Scuola universitaria professionale
della Svizzera italiana

SUPSI

University of Applied Sciences and Arts of Southern Switzerland
Department of Innovative Technologies

Data Challenge 3

DATA CHALLENGE PROJECT GROUP 2

D3A: Dyuman Bulloni, Manuel Ippolito, Nathan Margni
Fall Semester 2022

Examiner: Sandra Mitrovic
SUPSI, Lugano Switzerland

27/07/2023

Table of Contents

List of Figures

1. Abstract

2. Problem Definition

2.1 Task

The given task was to correctly predict whether a given customer would default or actually manage to repay their debt. Particular importance, given the topic, had to be posted on the explainability of the prediction. As the task has academic purposes, most models will be black boxes to test out different solutions.

2.2 Dataset Structure

For achieving in, we had at our disposal many data from previous customers and already the correct labeling for the result of their situation. The data is structured in the following features:

- Amount of the given credit, including both customer and their family.
- Gender
- Education Level
- Marital Status
- Age
- Histories of last payments records (April-September 2005)
- Amount of bill statement (April-September 2005)
- Amount of previous payments (April-September 2005)

All currencies are stored in the NT dollar currency.

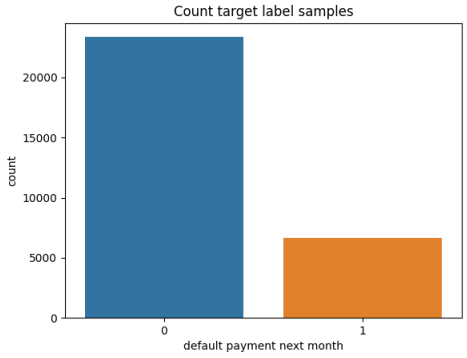
3. Dataset Analysis

To better understand the task and the data, we performed a simple Explanatory Data Analysis (EDA).

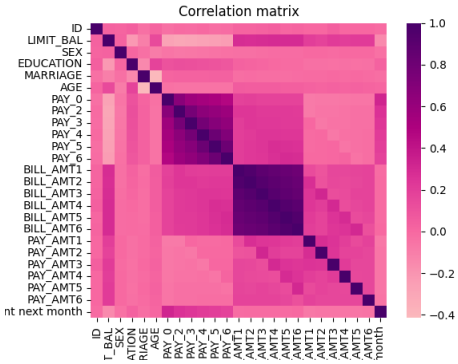
3.1 Our approach to the dataset

We first plotted the proportion of the classes in the target feature (see figure 1a) and discovered that it is highly imbalanced, with 78% of Not Defaults and 22% of Defaults.

We analyzed a correlation matrix (see figure 1b) between the features and it shows that the related features with temporal translation are highly correlated between them, especially "PAY_N" and "BILL_AMT_N" features.



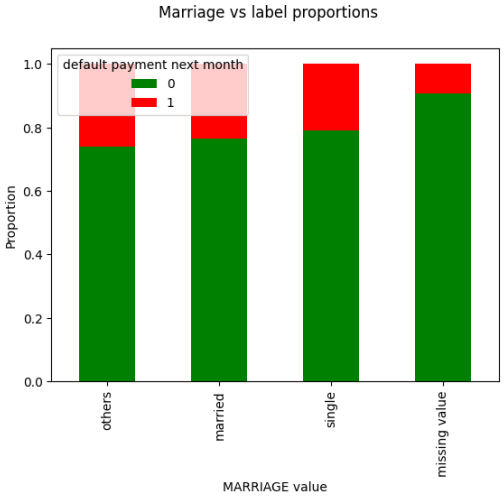
(a) Count of samples for both label class



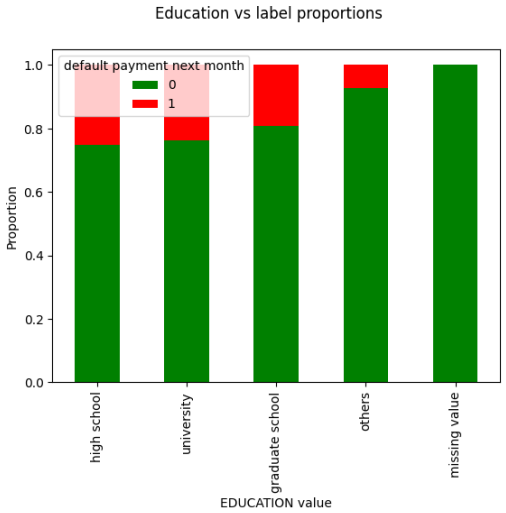
(b) Correlation matrix with all the variables

Then we made some plots to see if there was correlation between the features and the target, and some of them showed a relatively high correlation for example the marriage feature, where the plot (see figure 2a) showed that "married" and "other" status people are more likely to default than "single" and people with unknown status (missing value).

Also the Education Level showed a correlation with the target feature (see figure 2b): people that come from high-school or university have higher ratio of default than the others.



(a) Plot of label proportion depending on the marriage status

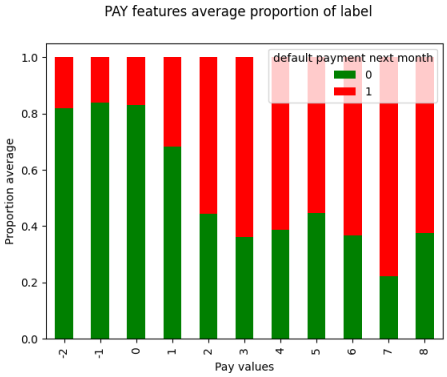


(b) Plot of label proportion depending on the education¿

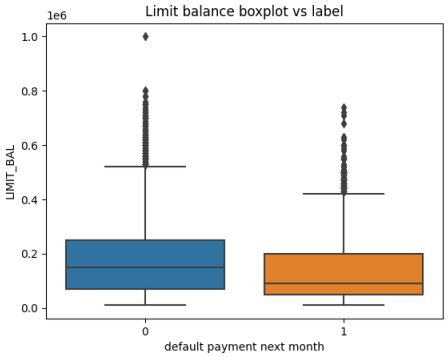
We analyzed the proportions for the PAY_N features' values and observed that when the payment is delayed (value higher than 1) the proportion of people that default is much higher than on the other cases (values lower than 1), this both by analyzing all the PAY_N columns separately or merged by computing the average proportion.

Below we show the merged one (see figure 3a).

Another plot that we analyzed is the one of LIMIT_BAL vs the target, with a boxplot (see figure 3b) and it shows that people that default tend to have a slightly lower balance limit.



(a) Plot of label proportion average for pay columns



(b) Boxplots of limit balance depending on the label

Thanks to this analysis we already knew more or less which features could be the best candidates for being the most relevant for our models.

4. Data Preprocessing

4.1 Data Handling

The dataset was provided already in a clean condition.

No feature had outliers considering a 1.5 interquartile range. Also, it is important to note that due to the specificity of the field, the presence of outliers except very peculiar numbers should be still considered relevant, due to the great difference that we could have for different client situations.

Missing values, where present, were already being treated as part of the information for the categorical feature. For example, in the features Education and Marital Status there were '0' values, and their meaning was missing value. Given that the information missing could still be considered as a valid information, we treated it simply as another category and kept it in our dataset.

We decided to rename the features on History of past payment (PAY_N), Amount of bill statement (BILL_AMT_N), and Amount of previous payment (PAY_AMT_N) by using as N the corresponding month number, so that their name can have a more useful meaning. These features thus went from having an N from 0 to 6 to having N from 9 to 4, indicating their corresponding months (9=September, 8=August, ..., 4=April).

4.2 Categorical Features

Categorical features (Gender, Education level, and Marital Status) have been treated with the One-Hot Encoding strategy, meaning that each category became a numerical binary feature (dummy feature). The only exception to this is the history of past payments (PAY_0, ..., PAY_6) features, which are structured in the following way:

- -2 → Non consumption
- -1 → Paid in full
- 0 → The use of revolving credit for one month
- From 1 to 8 → Payment delay corresponding to the number of months
- 9 → Payment delay for 9 or more months

As these features contain both categorical and ordinal information, our first idea was to extract with the one-hot encoding the first 3 (from -2 to 0) into binary features and keep the rest as an ordinal feature. After discussing it with our supervisor though, we opted to keep all features as simple ordinal variables since the change would have not impacted considerably the results, also considering that the 3 information they represent are in fact low, and the values are considered the higher they are since they represent the delay of payments.

4.3 Numerical Features

All the numerical features (amount of given credit and the amount of bill and previous statements between April and September 2005) represent the same type of data, the NT dollar currency.

Using a normal scaler such as MinMaxScaler will surely remove this proportion as it only considers the value for the feature that is scaled. Our solution was to approach it using a scaler that could be applied to all features in the same way: the normal-log scaler.

However, as we had some negative values between the columns, and those are not compatible with a natural logarithm scaling, we had first to sum to each value a big number such as M=500'000, to be sure that it will remove any possible negative values without basing ourselves on the data since cannot use the information of the test set and therefore we have no way to determine which value is considered valid. After the normal-logarithm transformation, we removed the ln(M)

from each value, to assure that the data contains the same information.

We decided to keep both the datasets, the normalized one and the natural-log transformed one, so that we could later test both of them and see which one was the best for our purpose.

5. Data Augmentation

Our dataset is highly unbalanced (78% 1s and 22% 0s) so we had to consider methods for solve or partially solve this problem.

There are different possible strategies to handle it:

- Do nothing and use the imbalanced dataset to train the models.
- Apply a Undersampling technique (Random under sampling, NearMiss, etc.).
- Apply a Oversampling technique (Random over sampling, SMOTE, SMOTENC, etc.).
- Mix Undersampling and Oversampling techniques.
- Use different class weights internally on the model (bigger weight for the minority class and a smaller one on the majority class).

The decision ended up being to try to balance the dataset by using oversampling and undersampling techniques.

We started by using SMOTE algorithm to oversample the minority class (5009 samples) to reach the same number of samples of the majority class (17491 samples). This approach created 12482 new synthetic samples for the minority class, so our new minority class was made of more than 2/3 by synthetic samples, and just less than 1/3 of real samples. A problem in our initial approach was found: some of our categorical features were previously encoded as binary numerical features (dummies, with either 0 or 1 as value), and thus the SMOTE algorithm was creating synthetic numerical data as if these were numerical features, giving as output values different from 0s and 1s.

We thus decided to use SMOTENC, another version of the SMOTE algorithm which can handle categorical features, and which was giving correct samples in the output. It is important to remember that the SMOTE algorithm creates new synthetic data from the given one.

Some other oversampling and undersampling techniques were tested, in particular: RandomOverSampler, RandomUnderSamper, and NearMiss. We discarded the NearMiss algorithm as it is based on distance measures, and thus can't handle our categorical features encoded as binary numerical features. The two algorithm RandomOverSampler and RandomUnderSampler instead do resample the given data, and thus doesn't suffer from problems related to distance measure and can handle dummies features.

We then decided to structure our experiments on SMOTENC and RandomOverSampler for oversampling, and RandomUnderSampler for undersampling.

Our idea to define the best data augmentation approach was to create different datasets using the above-mentioned tehniques and top test them on different models. For simplicity and to save time, the selected models were a simple Logistic Regression and an SVC with rbf as kernel.

The Data Augmentation phase was started with 2 datasets:

1. A normalized dataset.
2. A dataset transformed using a natural logarithm transformation, to better handle the skewness of the data.

For each dataset we made 5 new datasets for each selected model, so a total of 10 datasets.
Below the normalized datasets:

1. A normalized oversampled dataset made by using SMOTENC algorithm.

2. A normalized oversampled dataset made by using RandomOverSapler algorithm.
3. A normalized undersampled dataset made by using RandomUnderSampler algorithm.
4. A normalized dataset made by mixing SMOTENC and RandomUnderSampler algorithms, to reach a balanced dataset with not too many synthetic samples.
5. A normalized dataset made by mixing RandomOverSampler and RandomUnderSampler algorithms, to reach a balanced dataset with not too many synthetic samples.

The same for the dataset which was transformed using the natural logarithm: Below the normalized datasets

1. A natural-log transformed oversampled dataset made by using SMOTENC algorithm.
2. A natural-log transformed oversampled dataset made by using RandomOverSampler algorithm.
3. A natural-log transformed undersampled dataset made by using RandomUnderSampler algorithm.
4. A natural-log transformed dataset made by mixing SMOTENC and RandomUnderSampler algorithms, to reach a balanced dataset with not too many synthetic samples.
5. A natural-log transformed dataset made by mixing RandomOverSampler and RandomUnderSampler algorithms, to reach a balanced dataset with not too many synthetic samples.

The dataset made by mixing oversampling and undersampling was made by using a sampling_strategy = 0.3, for both the approaches of oversampling, so with SMOTENC and RandomOverSampler. We tested out percentages from 0.3 to 0.9, but this one was the one producing the best results.

To test the sets on the two selected models (Logistic Regression and SVC with rbf as kernel) we used as a scoring function the F1 score, as it is a more reliable metric when dealing with unbalanced datasets.

By testing the datasets on the two models we figured out that there were not many differences between the normalized dataset and the natural-log transformed dataset, but our preference went for the natural-log transformed dataset, as it can handle better the skewness of the data, and later on we could have normalized it for using certain models like neural networks.

By confronting the 3 techniques of data augmentation used (oversampling, undersampling, mixing over and undersampling) we have seen that also there we didn't find very different results between them. Our choice went to oversampling the minority class to have a bigger dataset.

The two oversampling algorithms tested also resulted in pretty similar results, so despite a very slightly better performance by using RandomOverSampler instead of SMOTENC, we preferred to use SMOTENC as it can create new synthetic data based on the given data, while RandomOverSampler just resamples the given data, and as we are creating many new samples in the minority class, we preferred to use SMOTENC not to introduce that many duplicates (12482 new samples) which could have later caused overfitting for some models.

The chosen dataset, for the reasons above, was the one that was transformed with a natural-log transformation and oversampled using SMOTENC.

By doing research, we found that the best practice to do Data Augmentation while using cross-validation is to apply the data augmentation algorithm at each fold. That is because using a pipeline with SMOTENC and the model can help ensure that the oversampling is performed consistently for each fold of the cross-validation process, and this is important because if the oversampling is performed only once, before the cross-validation, the same minority samples will be used in each fold (included validation), which can lead to overfitting. By using a pipeline, the SMOTENC oversampling can be performed on each fold of the cross-validation separately, and it can help prevent overfitting and improve the reliability of the model evaluation.

For these reasons, we did not directly use the selected augmented dataset, but we did use a pipeline with the preferred data augmentation technique and the model itself to perform cross-validation on the natural-log transformed dataset while tuning the model hyper-parameters, guaranteeing reliable model evaluation.

6. Experimental setup and results

We decided on the following N models to see how they perform and to confront their performances:

- Logistic Regression
- Support Vector Classifier
- XGBClassifier
- K-Means Feature Classifier
- Dense Neural Networks
- Convolutional Neural Networks
- Isolation Forest

The last one, Isolation Forest, is a model that is normally used for anomaly detection. During our researches we found that sometimes Autoencoders and other models dedicated to anomaly detection can perform well also for binary classification, by setting one class (in our case the majority class) as the normal class and the other one (our minority class) as anomalies class. We wanted to experiment with this approach so we decided to introduce this model for experimentation and learning purposes.

6.1 Logistic Regression

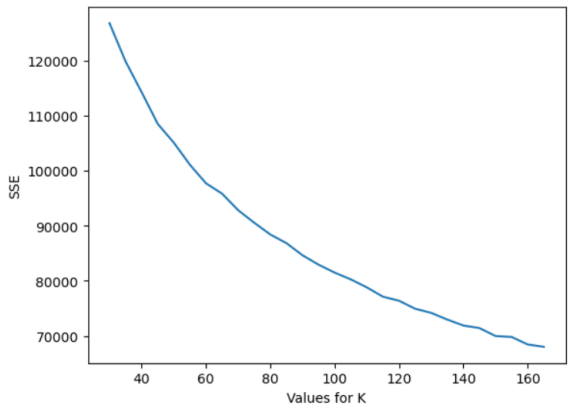
Logistic Regression was used just in the Data Augmentation phase, to help us in detecting the best combination of scaler, transformation and data augmentation techniques. We didn't spend time on it as we selected many other models to be tested, tuned and compared. //

Without tuning it, but just fitting it with the various datasets we made, it reached an F1 score of 0.476. The dataset used to reach this result is the one made by undersampling using RandomUnderSampler and the data was natural-log transformed before it. //

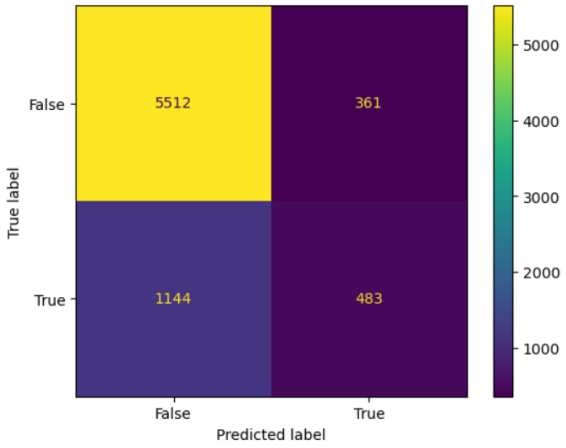
We could have spent some more time on it but we decided to focus on other more complex models for this project.

6.2 K-Means Feature Classifier

A first approach to the dimensionality of the data has been done by the KMeans algorithm.
The choice of the number of clusters is done using the Elbow Method, and as figure 4a shows the sweet spot was found for k=70.
The classification was then made on the cluster affiliation, transforming each cluster into the respective binary feature (and not using One-Hot Encoding structures as there is no possibility to demonstrate that the test set has a representative for all the clusters).
To classify these features we used a simple DecisionTreeClassifier. The resulting score is the model represented in figure 4b is f-score=0.39, which is definitely low.
Nonetheless, these features could provide additional information regarding the customers and could be used in other models. Due to time reasons, unfortunately, we did not have the possibility to run again each model with the consideration of these features.



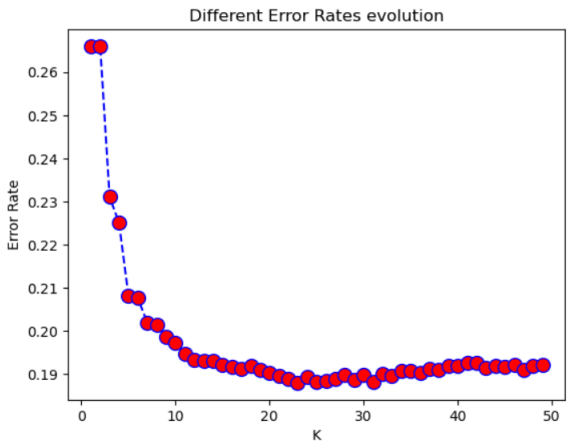
(a) Elbow Analysis for KMeans



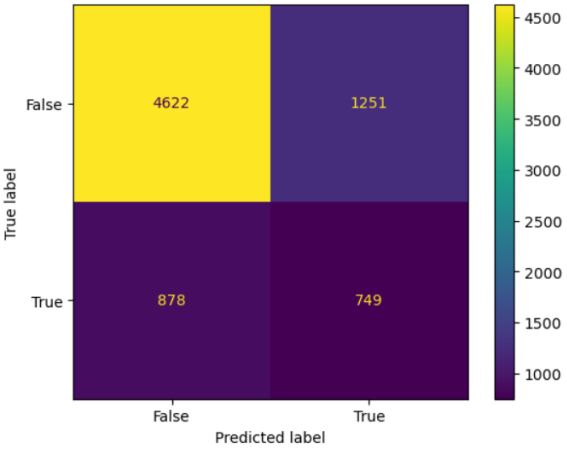
(b) DecisionTreeClassifier using KMeans features

6.3 K-nearest neighbors

We decided to use also this model to try to use better the dimensionality of our dataset, also considering the relationship between many of the features (all regarding currencies). The only hyperparameter that needed tuning was the leaf_node since the weights are better set as 'distance' for unbalanced datasets such as ours. We then studied the number of neighbors, which from a simple analysis shown in figure 5a has been fixed as 31. Unfortunately, even this model represented in 5b did not provide any valuable result, with an F-score of 0.413.



(a) Neighbors Analysis for KNN



(b) KNN Evaluation

6.4 Support Vector Classifier

We decided to use SVC for different reasons:

- It can usually reach high accuracy on many classification tasks.
- It have different kernels (linear, polynomial, rbf (radial basis function), ...), so they are flexible and verstaile for different types of classification tasks.
- It can support high-dimensional spaces and work well also with big datasets (not our case, but it is good thing).

In particular, our focus went on non-linear versions of SVC, like SVCs with polynomial or rbf as kernel.

As Mentioned in the Data Augmentation section, we used Pipelines to perform data augmentation on each fold while cross validating our model. The selected cross-validation type was StratifiedKfold cross validation, as it can assure that in each fold there is the same distribution of data, and this configuration, with a fixed random state, is shared between all the models tested, so that we can compare them fairly.

As SVCs can have different configurations, we performed a grid search using GridSearchCV. the following parameters were tested:

- Kernels: rbf, linear, polynomial, sigmoid.
- Degree (for polynomial kernels): 2, 3, 4, 5, 6.
- C (coefficient of linearization): 1, 100, 200, 300, ..., 1400.
- Gamma: scale (as default).

The same GridSearchCV procedure was performed using 3 different datasets:

1. Oversampled dataset using SMOTENC.
2. Oversampled dataset using RandomOverSampler.
3. Undersampled dataset using RandomUnderSampler.

We found that different datasets lead to different configurations of the SVC model. Below the best parameters per each model-dataset pair:

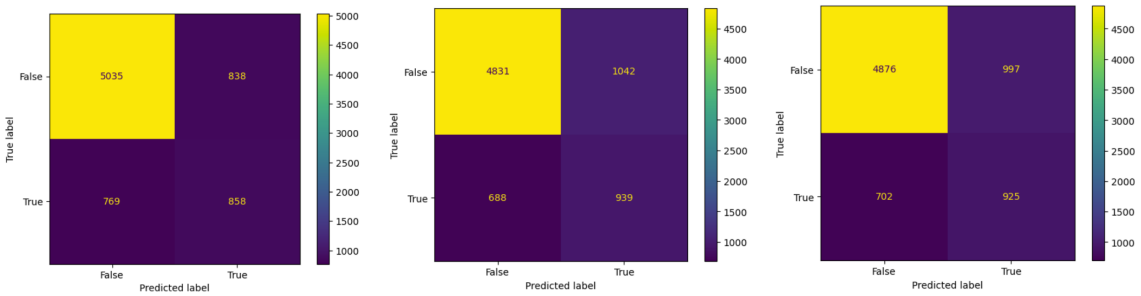
1. SVC with oversampled data using SMOTENC:
   - Kernel: rbf
   - C: 1300
2. SVC with oversampled data using RandomOverSampler:
   - Kernel: rbf
   - C: 500
3. SVC with undersampled data using RandomUnderSampler:
   - Kernel: rbf
   - C: 700

Below we list the best F1 scores reached by each of the configurations above when tested on the test set:

1. SVC with oversampled data using SMOTENC: 0.5164
2. SVC with oversampled data using RandomOverSampler: 0.5205
3. SVC with undersampled data using RandomUnderSampler: 0.5213

The results are all pretty similar, and the same also for the confusion matrices:

We were expecting the oversampled dataset from SMOTENC to be the best performer on the test set, as it creates new synthetic data, while the random sampler takes duplicates of the given ones, and should lead to a bit of overfitting on the models. The performances are very close to each other, but the undersampled dataset made by using RandomUnderSampler is the one with the highest score.

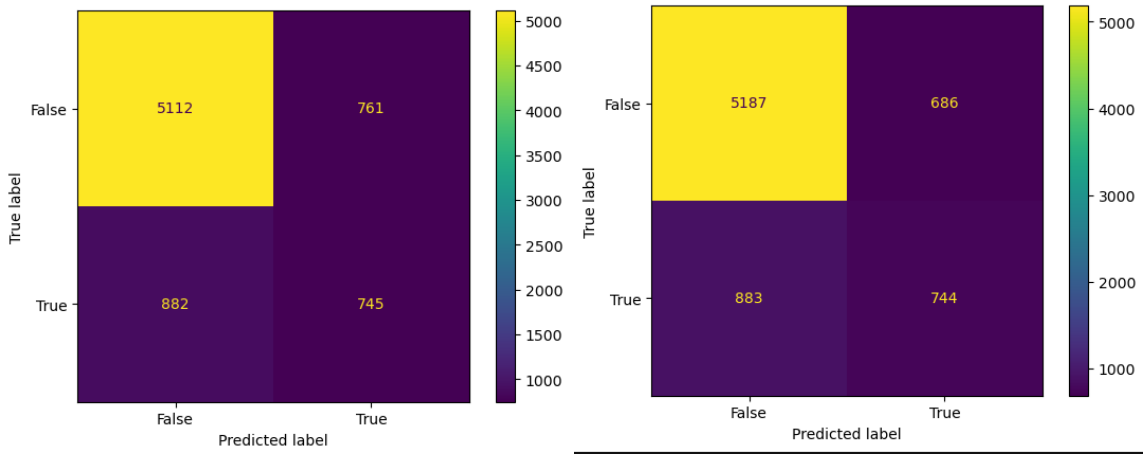


(a) SMOTENC (b) RandomOverSampler (c) RandomUnderSampler

Figure 6: SVCs confusion matrices when fitted on different datasets

6.5 XGBClassifier

eXtreme Gradient Boost Classifier is a scalable model that it's generally very precise and easily adapts to different types of data and problems. We chose it for this and also because, with respect to other models, it has a clear representation of feature importance and being tree-based could be theoretically explained also in this way. Using this model, we can also compare and explain the difference between feature importance for the model and the explainability of the results.
The model does not require a proper type of preprocessing, and given that the normal-logarithm scaled dataset was found to perform better, we stick to that also for this model.
The model was tuned properly using the Optuna library, easy to use and considerably faster as an approach with respect to more complete but slow GridSearch.
Thus we created two different models, that differ only in the consideration or not of the hyperparameter 'scale_pos_weight', which represents the weight given to the respective classes. Generally is 0 by default treating misclassifications in the classes equally, but this can be changed to reflect more on the data. It is particularly useful in unbalanced datasets such as our case. Given the non-consideration of the hyperparameter, the dataset used is the one augmented with the OverSampler strategy, where the other is the unbalanced version, both with the natural-logarithm scaler.



(a) XGBClassifier with augmented dataset (b) XGBClassifier with weighted scale

The model with an augmented dataset and no consideration of the hyperparameter shown in figure 7a has an f-score of 0.476, whereas the model represented in figure 7b has an f-score of 0.487. The unbalanced dataset thus performed slightly better and will be the one to be used in the explanation model chapter.

6.6 Dense Neural Networks

A fully connected neural network should be a good estimator when the data have a mix of categorical and continuous features because can learn complex non-linear relationships between the features and the target label. To pre-process the data we performed MinMax scaling because normalization help the network converging better. We created then a pipeline with the under-sampling method (because over-sampling was giving worst results) and the Keras model, we gave this pipeline to RandomSearchCV so that as explained before the Under-sampling method will be applied in the correct way on the splits. The hyperparameters that we passed to the search method is umber of neurons for the hidden layers, that with two layers will result in n*n possible combinations, we also tried to add dropout layers but that was not increase the performance so we decided to remove them. We used RandomUnderSampler to balance the data but then we discovered that only changing the classweights was bit better, this method consist in giving more weight to the minority class and vice versa doing this we can avoid using a pipeline. So the variants of models that we kept in the notebook are the following:

1. NN with 1 hidden layer using RandomUnderSampler
2. NN with 2 hidden layers using RandomUnderSampler
3. NN with 2 hidden layers using classweights

6.7 Convolutional Neural Networks

We also tried training fully convolutional networks since they should work well when the features are spatially correlated between them, like in our case as seen on the correlation matrix shown in the EDA section, we have groups of correlated features and for that, we thought FNCs could learn interesting patterns. The pre-process was the same as for the dense NN as the pipeline/classweights. The parameter grid passed to the RandomSearchCV instead had different parameters: the number of filters which is the number of feature maps that the network will learn and the number of kernels that specifies the number of weights for each filter. So the types of fcn that we kept are the following:

1. CNN with 1 hidden convolutional layer using RandomUnderSampler
2. CNN with 2 hidden convolutional layers using RandomUnderSampler
3. CNN with 2 hidden convolutional layers using classweights
4. CNN with 2 hidden convolutional layers and 2 dense using classweights

Finally obtaining the validation score from the random search instances we selected only the model with the higher score (0.54 fscore), although all were having similar scores(0.51-0.53), so the best model was the NN with 2 hidden layers and changed classweights, we finally evaluated it on the test set:

On the test set unfortunately we get a slitghty lower f1 score, this because of the randomness of the split on which we evaluate, we got 0.7 of auc score that is relatively a good score (0.5 mean random guessing)

6.8 Isolation Forest

While doing our researches, we found that autoencoders and other models mainly used for anomaly detection can also be used for binary classification, and sometimes they can also reach good performances. We wanted to test this theory out, so we searched for a model different from the autoencoders we already worked on in previous projects, and we found Isolation Forest a good choice.

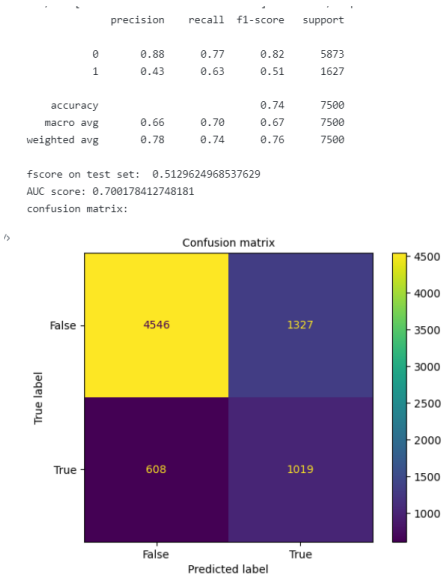


Figure 8: Classification Matrix of NN Model

Our idea was to take the majority class (Did not default) and consider the samples in it as the normal data point, and take the minority class (Default) and consider its samples as anomalies.

We still wanted to test it out on both, the oversampled dataset made using SMOTENC and the not augmented dataset, having 22% of samples in the minority class and 78% in the majority class.

As it is a tree-based model, we didn't perform scaling or transformations, so we gave it the cleaned original dataset, once with oversampled data and once with the original ones.

Again we used StratifiedKFold cross-validation with the same random state as the other models to guarantee the same distribution over the folds and fairness in comparison between the models.

We used a RandomSearchCV to speed up the time needed to find the best hyperparameters with respect to using gridsearchCV, as this model was meant as an extra in the project and we didn't focus on it.

The parameters in the grid for RandomSearchCV were:

- Number of Estimators: 100, 300, 500, 700.
- Maximum number of samples: 100, 300, 500.
- Contamination: 0.1, 0.3, 0.5.
- Maximum number of features: 5, 10, 15.
- Bootstrap: True, False.
- Number of jobs: -1

The best hyperparameters were found to be:

1. Isolation Forest with unscaled oversampled data using SMOTENC:
    - Number of Estimators: 700.
    - Maximum number of samples: 300.
    - Contamination: 0.1.
    - Maximum number of features: 10.
    - Bootstrap: True.

- Number of jobs: -1

2. Isolation Forest with original unscaled data:
   - Number of Estimators: 500.
   - Maximum number of samples: 300.
   - Contamination: 0.1.
   - Maximum number of features: 10.
   - Bootstrap: True.
   - Number of jobs: -1

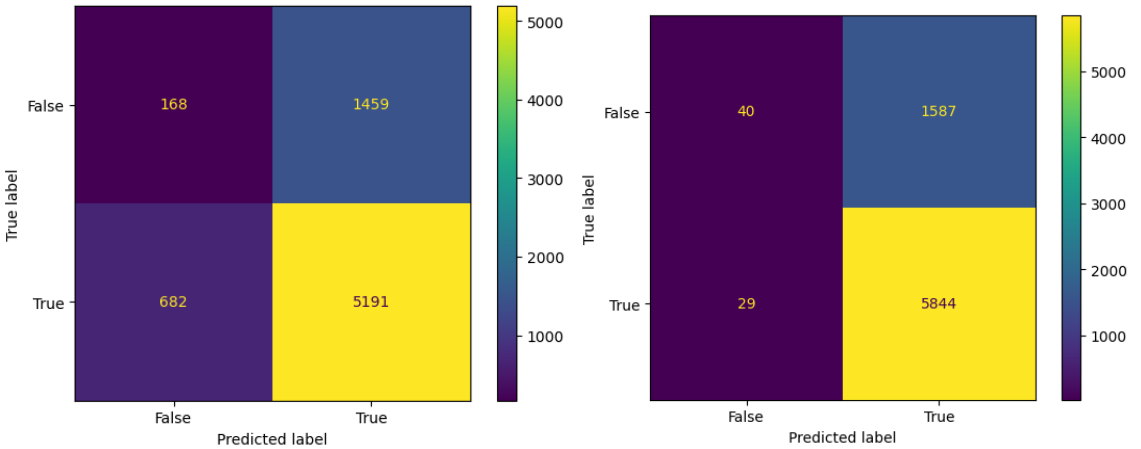
Below are the F1 scores obtained:

1. Isolation Forest with unscaled oversampled data using SMOTENC: 0.829
2. Isolation Forest with original unscaled data: 0.878

The model reached results too good to be true in real life. It is almost always predicting True (Did not default) and as the given data is very unbalanced, it is reaching high performances. It is not clear to us how the model reached 0.829 of F1-score when trained on an oversampled and balanced dataset made using SMOTENC, and then tested on the initial test set that is unbalanced.

The confusion matrices show that the models are predicting almost always True. The one trained on the oversampled data is predicting more Falses, and thus "generalizing" better, but still the results are quite weird.

Below the confusion matrices of the two Isolation Forest models: We also computed the ROC-



(a) Oversampled balanced dataset (b) Original unbalanced dataset

Figure 9: Isolation Forests confusion matrices when fitted on different datasets

AUC score, and it is 0.505, which is more reasonable as a result than the F1 scores above.

We didn't invest a lot of time on understanding why the F1-scores given from Isolation Forest are that weird due to lack of time, but it was nice to conduct this experiment.

6.9 Summary of the results

Best f-scores on the test set per model:

- K-means Feature Classifier: 0.391
- K-Neighbors Classifier: 0.413
- Support Vector Classifier: 0.521

- XGBClassifier: 0.487
- Dense Neural Network:
- Convolutional Neural Network:
- Isolation Forest: 0.878

7. Model Explainability

In this section we will analyze the following 3 models in terms of explainability:

- Support Vector Classifier
- XGBClassifier
- Dense Neural Networks

We want to check whether all the models found the same features relevant and which of the 32 features was the more relevant in general and for each model.

We selected 3 model explainability methods:

- LIME (Local Interpretable Model-agnostic Explanation)
- SHAP (SHapley Additive exPlanations)
- Permutation Feature Importance.

LIME is a local model explainer, meaning that is able to explain what is based on a single prediction rather than the actual behavior of the model. This model is very efficient due to its speed and the support of the different types of data, such as tabular models, text classifiers and even images. This evaluation is very effective against any biases, as explains the reason of the single prediction and therefore the ideal type of explanation required for a task such as the default of customers.

SHAP can be used for both, local and global model explanations and we used it for both purposes. The main idea behind it consists of a game theoretic approach, where it considers how much the features individually contribute to the individual prediction. Doing it on the whole dataset represents a global model that explains the variable importance of the model.

Permutation Feature Importance is a global model explainer that explains how the features are important for the model. It is used only for tabular data. The concept behind it is to randomly shuffle a single feature value and reevaluate the model, breaking in this way the relationship between feature and target and therefore making the score dropping with respect to how much the feature is relevant for the target. **explainability**

7.1 Support Vector Classifier

Local Explanation of SVC model:

- LIME explanation: In this case sample number 7 (index 7 in the dataset) was analyzed. It is a 1 (Default) and it was predicted correctly as Default. We choose this sample to make the example of a customer asking why he was identified as a Default while being actually a Default (other examples in the other models).

  The most relevant features for sample 7 are the PAY_N features, in particular, PAY_9 and PAY_8, and they represent a one-month payment delay in month 8 (August) and a 2 months delay in 9 (September), which let us suppose the payment delay started in August and was

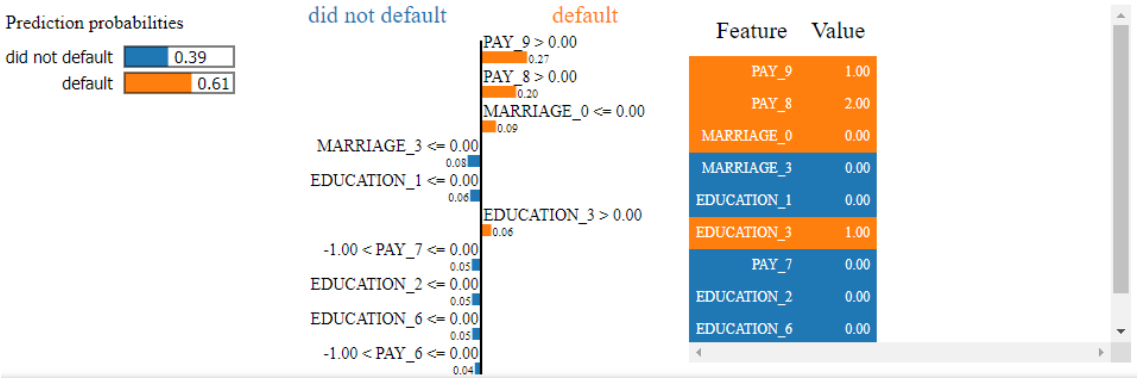


Figure 10: LIME local explanation of SVC on sample 2

continued on September. The other relevant features are Marriage_N features and Education_N features. Marriage_0 represents that the marital status was not specified (missing value), while Marriage_3 represents other statuses besides Single, Married, and missing value. Education_1 = 0 means that the education level of the person is not high school, and Education_3 = 1 tells us that the person's education level is graduate school.

- SHAP explanation: The relevant features identified by SHAP are almost the same as the

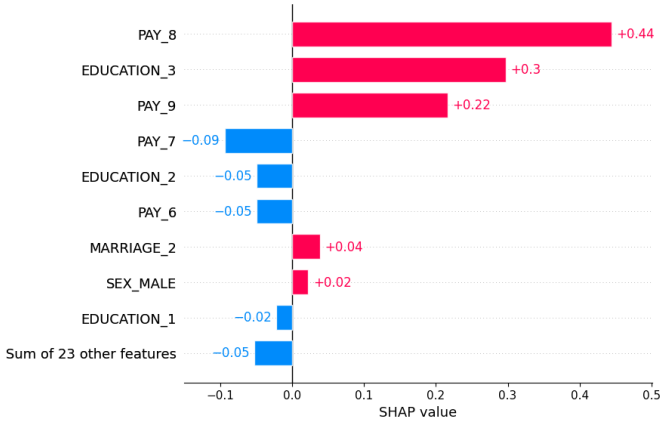


Figure 11: SHAP local explanation of SVC

ones found by LIME. PAY_N features, Education_N are the most important, followed by Marriage and in this case also Sex, which is quite strange. PAY_8, PAY_9, PAY_7, PAY_6, so information on payment delays in August, September, July, and June are important for the model. Education_3, Education_2, and Education_1, so features representing the level of education of the person, are the second most important features. Marriage_2 and the Sex are also in the top 10 most relevant features, but closer to the bottom of them, and they represent whether the person is single or not, and whether it is male or not.

Global Explanation of SVC model:

- SHAP explanation: SHAP was used also for the global feature importances over the whole dataset. Here we see clearly that the PAY_N features are the most relevant ones, followed by the Education_N features, similar to what we found in the local explanation of sample 7.

- Permutation Feature Importance explanation: Our findings from previous local and global explanations are supported by the Permutation Feature Importance analysis: The PAY_N

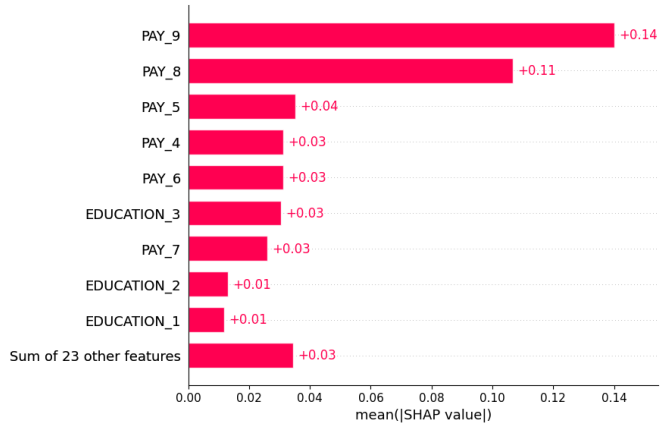


Figure 12: SHAP global explanation of SVC

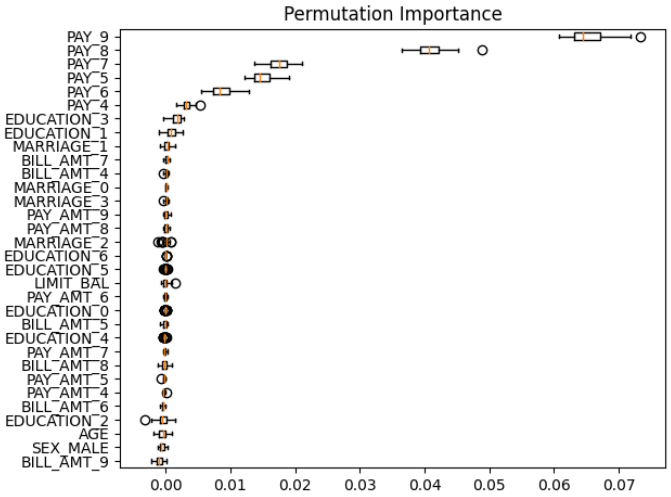


Figure 13: Permutation Feature Importance explanation of SVC

features are always on top for their relevance, followed by Education, Marriage, and in this case, also BILL_AMT_N features, representing the amount statements per each month.

7.2 XGBClassifier

It was chosen for the local explanation of this model sample 4, as it provides one of the most interesting cases for this field, the people that will default that are predicted as they will not default. This type of misclassification is the most cancerous for the bank, and thus we want to prevent those errors as much as possible.

Local Explanation of XGBClassifier model:

- LIME

- SHAP explanation:
  Interestingly, the features that are considered more relevant change between the two explanations: shap model have few features where the most important is the bill amount of month 7, followed by PAY_AMT_8 and PAY_4, where the LIME considers more important the education and only secondly PAY_4 and PAY_AMT_5.

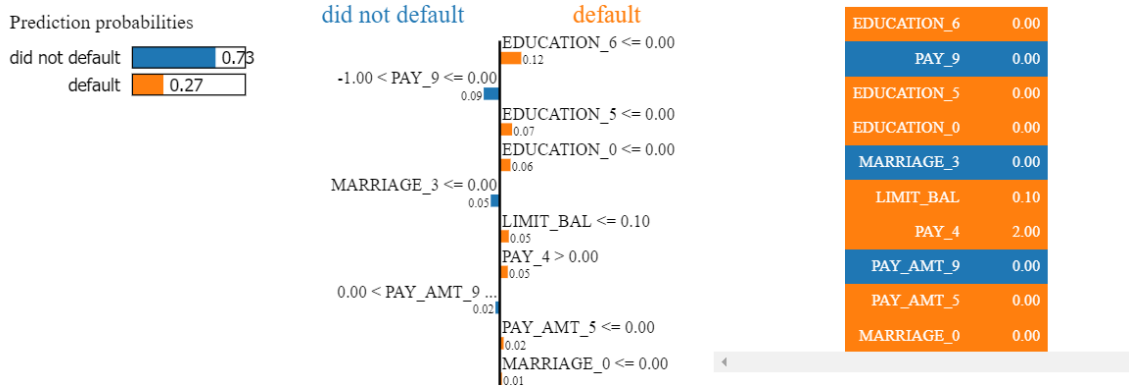


Figure 14: LIME local explanation of XGBClassifier on sample 4

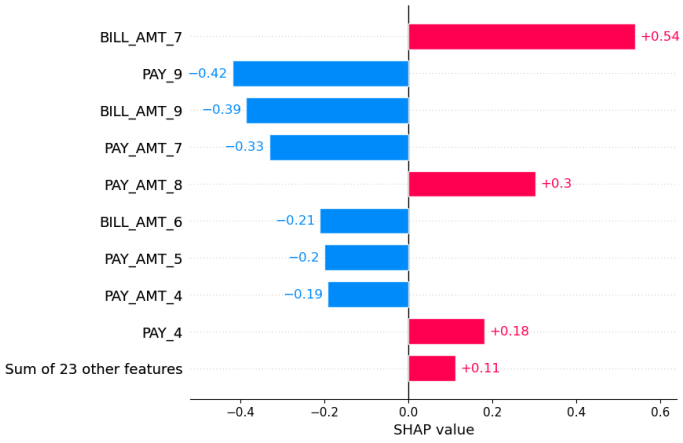


Figure 15: SHAP local explanation of XGBClassifier on sample 4

This difference is really peculiar since SHAP does not even consider the most important education as relevant, but it is not unreasonable to think that with a higher status of education set to 0 (and therefore other "inferior" categories), the person will likely not be able to repay their debts due to their capacity.

Global Explanation of XGBoostClassifier model:

- Feature Importance by XGBoostClassifier:
- Permutation Feature Importance explanation:
- SHAP explanation:

These 3 explanations seem reasonably similar, meaning that globally the model is considered more efficient on PAY_9. The other features change, even if there are many recurrences such as education_N and various PAY_N. Only SHAP considered important LIMIT_BAL, would be interesting to dig more on the reason. Our guess is that using the individual performance from game theory, LIMIT_BAL is actually more important with respect to the ratio towards the other features.

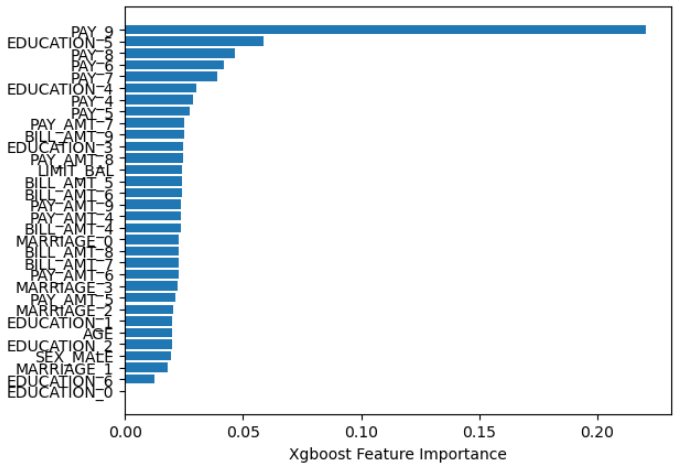


Figure 16: Feature Importance by XGBClassifier

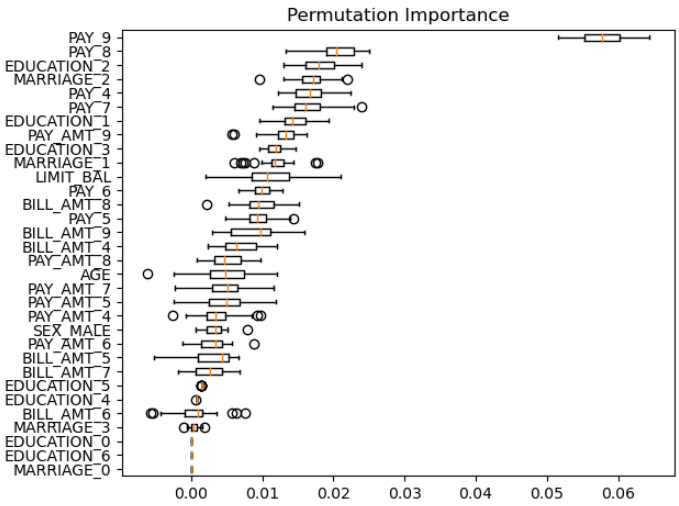
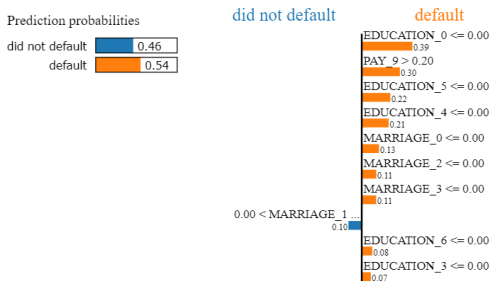


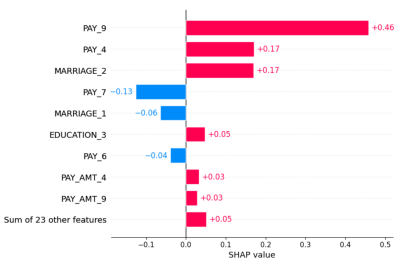
Figure 17: Permutation Feature Importance explanation of XGBClassifier

7.3 Dense Neural Networks

For this model we checked on a False positive sample the local explanation, we can see that on lime the fact that education is not graduate school is the most important factor followed by pay 9 being 1 also a lot important, checking on that sample feature we noticed that all others pay are -2 and therefore is possible that when the last payment was in delay for the first time brings on predicting default. From the shap values, the first most important is in fact pay 9 with a highly positive shap value meaning that this feature increases the probability that this sample is a default



(a) Lime local explanation NN



(b) Shap local explanation NN

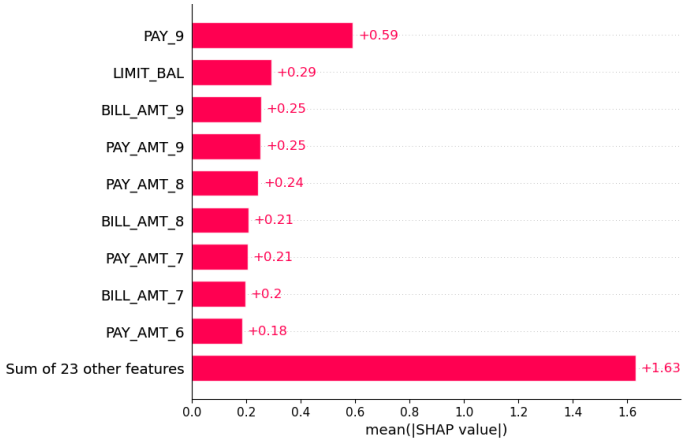
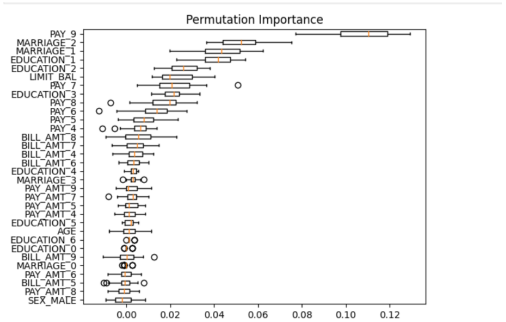
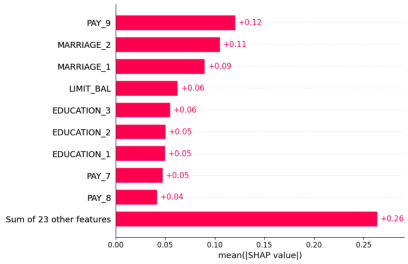


Figure 18: SHAP global explanation of XGBClassifier

The global explanation for the NN with Permutation Feature Importance shows that PAY_9 is the most important feature, and this makes sense because it has information about the last payment that should be the most relevant to the actual status of the user, it is followed by Marriage and Education status that are also very important. Also, the Shap global explanation shows that PAY_9 and Marriage status are relevant, it changes a little that limit balance and education 1 have an inverted position between permutation and shap but in general the ranking is similar.



(a) Feature permutation Global explanation NN



(b) Shap Global explanation NN

7.4 Summary of the results

Here we list the 3 groups of features that were more relevant, in average, for all the models for global explanation:

- PAY_N
- EDUCATION_N
- MARRIAGE_N

In particular PAY_9, PAY_8, Education_3, Education_1, Marriage_1, Marriage_2, were the most relevant features for our global model's explanation.

8. Conclusions

8.1 Critical reflection

One thing that we regret not digging further is the correlation between the features: as many of them represent a temporal range (April-September 2005), the evolution of such data should be very important to determine if a client will default or not.
As the models showed, the last month is really making a difference, but in our opinion approaching the data by extracting numerous features related to the evolution of the trend with respect to other clients, or even considering the singular differences between the months in respect to their total amount available (as even a small payment could increase a lot if put on certain perspectives), could have been helpful in performing better analyses.
Also, the features extracted with KMeans of the cluster affiliation have not been used mainly due to its late completion with respect to the timetable, and maybe they could have been used in an interesting way.

Another path that we only briefly considered is doing binary classification using anomaly detection. Our approach with Isolation Forest led to disappointing scores since it does not recognize properly the anomalies and considers almost all clients as able to repay their debt. Therefore we focused more on the other models, but it would have been interesting to verify if there was room for improvement.

8.2 Personal Conclusions

We liked this project and found it a good opportunity to dig into and apply explainability strategies to black-box models.
The fact that reaching a reasonable f-score with the dataset provided a challenge at the beginning but after trying different things and not seeing it improve it also lowers the general expectations of our work.

@softwarereback2020pandas, author = The pandas development team, title = pandas-dev/pandas: Pandas, month = feb, year = 2020, publisher = Zenodo, version = latest, doi = 10.5281/zenodo.3509134, url = https://doi.org/10.5281/zenodo.3509134

@articlescikit-learn, title=Scikit-learn: Machine Learning in Python, author=Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., journal=Journal of Machine Learning Research, volume=12, pages=2825–2830, year=2011

@articleexplainability models, title = SHAP vs Lime vs Permutation Feature Importance, url= https://pub.towardsai.net/model-explainability-shap-vs-lime-vs-permutation-feature-importance-98484efba066

https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

https://towardsdatascience.com/5-smote-techniques-for-oversampling-your-imbalance-data-b8155bdbe2b5