Data Science - Project 1
Data creation and transformation

I. Dataset's choice

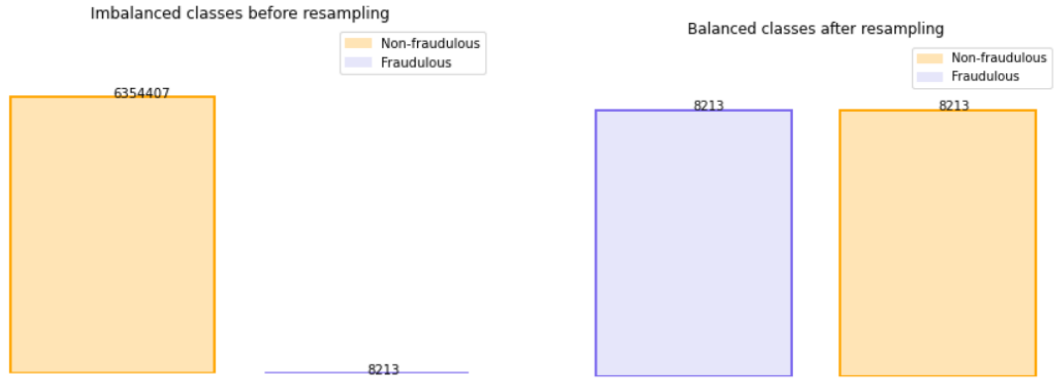
The chosen dataset is issued from kaggle.com and is accessible in the source section of this report. It is used to predict fraudulent bank transactions. A transaction is characterized by different features, some which are categorical and some numeric. We can classify a transaction with the following types: debit, payment, transfer, cash-in and cash-out. Each row of the dataset is also identified by the number of steps, the amount of money engaged in the transaction, the old and new balances for the original and destination accounts as well as their names.

Concerning the fraud detection, it depends on the transaction's type. Indeed, for a transfer, it will be considered as a trickery when the recipient's balance does not match the estimated result. For the cash-out operation, the fraud consists in retrieving money from the beneficiary's account but also from another one. Then, the person would steal money from a random account.

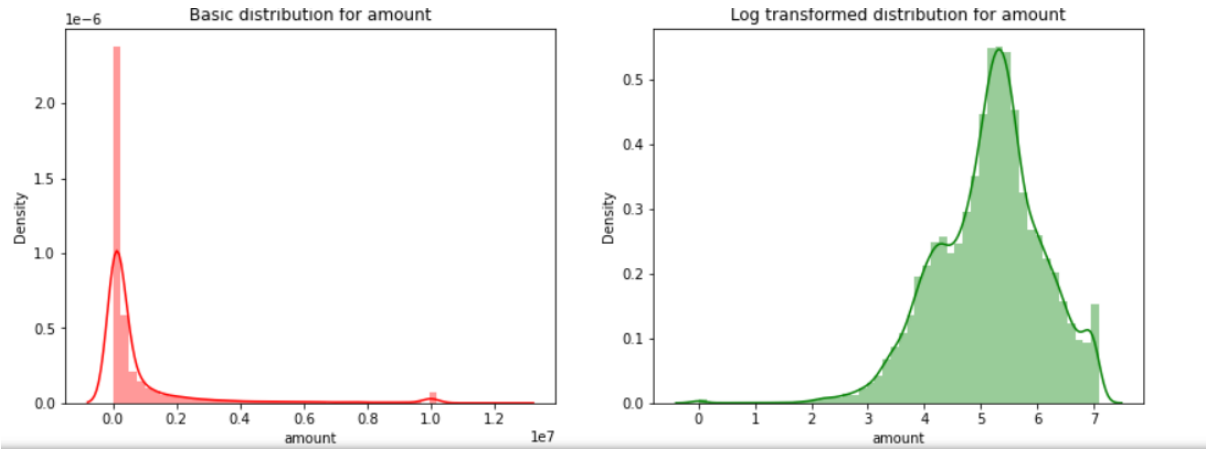
II. Mandatory pre-process

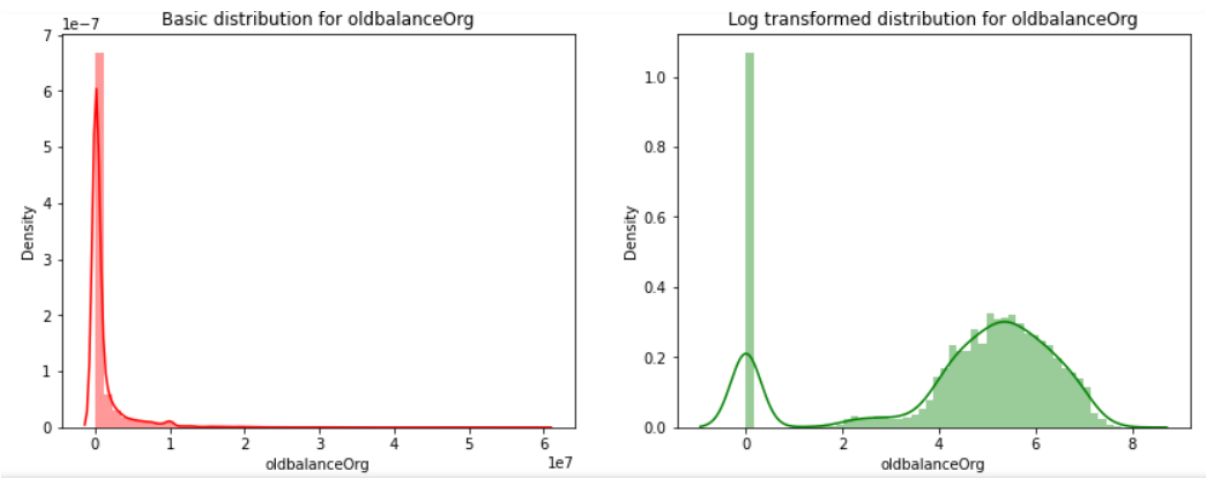
As mentioned above, the dataset has categorical features which are the following: the transaction's type, the initial account's name and the recipient account's name. On the first hand, we know that there are only five different types for transactions. We can then encode it using the LabelEncoder object from the sklearn.preprocessing package. This feature transforms a String object into a number and consign it into a dictionary to assign the same number to the corresponding String. On the other hand, the account names are unique, it would not be optimal to encode it directly using the LabelEncoder. However, it is possible to distinguish an alphabetic prefix followed by a sequence of numbers for each name. A good solution would be to split the name features into two new features: the account prefix and the identification number. Then, this prefix can be label encoded to fit to the machine learning models. Here, we are doing some **feature creation**.

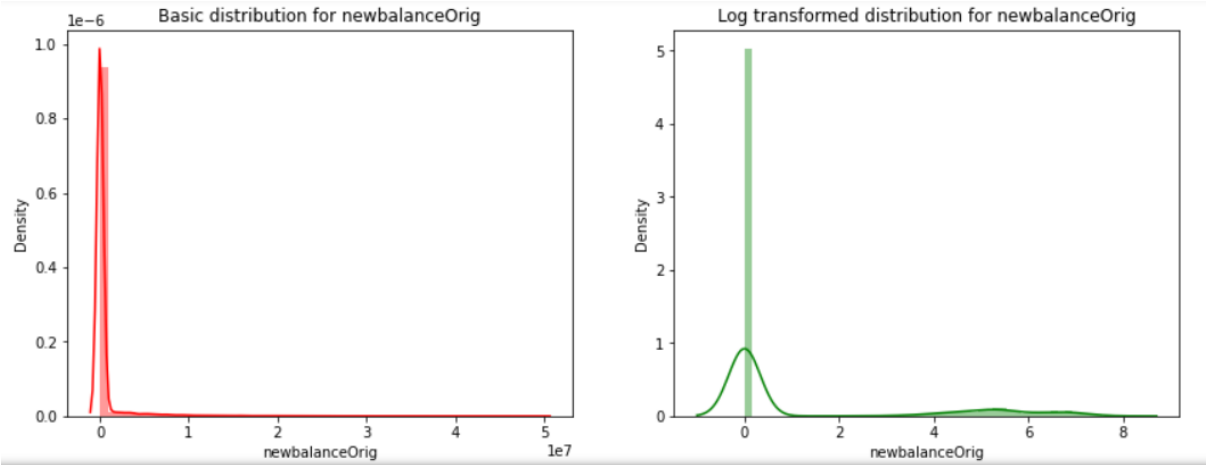
Before processing those data engineering functions, it is important to sample the dataset. Indeed, the dataset is composed of more than 6,000,000 rows which is too much in terms of computing time and resources. Moreover, the dataset is very imbalanced since there are only 8,000 cases of fraud against nearly 6,000,000 normal transactions. Then, the sampling strategy would be to keep 16,000 rows: 8,000 fraudulent transactions and 8,000 non-fraudulent transactions. We did some **data reduction** since we sampled the data in order to balance the classes.

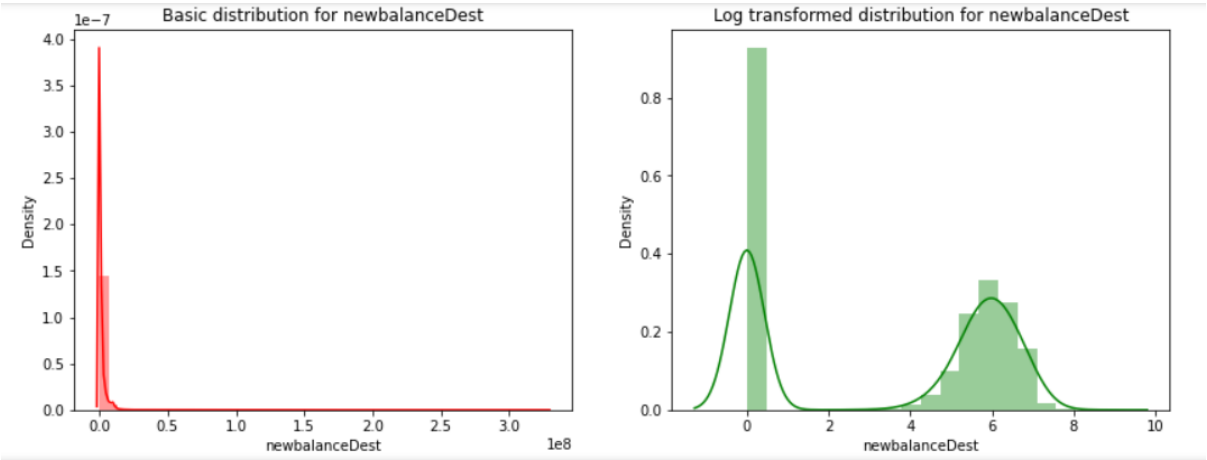


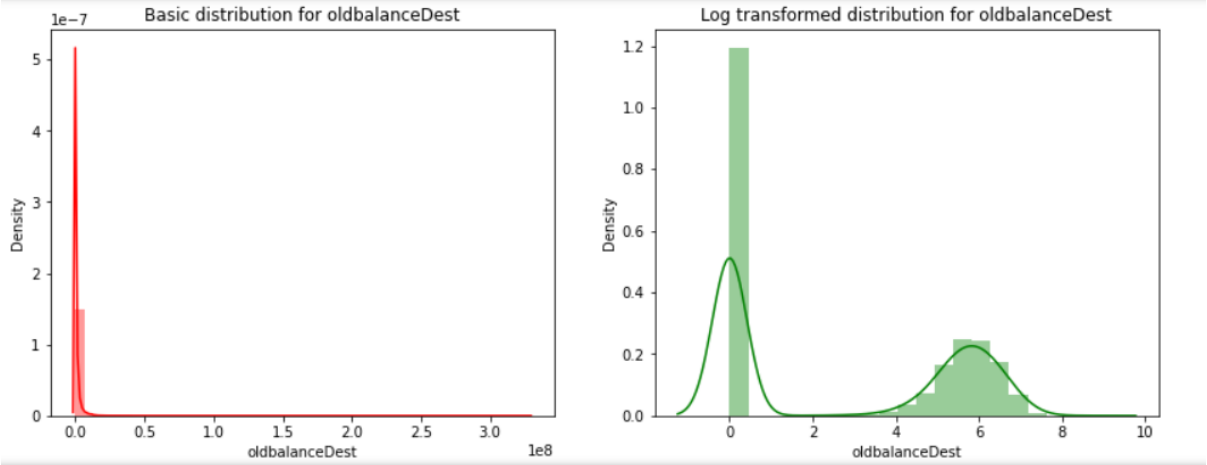
Moreover, it would be interesting to look at the feature's distributions. Indeed, it is well known that machine learning algorithms perform better on normal distributions than others. Thus, applying the log transformation on the features which do not have this kind of distribution will boost the accuracy. The log transformation is a part of **feature transformation**.



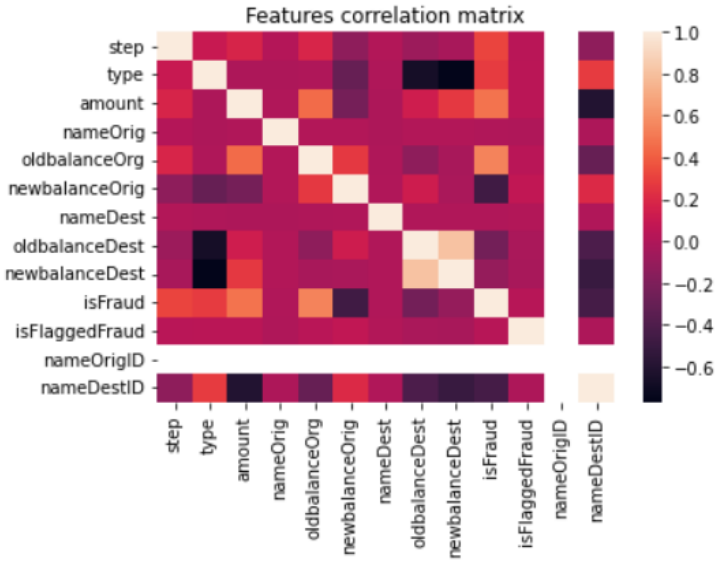






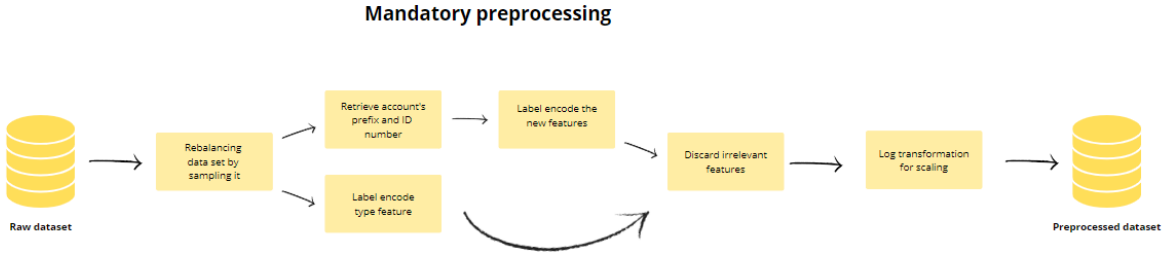


Finally, it would be interesting to check feature's unique values and to search for features which keep the same values in the whole dataset. Then, the 'nameOrigID', issued from the 'nameOrig' feature split, and 'isFlaggedFraud' features have been found to be irrelevant. Also, features which are highly correlated can be considered as irrelevant since they would have the same effect on the result. The following correlation matrix has been computed to observe the dependency of each feature.



The highly correlated features are the couples oldbalanceOrg/newbalanceOrg and oldbalanceDest/newbalanceDest. In general, discarding them would be appropriate. However, since fraud principles resid in the fact that some accounts are illicitly drained or others do not receive the estimated amount of money, it would be inefficient to delete those features.

To sum up about the mandatory preprocessing, those steps allow to train a machine learning model on well fitted features by resampling it, transforming and discarding features. In this case, the whole scenario can be schematized as below:



In reality, rebalancing the classes, transforming the features to fit a normal distribution and discarding irrelevant features are not part of mandatory preprocessing. Indeed, the machine learning model should be able to perform without it. We made the choice to consider it as mandatory since we do not want to focus on those points. The real scope of this study is the **comparison of the data imputation methods**. Then, the starting point of the experiments is from those already well preprocessed data.

III. Data's description

After the mandatory pre-processing, the dataset table looks like below:

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	nameDestID
0	0.0	4	2.257679	2.257679	0.0	0.000000	0.0	1	0
1	0.0	1	2.257679	2.257679	0.0	4.325967	0.0	1	0
2	0.0	4	3.448088	3.448088	0.0	0.000000	0.0	1	0
3	0.0	1	3.448088	3.448088	0.0	4.418334	0.0	1	0
4	0.0	4	4.303801	4.303801	0.0	0.000000	0.0	1	0

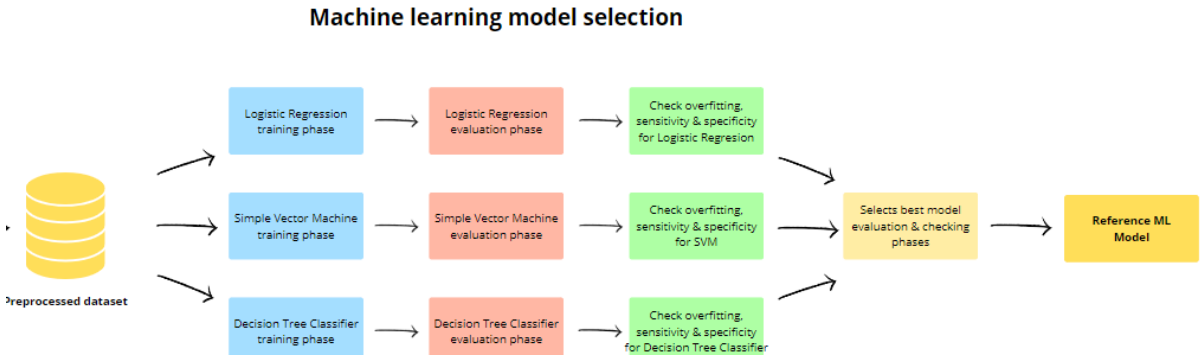
All the features are now numeric, especially the type and the nameDestID ones. Also, the data are all on the same scale because of the log transformation done in the preprocessing part. The data frame is now ready to be used in different machine learning models to choose the reference for the experiments.

IV. Machine learning algorithm reference

All the following machine learning models are classifiers since the task is to detect if the transaction is fraudulent or not. Then, to compare those models, looking at the accuracy is primordial but sensitivity and specificity must also be considered. Specificity and sensitivity are used to describe respectively the type I and II errors. Type I errors consider the false positive results, which are the predictions classified as positive whereas they should be negative. Example: a healthy patient is considered sick and is treated for nothing. On the other hand, type II errors consider the false negative results, which are the predictions classified as negative whereas they are positive. Example: A doctor considers a sick patient as sain and does not give him medications?

To evaluate those models, their default scoring methods are used.

To select the best machine learning model, the following protocol has been applied on the data set



A. Logistic regression

Sklearn provides a good logistic regression model. To use it with the current dataset, it is important to modify the max number of iterations. Otherwise, the computation would fail. In this case, a value of 1,000,000 should prevent it from occurring. Concerning the loss function and the scoring function, the loss is computed by a cross-entropy function and the model's score corresponds to the mean accuracy on the data and labels.

Here are the results from the Logistic Regression part of the protocol:

```
Model's accuracy on reference training dataset:  0.9692866878691504
Model's accuracy on reference test dataset:  0.965504519461354
Difference between accuracies:  0.003782168407796438
Sensitivity =  0.9813323572474377
Specificity =  0.9494235775381182
```

B. Simple machine vector

At the beginning, working directly with C-Support Vector seemed to be a good solution. However, the documentation specifies that over ten thousands of data, working with a SGDClassifier is better. Indeed, it implements a stochastic gradient descent working on a mini-batch of data to prevent too long computation time induced by the use of the classic gradient descent. This model still works with a linear support vector machine to classify the data. Concerning the scoring function, it works the same way as the logistic regression.

Here are the results from the Simple Vector Machine part of the protocol:

```
Model's accuracy on reference training dataset:  0.9713766469786461
Model's accuracy on reference test dataset:  0.967718133185759
Difference between accuracies:  0.0036585137928870637
Sensitivity =  0.9842606149341142
Specificity =  0.9509111193752324
```

C. Decision trees

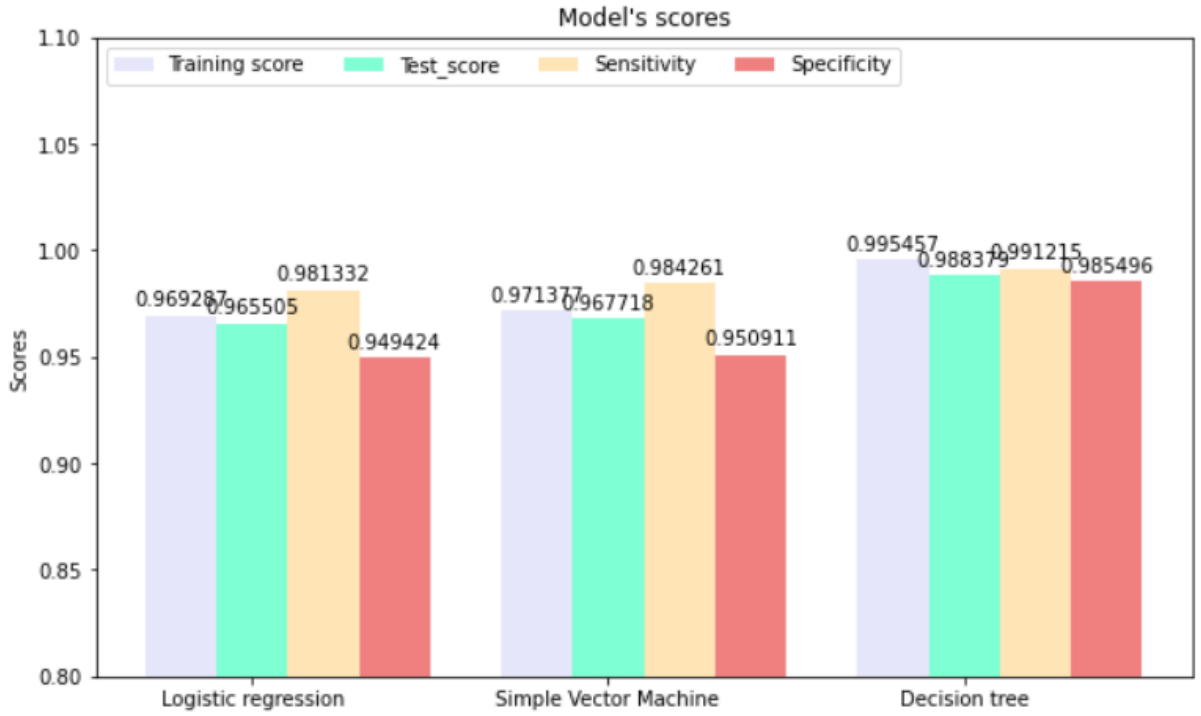
With decision trees, it is important to be careful about depths. Indeed, if this parameter is too small, the model will be in case of underfitting. In the same way, if the maximum depth is too big, the model will be too specific and will lead to cases of overfitting. In this case, to keep the difference between training and testing models great and to prevent overfitting, the tree's maximum depth will be 10.

Here are the results from the Decision Trees part of the protocol:

```
Model's accuracy on reference training dataset:  0.9954566106315311
Model's accuracy on reference test dataset:  0.9885629957572404
Difference between accuracies:  0.006893614874290743
Sensitivity =  0.9912152269399708
Specificity =  0.9854964670881369
```

D. Results overview

To be able to choose the best model, it is possible to plot their characteristics on a bar chart to better compare them. From all the statistics obtain before, it was possible to create the following plot:



From those results, it is easy to conclude that the best suited model is the Decision Tree. Indeed, it presents the best accuracy without tending to overfitting. Also, both its sensitivity and specificity are excellent. Then, this model and its scores on the initial dataset will be used as references for the following experiments. We could also consider the other models since they are all performing well on the data.

V. Going further with the decision trees

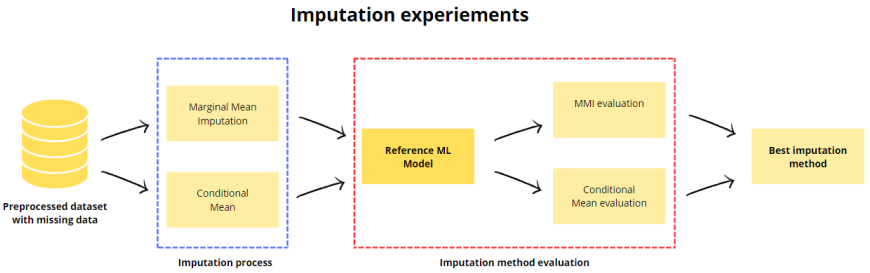
Decision trees are non linear machine learning models that can be used for both classification and regression. In this case, we focus on the decision tree classifiers. This model provides a lot of parameters to adjust itself to the training data. However, it is important to configure these parameters otherwise the model would learn too well on the training data and it would result in overfitting. In the case of these experiments, we limit the depth of the tree to 10 to prevent overfitting. We could also limit the sample leaves and other parameters but the most important is the tree depth because each level provides a new condition to classify the data into the estimated class.

VI. Preprocessing experiments

Since one goal of the project is to restore missing data by the use of algorithms studied in class, we must randomly remove some cells on some specific columns. In the created method, both dataframe, number of data to be removed and target columns must be given as parameters. Since it is random, the missing data distribution in columns should be uniform, meaning that with a high number removed data, there should be approximately the same amount of missing values in each column. In the case of these experiments, we will randomly remove 12000 data from the columns 'type', 'amount', 'oldbalanceOrg' and 'newbalanceDest'. The dataframe on which the experiments will be performed looks like this:

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	nameDestID
0	0.00000	NaN	0.353662	0.353662	0.0	0.000000	NaN	1	0
1	0.00000	1.0	0.353662	0.353662	0.0	0.636083	0.000000	1	0
2	0.00000	4.0	0.537578	NaN	0.0	0.000000	0.000000	1	0
3	0.00000	1.0	0.537578	0.537578	0.0	0.645259	0.000000	1	0
4	0.00000	4.0	0.633852	0.633852	0.0	0.000000	0.000000	1	0
5	0.00000	1.0	NaN	NaN	0.0	0.579455	0.611131	1	0
6	0.00000	1.0	0.749666	0.000000	0.0	0.302893	0.843114	1	0
7	0.00000	NaN	0.785776	0.785776	0.0	0.000000	NaN	1	0
8	0.00000	NaN	0.785776	0.785776	0.0	0.000000	0.805384	1	0
9	0.00000	4.0	0.657520	0.657520	0.0	0.000000	0.000000	1	0
10	0.00000	1.0	NaN	0.657520	0.0	0.652567	0.588602	1	0
11	0.00000	4.0	0.643372	0.643372	0.0	0.000000	0.000000	1	0
12	0.00000	1.0	NaN	0.643372	0.0	0.597505	0.656430	1	0
13	0.00000	NaN	0.709553	0.562664	0.0	0.000000	NaN	1	0
14	0.00000	4.0	0.730096	0.730096	0.0	0.000000	0.000000	1	0
15	0.00000	NaN	0.730096	NaN	0.0	0.000000	0.730096	1	0
16	0.30103	4.0	0.781029	0.781029	0.0	0.000000	0.000000	1	0
17	0.30103	1.0	0.781029	NaN	0.0	0.000000	0.781029	1	0
18	0.30103	4.0	NaN	0.776982	0.0	0.000000	NaN	1	0
19	0.30103	1.0	NaN	NaN	0.0	0.709425	0.781021	1	0

In this project, we focus on the study of different imputation methods. In our case, the two methods that interest us are the Marginal Mean Imputation (MMI) and the Ensemble Learning Imputation.



A. Marginal Mean Imputation

The principle behind the marginal mean imputation is simple. As a first step, it is important to compute the means of all the columns where there are missing values which should be restored. Then, all these missing values are replaced by the corresponding mean. This imputation method leads to a huge approximation but what is the impact on the fraud prediction ?

To be able to make a comparison with the previous results, we compute the score on the prediction of the reference model which is a DecisionTreeClassifier. Thus, we obtain those results:

```
Model's accuracy on reference training dataset:  0.9741026805997274
Model's accuracy on reference test dataset:  0.9640287769784173
Difference between accuracies:  0.01007390362131011
------ Stats on predictions -------
Accuracy =  0.9640287769784173
Sensitivity =  0.9685212298682284
Specificity =  0.9594644849386389
```

Globally, these scores are really good even if a bit lower than those from the originally preprocessed dataset. This decrease in accuracy can be explained by the coarseness of the imputation method. Indeed, because the mean tries to fit a maximum of data, it sometimes rounds too much data which leads to this small decrease. Lastly, we notice that the training and test scores are close so we can conclude that there is no overfitting. Using this method can be a good solution to predict some results with incomplete dataset.

B. Ensemble learning

The ensemble learning imputation method relies on the use of machine learning to predict the value of the missing data from the existing data. We create an EnsembleLearningImput object to provide the necessary methods for restoring the data. It is important to note that a machine learning model always takes the same kind of data as features. If there are some missing features, it will result in an error. To prevent this kind of issue, we create a machine learning model for each arrangement of data entries to be able to predict all the different labels. An arrangement of data can be described by a descriptor which consists of n characters which are either 0 or 1, 0 meaning that the feature exists and 1 meaning that the

feature is missing (NaN). Each machine learning model is only trained with their associated descriptor and the predictions are then made following the same scheme. If there is enough training data, all the missing data should be restored accurately at the end of the processing.

Note that following the type of the variable to predict, the model changes. Indeed, the 'type' column is a categorical variable and then needs a classifier instead of a regressor. The chosen models here are the DecisionTreeClassifier and the GradientBoostingRegressor because they provide the best results.

You can find below the restoration accuracy of each column. Globally, it is between 83% and 98% which is huge for data restoration if we compare it to the Marginal Mean Imputation.

```
Accuracy on the restoration of the type (type=classification)
Model's accuracy on reference training dataset:  0.8575249209822514
Model's accuracy on reference test dataset:  0.8169219547775346
Difference between accuracies:  0.04060296620471682
----------------------------------------------------
Accuracy on the restoration of the amount (type=regression)
Model's accuracy on reference training dataset:  0.876007765228949
Model's accuracy on reference test dataset:  0.8275472668723067
Difference between accuracies:  0.048460498356642234
----------------------------------------------------
Accuracy on the restoration of the oldbalanceOrg (type=regression)
Model's accuracy on reference training dataset:  0.8198103838062974
Model's accuracy on reference test dataset:  0.7958179887000945
Difference between accuracies:  0.02399239510620299
----------------------------------------------------
Accuracy on the restoration of the newbalanceDest (type=regression)
Model's accuracy on reference training dataset:  0.9848872527035477
Model's accuracy on reference test dataset:  0.9830219551414845
Difference between accuracies:  0.0018652975620632173
----------------------------------------------------
```

Concerning the prediction from the restored data, you can find the scores below:

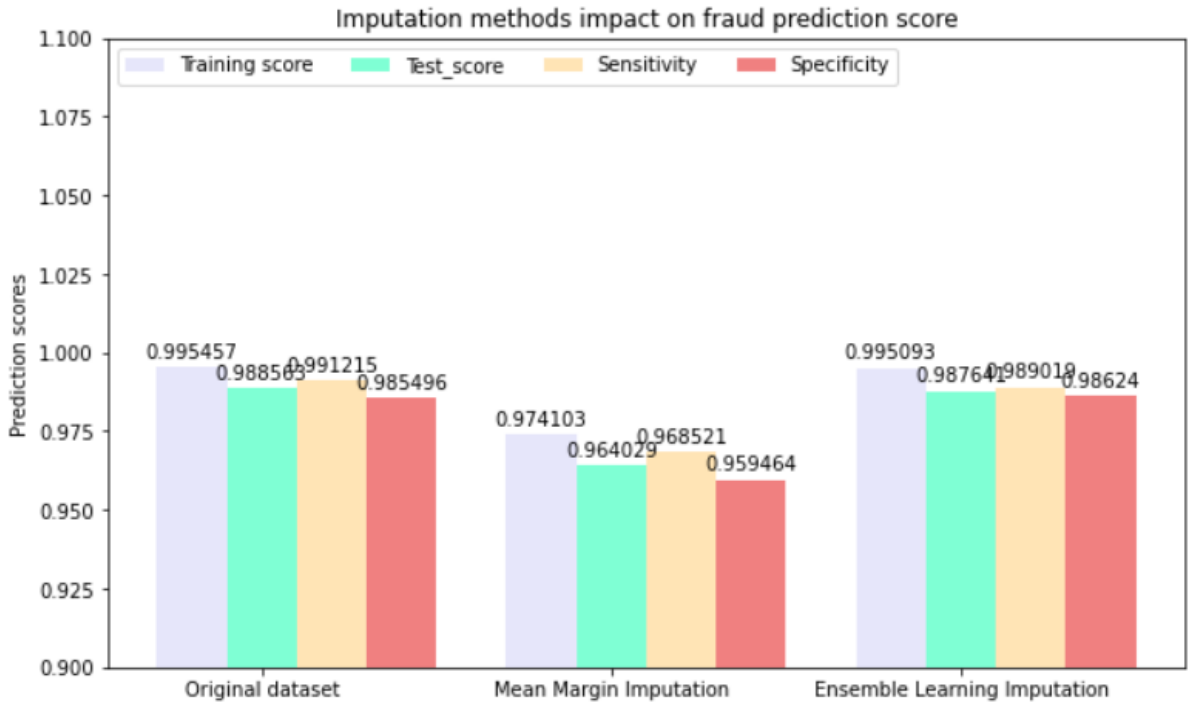
```
Model's accuracy on reference training dataset:  0.9950931394820536
Model's accuracy on reference test dataset:  0.9876406567054049
Difference between accuracies:  0.007452482776648672
------ Stats on predictions -------
Accuracy =  0.9876406567054049
Sensitivity =  0.9890190336749634
Specificity =  0.9862402380066939
```

Then, we can predict the fraud flag from the restored data to observe how it impacts the result. We notice better results than for the Marginal Mean Imputation which is normal since the estimated values are closer to their real ones. Thus, the training and test scores are obviously closer to those from the original dataset. Lastly, the gap between the training and test scores is small so we can conclude that there is no overfitting in the prediction. It is

important to note that restoring data can take some time since it uses a lot of machine learning models which need to train on a batch of data and predict the expected value one by one.

VII. Conclusion on experiments

To conclude on these experiments, we notice that the Ensemble Learning Imputation method provides better scores than for Marginal Mean Imputation. Indeed these scores are very close to those from the original dataset. But at what cost ? In fact, it takes a lot more time to compute the data restoration before being able to predict the fraud flag. If we look at the Margin Mean Imputation, we notice that the score is very good for a computing time very fast. Depending on the number of data to restore and on the resources that we have (time and computation power), both solutions can be taken into account.



Sources

https://www.kaggle.com/datasets/vardhansiramdasu/fraudulent-transactions-prediction
https://matplotlib.org/stable/gallery/lines_bars_and_markers/barchart.html
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.make_pipeline.html
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html#sklearn.linear_model.SGDClassifier
https://scikit-learn.org/stable/modules/tree.html#tree