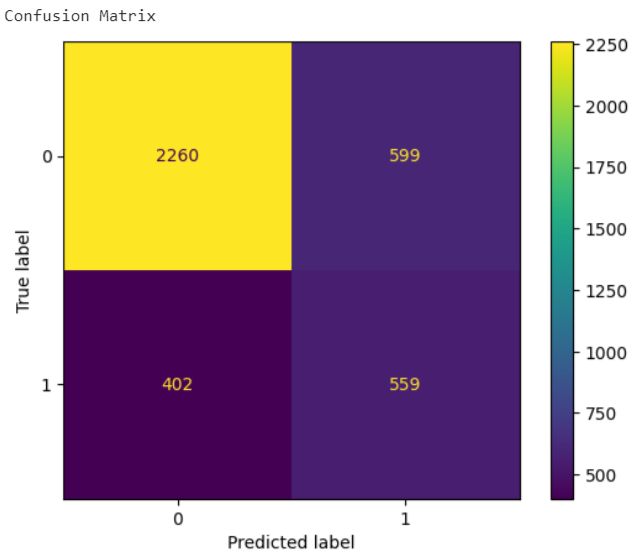
## 

## 5.5.1 OVERSAMPLED DATASET

The best score is given by the following combination of parameters:

* C= 10.0
* kernel= rbf

The SVM with this parameter configuration was trained, obtaining the following performances on the test set:

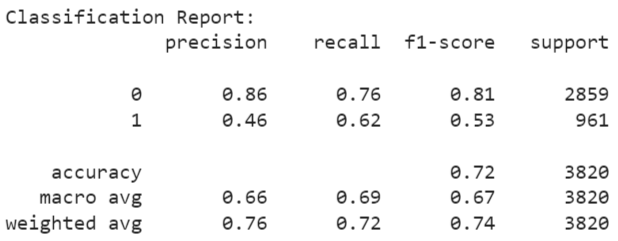
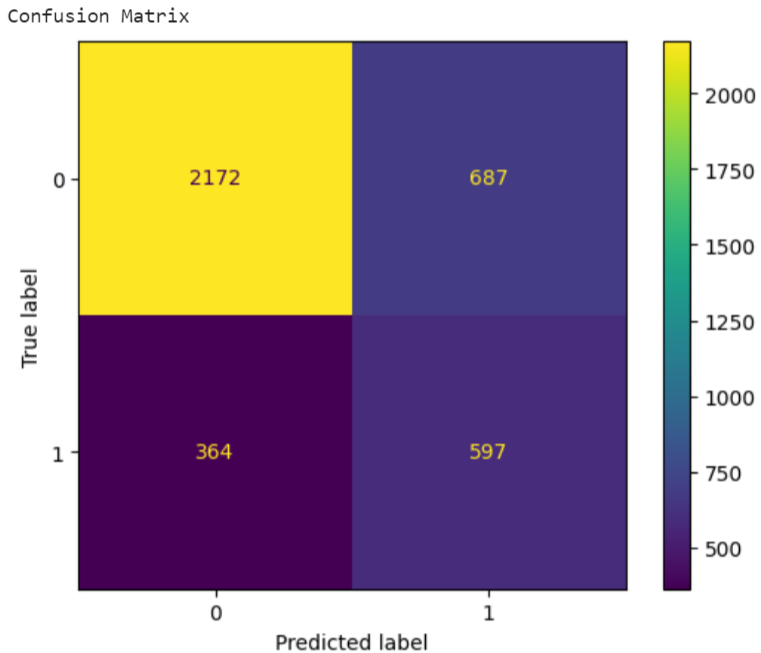


## 5.5.2 UNDERSAMPLED DATASET

The best score is given by the following combination of parameters:

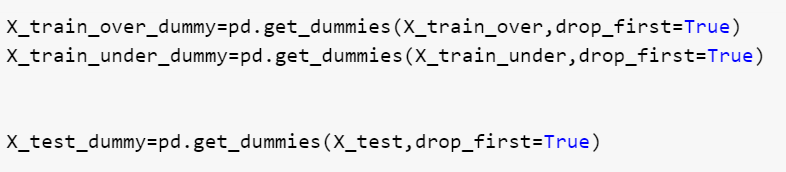
* C= 10.0
* kernel= rbf

The SVM with this parameter configuration was trained, obtaining the following performances on the test set:



## 5.6 NEURAL NETWORKS

The feature encoding previously applied to data, that consists of assigning an integer to each possible category, has the disadvantage that the numeric values can be misunderstood by some algorithms as the higher-level information is lost in translation. This issue can be addressed by another approach called One-Hot Encoding, that is the most recommended with Neural Networks. This technique creates a new set of dummy variables that is equal to the number of different categories for each feature. For example, the categorical variable gender has three categories called ‘Male’, ‘Female’ and ‘Other’, therefore three dummy variables are necessary to encode it. Each dummy variable takes value 0 or 1 to indicate the exclusion or inclusion of a category. However, this approach increases dimensionality making training slower and more complex. To alleviate this effect, we can use the Dummy Encoding technique that instead of creating a number of dummy variables that is equal to the number of categories k for each feature, it creates k-1 of them. Thus, to encode the same gender variable with three categories, only two dummy variables are now necessary. This approach can be implemented in Pandas by using its get\_dummies function.



After that, the feature number has been increased to 54.

To easily implement a neural network, Keras was used. It’s a deep learning API written in Python, running on the top of the machine learning platform Tensorflow.

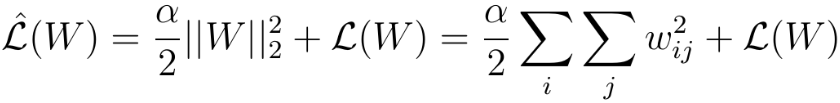
## 5.6.1 OVERSAMPLING

The Sequential constructor was initialized and four layers implemented:

* The Input layer with 54 neurons, since it is the number of features in training set, plus 1 for the bias;
* First hidden layer with 50 neurons
* Second hidden layer with 30 neurons
* The output layer with 1 neuron and the sigmoid activation function for the binary classification.

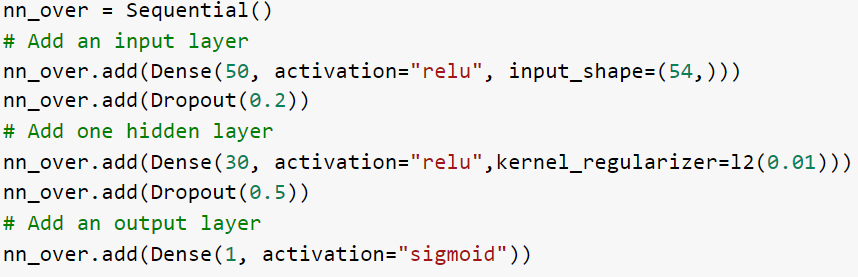
This configuration leads to a number of trainable parameters that is equal to 4311. Binary Cross Entropy was chosen as loss function, Adam as optimizer and 100 as the number of epochs.

Furthermore, there is another aspect to take into consideration: overfitting. Overfitting happens when the model learns too many details from the training set while its performance is poor on unseen samples: it fails to generalize the features or patterns in the training set. There are several techniques to prevent this issue. One of the most common is L2 regularization that updates the loss function by adding another term, the regularization term, which is defined as the sum over all squared weight values of the weight matrix. The regularization term is weighted by a scalar called regularization rate and divided by two.

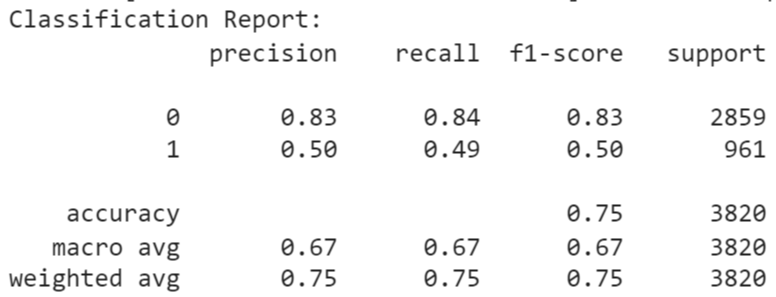
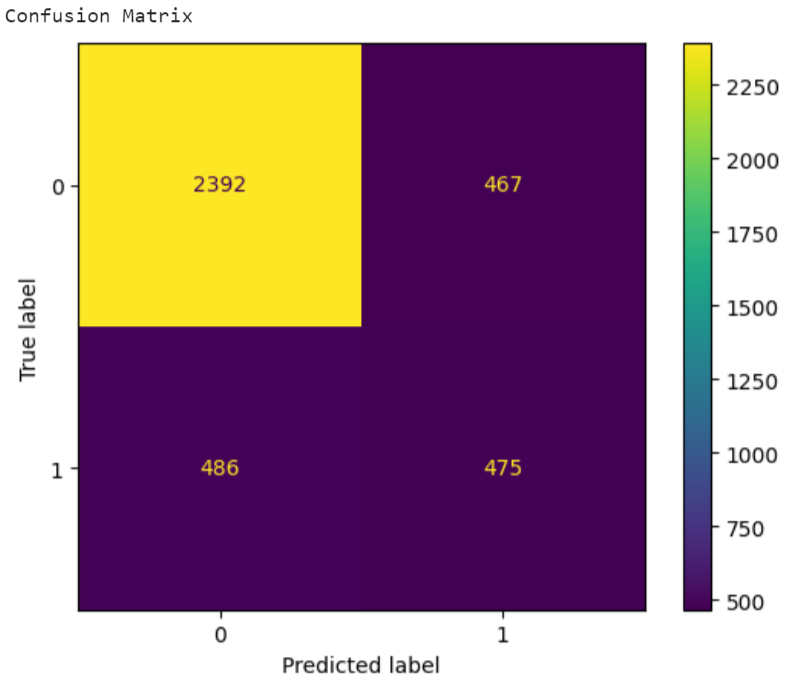


Here the idea is to combine L2 with dropout. This technique refers to dropping out some nodes in a neural network, based on a probability, so that all the forward and backwards connections with a dropped node are temporarily removed. This process makes neurons independent of each other so that all of them can perform better with less noise. To give an example, if a layer has 50 neurons and a dropout is applied with a probability of 0.5, then 25 neurons would be randomly dropped in every iteration.

Therefore, both L2 and Dropout techniques were implemented on the neural network.

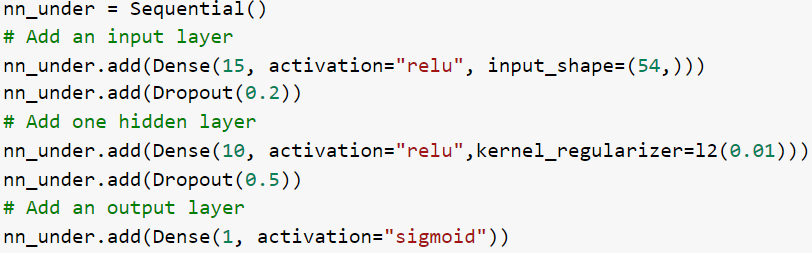


The neural network with this structure was trained, obtaining the following performances on the test set:

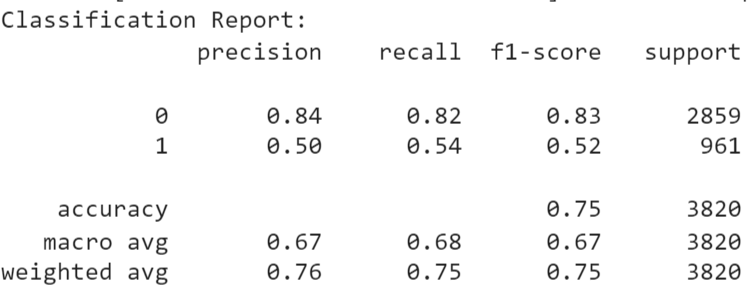
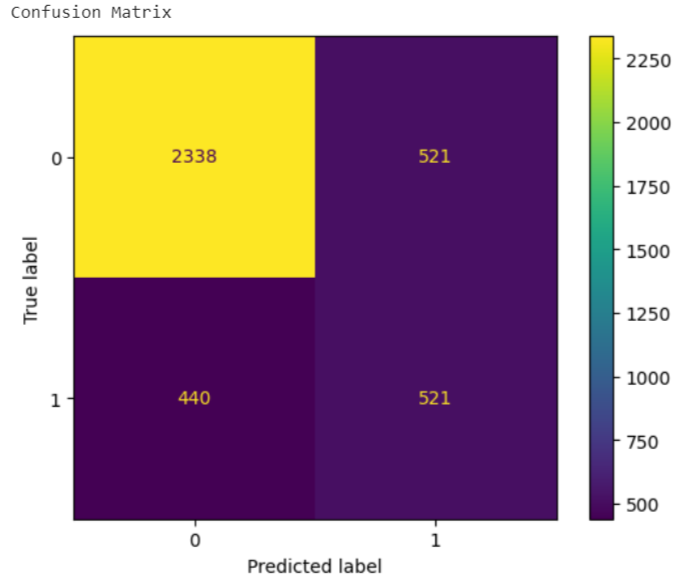


## 5.6.2 UNDERSAMPLING

Since the undersampled training set consists of around 8000 records, the number of nodes was reduced with respect to the one used for the oversampled case to decrease the amount of parameters. The new neural network has 15 neurons in the first hidden layer and 10 in the second, leading to a number of trainable parameters equal to 996.



The neural network with this structure was trained, obtaining the following performances on the test set:



# 

# 

# 

# 

# 

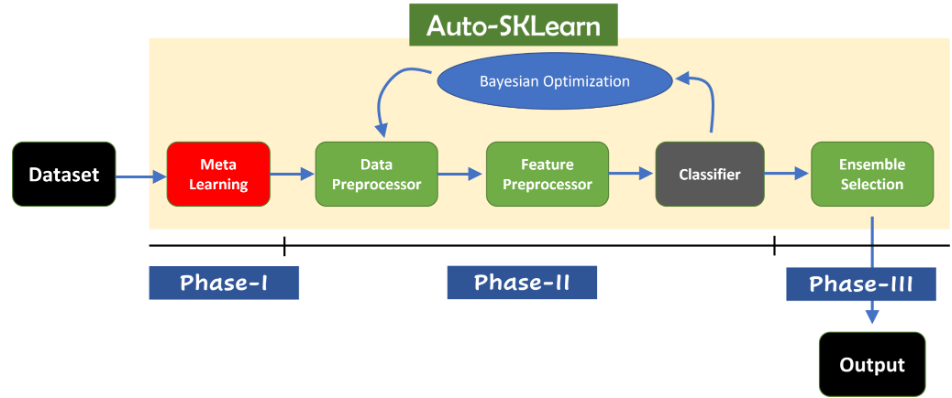
# 

# 

# **6. AUTO-SKLEARN**

Auto-Sklearn is an automated machine learning toolkit and a drop-in replacement for a scikit-learn estimator. It frees a machine learning user from data preprocessing, hyperparameter optimization, model selection and evaluation. Thus, it automatically discovers well performing models for predictive tasks with a very little user involvement.

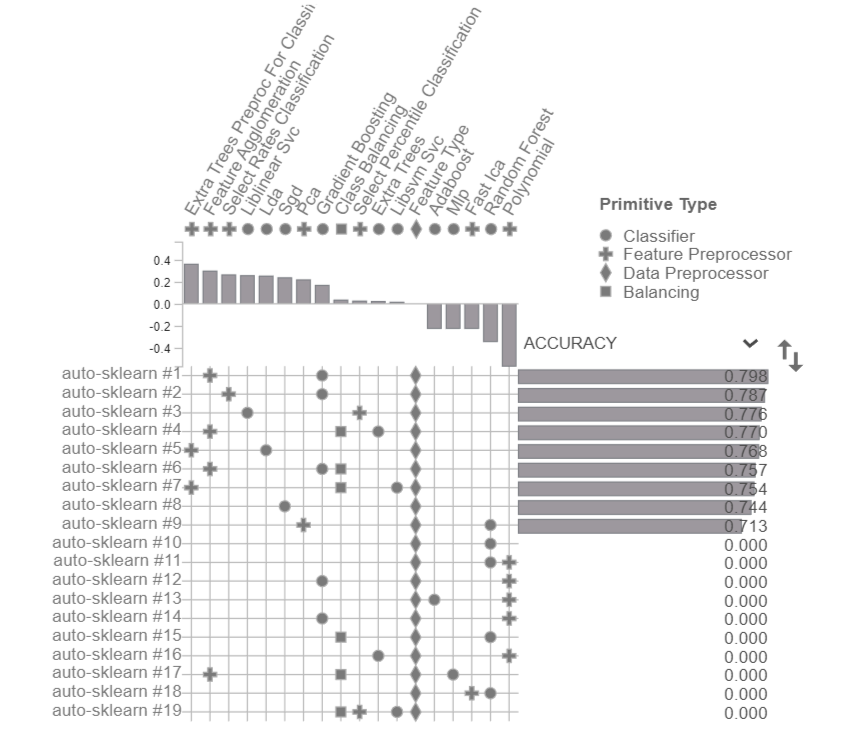
Auto-Sklearn consists of 15 pre-configured algorithms, such as AdaBoost, Random Forest, Extra Trees, KNN, Naive Bayes, 14 methods for feature processing and 3 methods for data preprocessing. Its architecture is shown in the following picture.



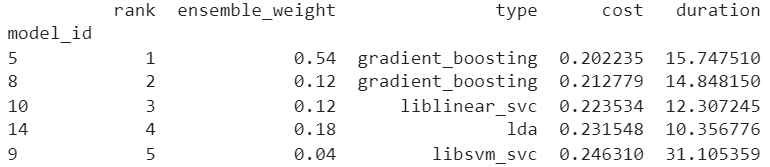
The user supplies the pipeline with raw data which has to be previously divided into training and testing sets. The Meta Learning phase is then executed, which is one of the greatest advancements of this framework: it computes the similarity between the dataset provided and others already known (from OpenML), and if there is a match, a list of techniques that performed well on such dataset is passed to be investigated through the pipeline. The optimization cycle is the second phase: a data preprocessor, a feature preprocessor and a classifier are randomly selected, and then the bayesian optimizer is used to optimize their hyperparameters until a threshold is reached (set by per\_run\_time\_limit parameter). This cycle is repeated for each available classifier until another threshold is reached (set by time\_left\_for\_this\_task parameter). During the third phase, an ensemble of all the sub-pipeline combinations, ranked from the most accurate to the least accurate, is built. Ensemble models combine weighted outputs of multiple trained models to provide a final prediction.

Auto-Sklearn was applied to our dataset obtaining an accuracy score around 0.78. The time\_left\_for\_this\_task parameter, which represents the maximum number of seconds allowed for the entire pipeline search, was set to 60s\*10. By increasing this value auto-sklearn has a greater likelihood of discovering superior models but also overfitting data, therefore, it’s important to find a good balance.

The pipeline is shown in the following picture.



Below we can also see the ensemble of models with their ranks and weights.



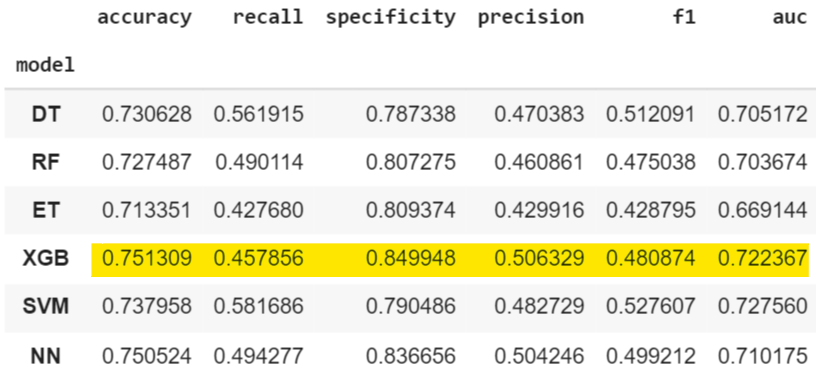
# 

# 

# **7 COMPARISON AND CONCLUSION**

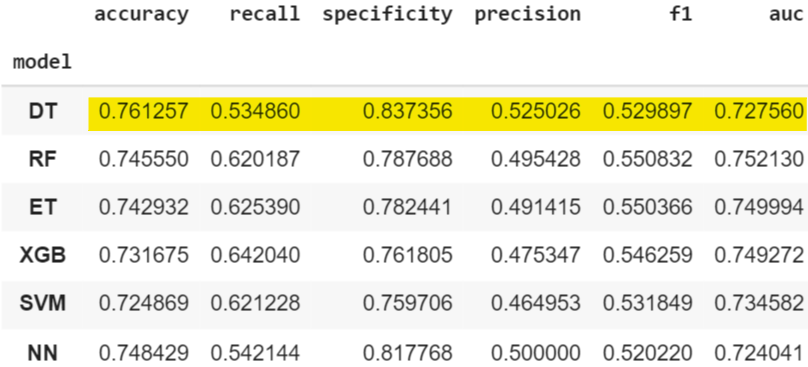
After the detailed phase of data preprocessing and the training of several classifiers, the results can be easily compared since model performances were computed using the same metrics on the same test set.

*OVERSAMPLING*



All classifiers collected very similar results. The one which performs slightly better in terms of accuracy, specificity and precision is XGBoost.

*UNDERSAMPLING*

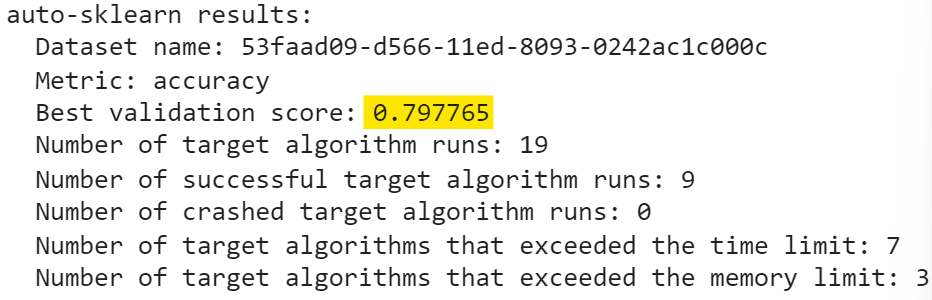


All classifiers collected very similar results. The one which performs slightly better in terms of accuracy, specificity and precision is Decision Tree.

Despite the application of data sampling techniques, all models trained on the oversampled and undersampled datasets predict better the majority class with respect to the minority one. In fact, the results obtained in confusion matrices and in the previous tables show that:

* the percentage of correct positive predictions is lower than the one of correct negative predictions
* recall and precision which focus on the positive class are usually low (max recall value is 0.64, max precision value is 0.52)
* specificity which focus on the negative class is higher (max value is 0.85)

*AUTO-SKLEARN*

**

Therefore, after the application of a complete set of data preprocessing techniques, such as feature removal, handling of missing values, feature scaling, data sampling and feature encoding, and the training of several models, the performances obtained are almost the same. The slightly best model capables of predicting whether a candidate really wants to work for the company after the training courses or will look for a new job, seems to be the decision tree classifier trained on the undersampled dataset. In conclusion, this model was used to predict target values of those records provided by Kaggle for the submission.

LINK KAGGLE: <https://www.kaggle.com/datasets/arashnic/hr-analytics-job-change-of-data-scientists?datasetId=1019790>

LINK COLAB: <https://colab.research.google.com/drive/1WcU3TP9dqqVQqjHp3DNtBUSKuiIYKDEd?authuser=1#scrollTo=NnB0d-phsHKZ>

<https://colab.research.google.com/drive/1Wcd5TIxAPwufh-8nG58ijGApK4t5AQNd?authuser=1>