CEI Lusis

Chaire de recherche Lusis:
Détection de fraude au paiement par carte : étude sur la causalité

CentraleSupélec

lusis

Agenda

Fraud detection

Explainability

LIME

SHAP

Computation time

Analyses

CentraleSupélec lusis

Fraud Detection

Introduction

Financial institutions are working to improve user trust in their fraud detection system. As a result, **explainability** is becoming a key component in machine learning.

Other challenges :

Frauds adapt to new detection methods and pattern of users evolve with time.

There is a very small proportion of fraud compared to total transactions.

Fraud detection algorithms should be real time to be efficient.

Models for fraud detection

There are two types of fraud detection models : expert-driven and data-driven models.

- Expert driven models : Set of predefined rules applied to different scenarios.
- Data driven models : Statistical methods or machine learning algorithms.

Possible machine learning algorithms : Artificial neural networks and random forests lead to best results, with random forests being more interpretable.

Models used for the study :

- Random Forest (from Scikit-learn → split=0.2, max_depth=8, random_state=40)
- Neural Network (provided by Lusis)

Random Forest

Dataset : small dataset (369 821 rows)

Features : 22 basic features

Split train/test : 33%

Model : RandomForestClassifier (scikitlearn)

Model parameters :
- max_depth=5
- random_state=40

Accuracy : 94%

False discovery rate : 5,3%

Confusion Matrix		Actual	
		Fraud	Non Fraud
Predicted	Fraud	4516	253
	Non Fraud	6995	110277

Neural Network

Dataset : small dataset (369 821 rows)

Features : 20 basic features + 95 derived features from mcc and pospayenvcode
(conversion of categorical variables)

Split train/test : 33%

Feature engineering : Standardize features by removing the mean and scaling to unit variance

Model : Lusis NN (2 dense layers + 1 activation layer)

Accuracy : 99,7%

False discovery rate : 1,6%

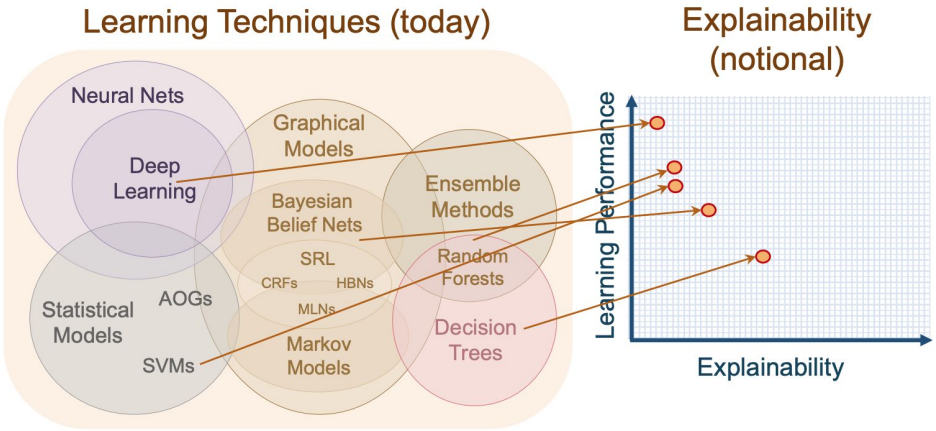
Confusion Matrix		Actual	
		Fraud	Non Fraud
Predicted	Fraud	11303	181
	Non Fraud	208	110349

Explainability

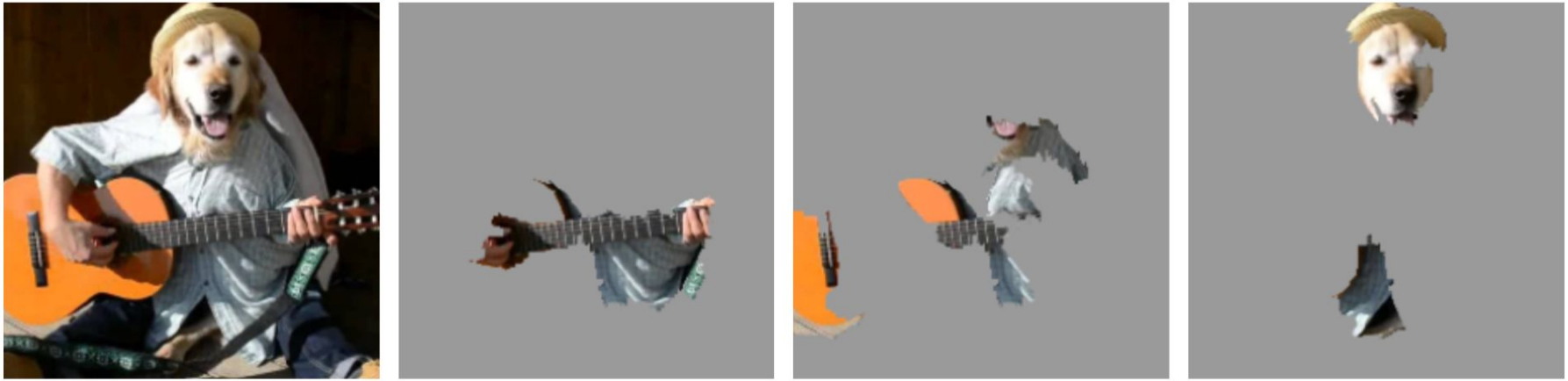
CentraleSupélec lusis

Explainability - Concept

Explainability: Motivated by the opaqueness of so called “black-box” approaches it is the ability to provide an explanation on why a machine decision has been reached.



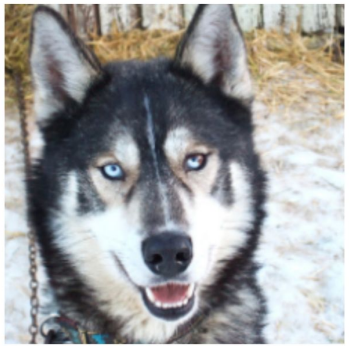
Goal of Explainability



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$), "Acoustic guitar" ($p = 0.24$) and "Labrador" ($p = 0.21$)

Goal of Explainability



(a) Husky classified as wolf

(b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

	Before	After
Trusted the bad model	10 out of 27	3 out of 27
Snow as a potential feature	12 out of 27	25 out of 27

Table 2: "Husky vs Wolf" experiment results.

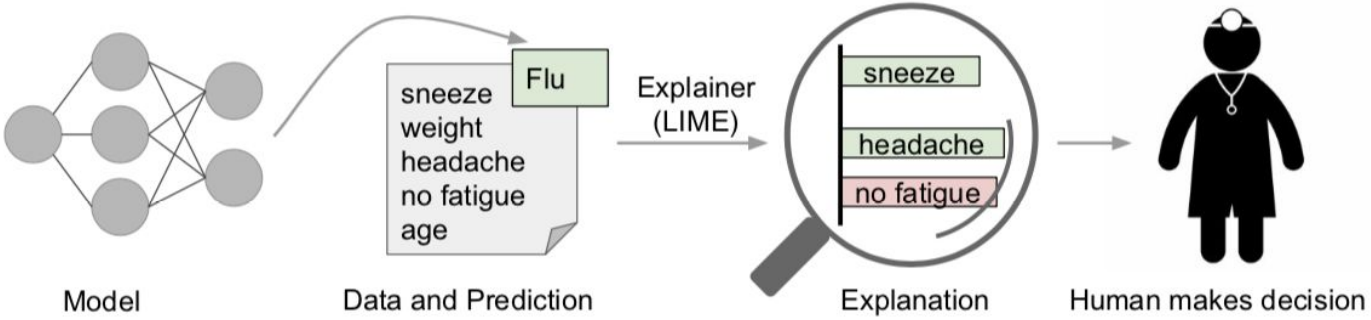
Approaches for explainability

There are 4 main ways to explain an AI model :

- Input Attribution : Explain a model using the input features and attributing a weight to each feature (LIME, SHAP, Integrated Gradients)
- Concept testing/extraction : Extracting information from the internal state of a model
- Example influence/matching : Using significant example sets to explain a model
- Distillation : Learn an explainable model (ex: Decision Tree) from a black box

Input attribution : LIME and SHAP

LIME and SHAP are methods used to explain the predictions of a black box using a local approximation.



Input attribution : LIME and SHAP

LIME and SHAP explain the models by tweaking the input and modeling the changes in prediction. This new input is still close to the original data point.

For example, if the model prediction does not change much by tweaking the value of a variable, that variable for that particular data point may not be an important predictor.

As a result, LIME and SHAP are model agnostic, which is important to compare two different types of classifiers.

LIME

“Why Should I Trust You?” Explaining the Predictions of Any Classifier

Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin

Two different definitions of trust:

- Trusting a prediction : LIME (Local Interpretable Model-agnostic Explanation)
- Trusting a model : SP-LIME (Submodular Pick - LIME)

LIME

The criteria of LIME:

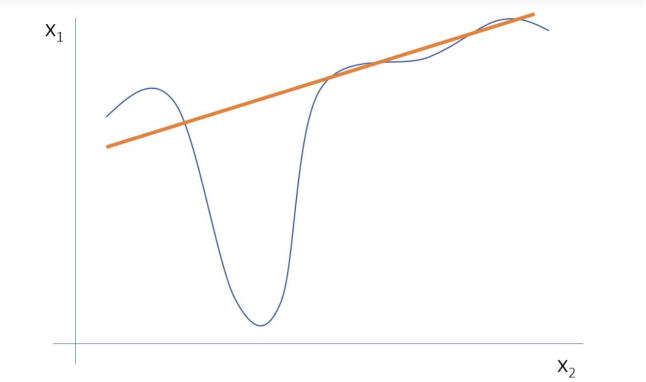
- Interpretable : The features used must be understood by humans
- Local fidelity : the explanation must be locally faithful to the model, even if it is not globally faithful, which is not always possible

Advantages:

- Fast : LIME can provide real-time explanations
- Consistent : Sum of the impact of each feature is equal to the total impact

Drawbacks

- The local approximation is linear, which means that the model cannot be explained if it is highly non-linear.
- The representations may not be powerful enough to explain some behaviors (example : Sepia -> Retro)
- Random components may lead to different explanations for same input
- It requires some tweaks to work on a non-generic model



SP-LIME

SP-LIME evaluates trust in the model as a whole.

The user will inspect a set of instances selectec during a step call the **pick step**, this inspection will denote the global importance of that component in the explanation space.

The set of instances has to be non-redundant and achieves the highest coverage. The set created, is called the **submodular pick**.

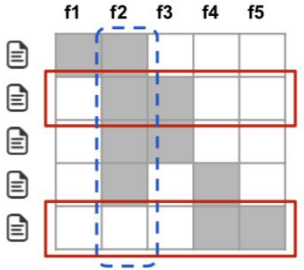
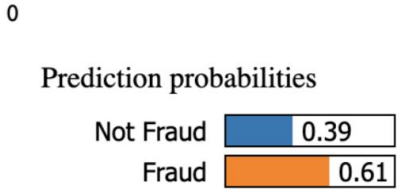
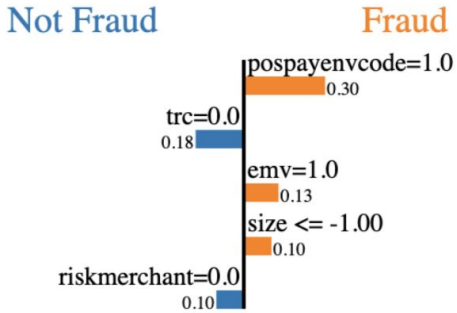


Figure 5: Toy example $\mathcal{W}$. Rows represent instances (documents) and columns represent features (words). Feature f2 (dotted blue) has the highest importance. Rows 2 and 5 (in red) would be selected by the pick procedure, covering all but feature f1.

LIME - False positive

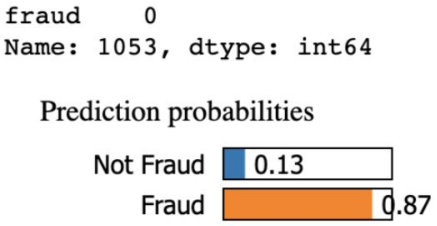
Random Forest

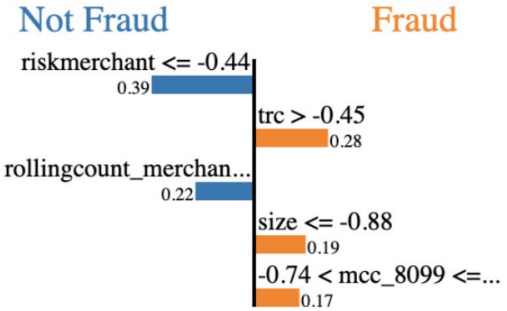




Feature	Value
pospayenvcode=1.0	True
trc=0.0	True
emv=1.0	True
size	-2.00
riskmerchant=0.0	True

Neural Network



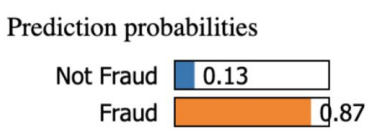


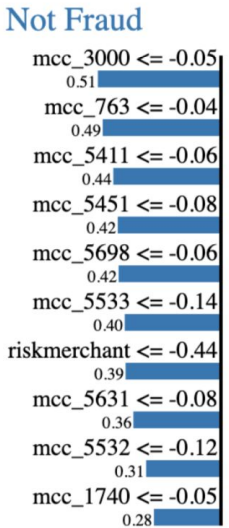
Feature	Value
riskmerchant	-0.44
trc	2.20
rollingcount_merchant_2h	-1.64
size	-0.88
mcc_8099	1.36

Implementation problem

Bug in LIME implementation when we changed the number of explicative feature of the output (from 5 to 10 with the neural network)

```
fraud    0
Name: 1053, dtype: int64
```





Fraud

Feature	Value
mcc_3000	-0.05
mcc_763	-0.04
mcc_5411	-0.06
mcc_5451	-0.08
mcc_5698	-0.06
mcc_5533	-0.14
riskmerchant	-0.44
mcc_5631	-0.08
mcc_5532	-0.12
mcc_1740	-0.05

SHAP

Shapley Values - SHAP

The SHAP explainer is based on Shapley values, which can be used in game theory. For example, for a company with a set of three employees, the Shapley values consider all the ways that adding each employee improved the profit compared to not having those employees.

SHAP value is the contribution of a feature to the difference between the actual prediction and the mean prediction.

Contrary to LIME, SHAP does not assume that the local model is linear, which means that calculation is very time expensive as it checks all the possible combinations.

SHAP pros and cons

The SHAP method has several advantages compared to other explainers:

- Like LIME, it is **consistent** : the sum of the individual impact is equal to the total impact
- unlike LIME, it there is no linear approximation, so it is more **stable** and **accurate**

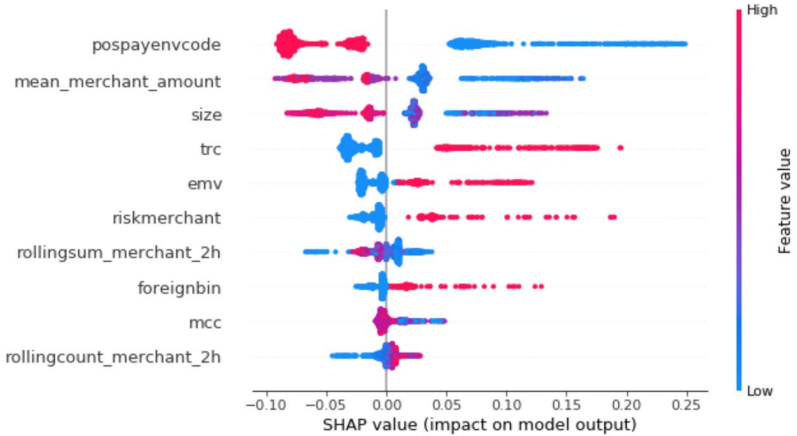
However, the SHAP method is very **time consuming**.

The standard, model-agnostic method is implemented in **KernelExplainer.** However, due to its long running time, optimized versions of SHAP have been implemented.

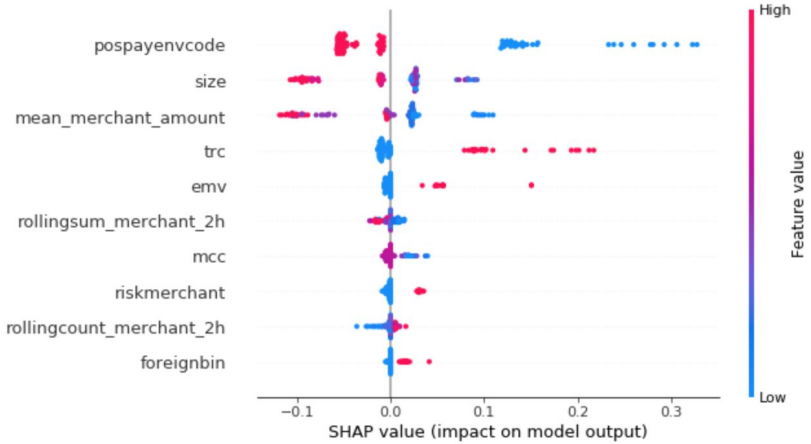
SHAP for random forests

TreeExplainer is optimized for tree-based models.

We compute with the normalized discounted gain a **95% similarity** for 100 explanations between TreeExplainer and KernelExplainer.



Global feature importance for TreeExplainer

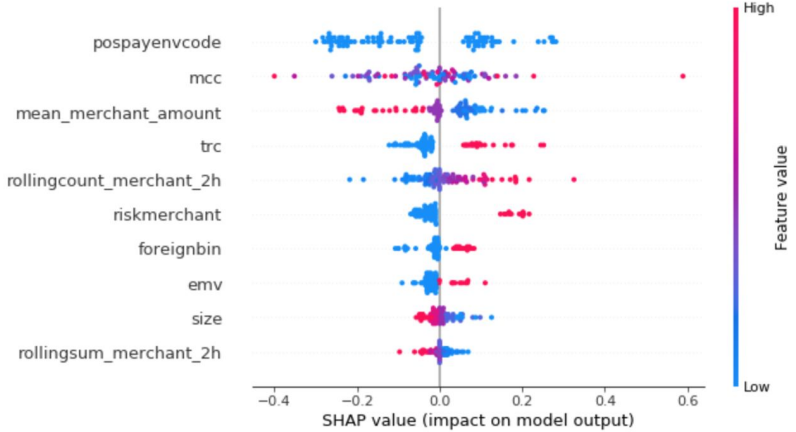


Global feature importance for KernelExplainer

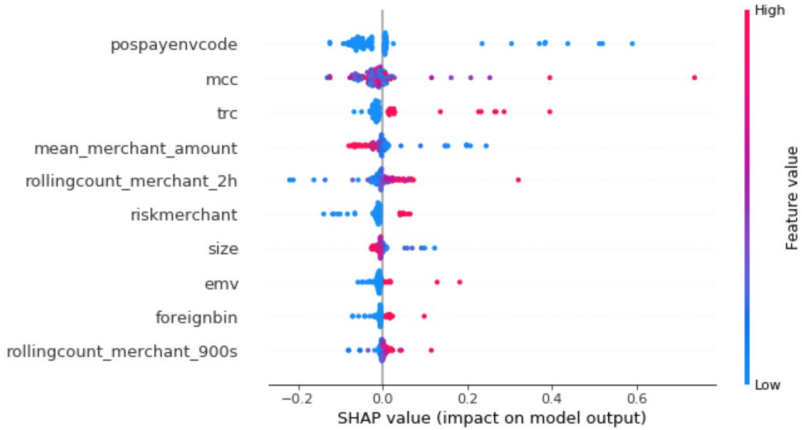
SHAP for neural networks

DeepExplainer is optimized for neural networks.

We compute with the normalized discounted gain a **48% similarity** for 100 explanations between DeepExplainer and KernelExplainer.

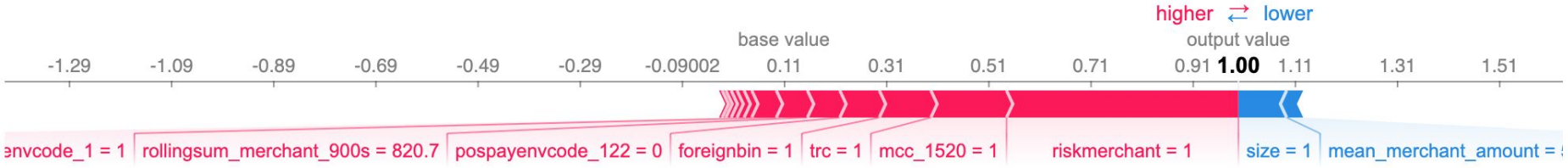


Global feature importance for DeepExplainer



Global feature importance for KernelExplainer

SHAP KernelExplainer vs DeepExplainer



SHAP KernelExplainer

SHAP DeepExplainer

SHAP Explainers

- **TreeExplainer** optimisation is based on feature dependence properties of tree-based models.

  It is faster than **KernelExplainer** and the explanations are similar on the model, so it is the best to use for random forests

- **DeepExplainer** is a combination of SHAP and DeepLift optimized for Neural Network models.

  DeepExplainer is faster than **KernelExplainer,** but the explanations are different, so we analyze both further on the neural network

Computation time

LIME and SHAP algorithms

LIME and SHAP have 3 computation stages

- Creation of the explainer from training values
- Computation of the explanation for test examples
- Eventual plotting of the values

For LIME, we will plot the creation time of the explainer, as the computation time is not evolving while changing parameters.

For SHAP, we will plot the computation time of the values, as it increases while changing the parameters.

SHAP

SHAP computation has 3 stages :

- Creation of the explainer from training values
- Computation of the SHAP values for test examples
- Eventual plotting of the values

We will plot the computation time of the SHAP values, as the creation of the explainer is very fast, and the plotting of the values is not recommended for a large scale.

Explainers : TreeExplainer for Random Forests and DeepExplainer for NN.

LIME

LIME computation has 3 stages :

- Creation of the explainer from training values
- Computation of the explanation for the exemple
- Eventual plotting of the values

We will plot the creation time of the explainer, as the computation time is not evolving while changing parameters, and the plotting of the values is not recommended for a large scale.

Random Forests

The computation time of an instance does not increase with the size of the test set for SHAP and LIME :

LIME mean computing time : 23 ms by instance

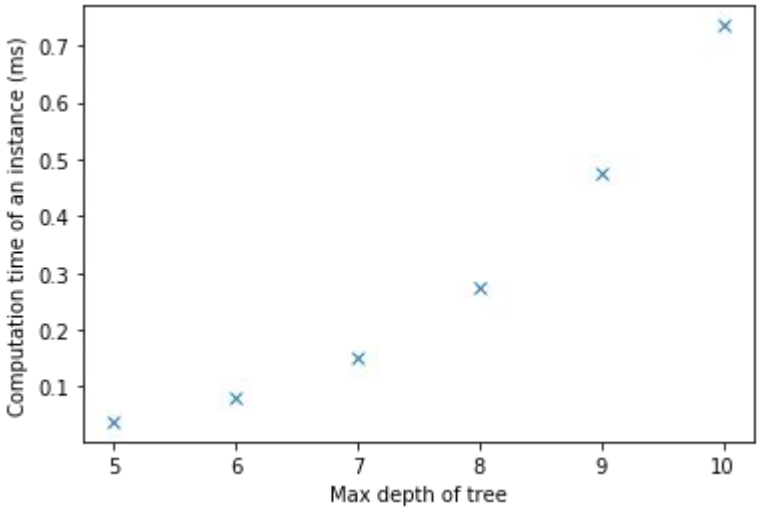
SHAP (TreeExplainer) mean computing time : 0.27 ms by instance

Parameters :

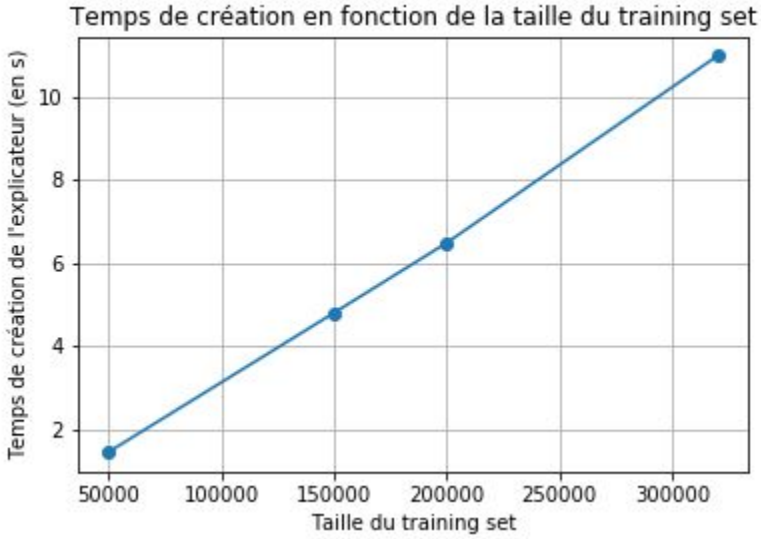
- Xtrain = 247780
- Xtest=122041
- max_depth=8
- split = 33%

Random Forests

Tree depth
SHAP TreeExplainer



Train dataset size
LIME Explainer



Neural Network

The computation time of an instance does not increase with the size of the test set for SHAP and LIME :

LIME mean computing time : 1.6 s by instance

SHAP mean computing time : DeepExplainer : 70 ms by instance, KernelExplainer : 3-6s by instance

Parameters :

- Xtrain = 247780
- Xtest=122041
- split = 33%
- features = 115

Number of features

The number of features is also to take into account :

For the neural network with usual parameters:

Using 22 features (without one hot encoding of categorical values)

→ Computing time by instance for SHAP DeepExplainer : 18 ms (vs 70 ms for 115 features)

→ Computing time by instance for LIME : 23 ms (vs 1.63 s for 115 features)

→ Creating time by instance for LIME : 2s (vs 11s for 115 features)

Usability of LIME and SHAP

- LIME computing time is constant for all the types of execution (except the number of features for NN), while the creation time increases with the training set size for both models.

- SHAP explainer creation time is constant, but the computing time increases with the size of the model.
- SHAP TreeExplainer is faster than LIME with similar results
- SHAP DeepExplainer is faster than LIME but results are different
- SHAP KernelExplainer cannot be used for real time

Analyses

Predictions evaluation

Based on the rules RCS1 - RCS2 - RCB1 - RCB2 - RCB4 we created a score between -1 and 1 to assess whether the explanation meets the criteria for the creation of the fraud.

Random Forest :

LIME

Analyse	RCS1	RCS2	RCB1	RCB2	RCB4
Simplifiée	0.5	1	0.4	0.43	0.75
Complète	0.5	1	0.27	0.25	0.72

SHAP

Analyse	RCS1	RCS2	RCB1	RCB2	RCB4
Simplifiée	-	1.0	0.65	0.47	0.96
Complète	-	1.0	0.25	0.17	0.54

Predictions evaluation

Neural network :

LIME

Analyse	RCS1	RCS2	RCB1	RCB2	RCB4
Simplifiée	0.5	-0.05	0	0.14	0
Complète	0.5	0	-0.18	-0.07	0

SHAP

Here the result for the $DeepExplainer$.

Analyse	RCS1	RCS2	RCB1	RCB2	RCB4
Simplifiée	1.0	1.0	0.55	0.22	0.65
Complète	1.0	1.0	0.18	0.17	0.65

Here the result for the $KernelExplainer$.

Analyse	RCS1	RCS2	RCB1	RCB2	RCB4
Simplifiée	1.0	1.0	0.63	0.29	0.69
Complète	1.0	1.0	0.27	0.21	0.69

Comparison with a Decision Tree

A decision tree is an explainable model. We compared the decision path to the explanation in order to score their similarity.

LIME	RCS1	RCS2	RCB1	RCB2	RCB4
Explication	1.0	-	0.6	0.33	0.75
Chemin	1.0	0.5	0.6	0.49	0.75

SHAP	RCS1	RCS2	RCB1	RCB2	RCB4
Explication	1.0	0.5	0.8	0.5	0.75
Chemin	1.0	0.5	0.6	0.49	0.75

Conclusion

Conclusion

- SHAP seems to be more robust than LIME, and its repo is much more active on Github
- Next steps in the study : To create an explainer optimised for the neural network in order to have real time explanation

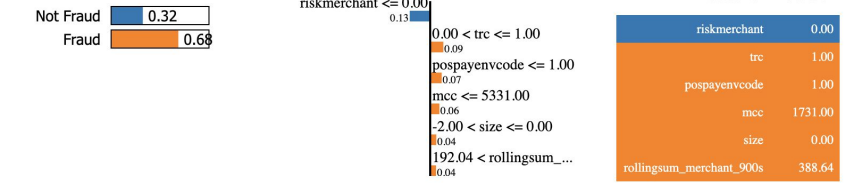
Annexes

Annexes

RF LIME : Confusion matrix

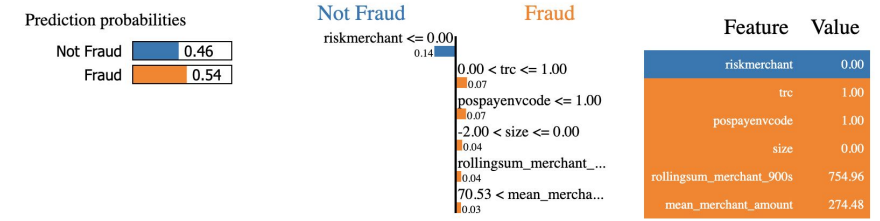
Vrai positif : 4 516

```
fraud    1
Name: 541, dtype: int64
```



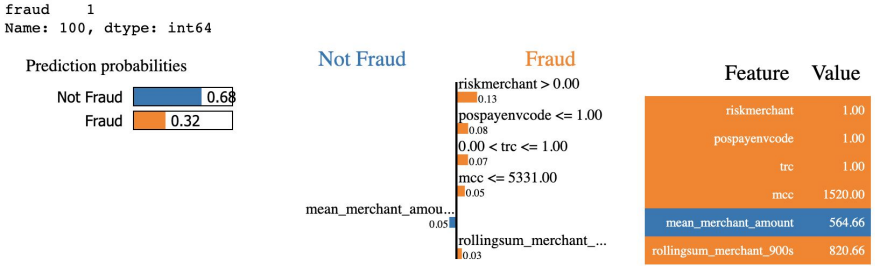
Faux positif : 253

```
fraud    0
Name: 8951, dtype: int64
```



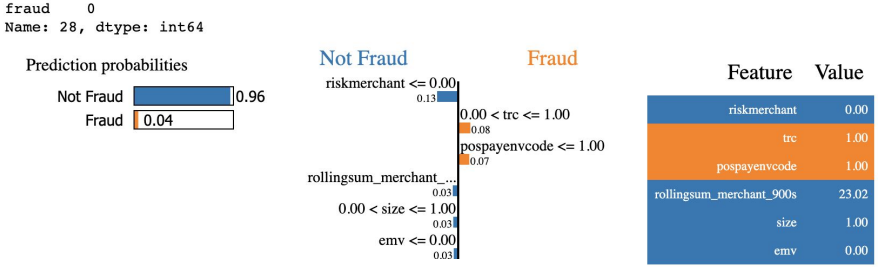
Faux négatif : 6 995

```
fraud    1
Name: 100, dtype: int64
```



Vrai négatif : 110 277

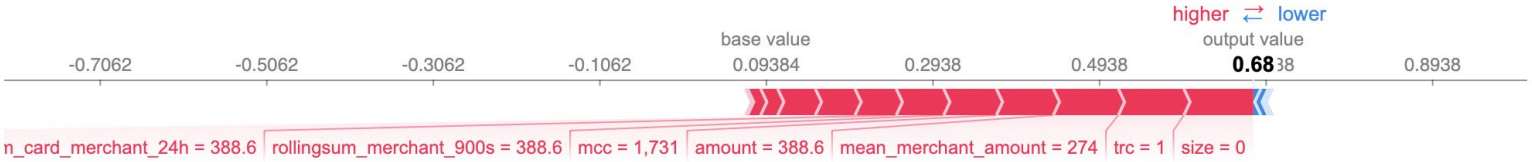
```
fraud    0
Name: 28, dtype: int64
```



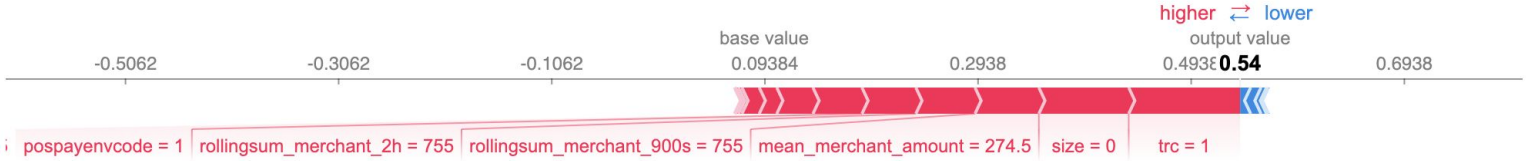
Annexes

RF SHAP : Confusion matrix

Vrai positif
4 516

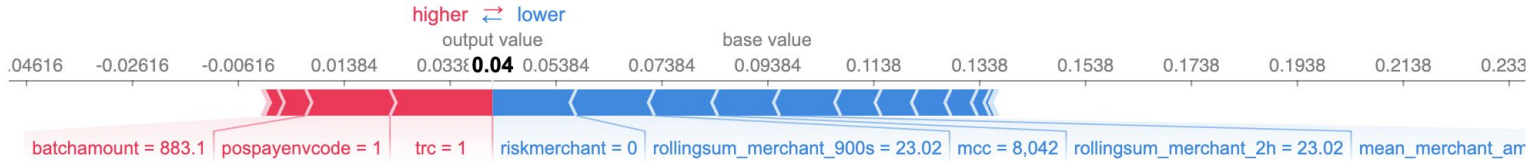


Faux positif
253



Faux négatif
6 995

Vrai négatif
110 277

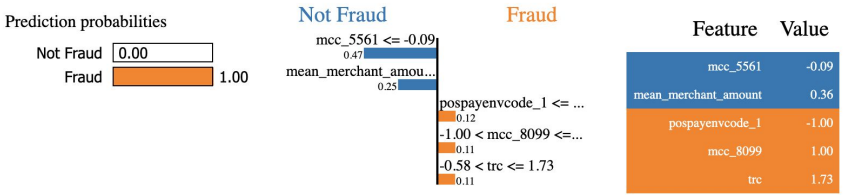


Annexes

NN LIME : Confusion matrix

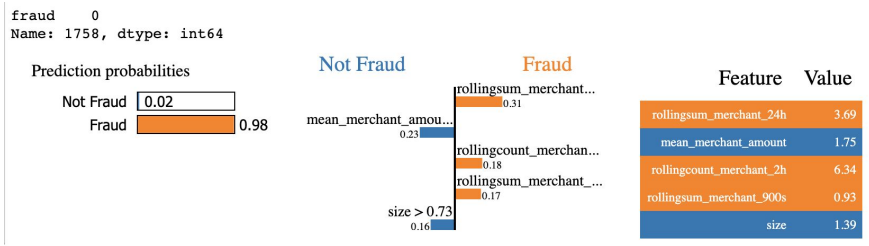
Vrai positif : 11 303

```
fraud    1
Name: 122024, dtype: int64
```



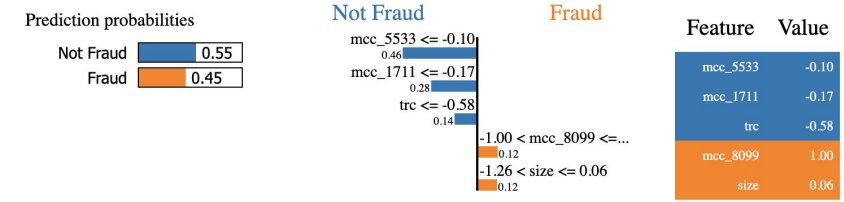
Faux positif : 181

```
fraud    0
Name: 1758, dtype: int64
```



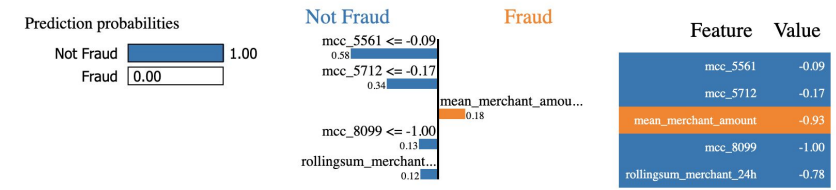
Faux négatif : 208

```
fraud    1
Name: 119811, dtype: int64
```



Vrai négatif : 110 349

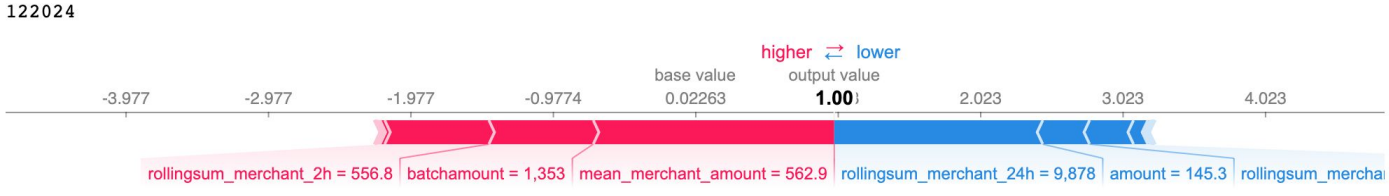
```
fraud    0
Name: 6, dtype: int64
```



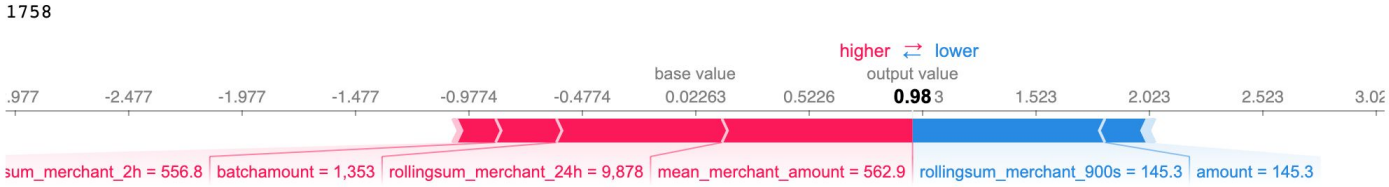
Annexes

NN SHAP : Confusion matrix

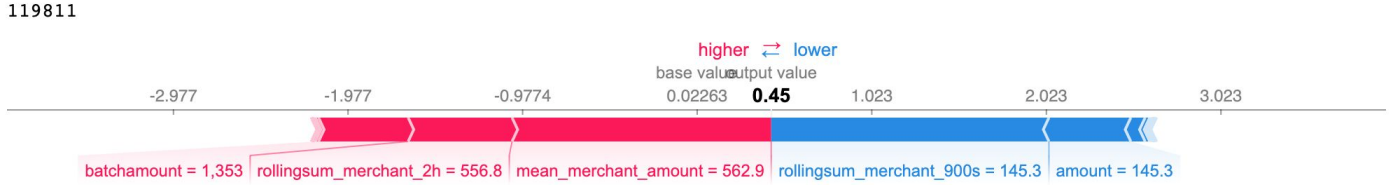
Vrai positif
11 303



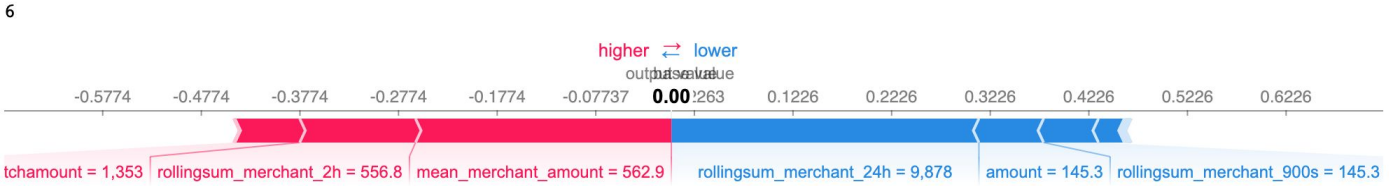
Faux positif
181



Faux négatif
208



Vrai négatif
110 349



Back

RCB1	RCB2	RCB4
$pospayenvcode == 1$	$pospayenvcode == 1$	$pospayenvcode == 1$
$size < 1$	$size < 1$	$size < 1$
$RSM\_2h >= 2000$ ou $RCM\_2h >= 5$	$RSM\_900s >= 200$ ou $RCM\_900s >= 3$	$RSM\_24h >= 5000$ ou $RCCM\_600s >= 4$
$mcc! = 7991$	$super\_market\_list == 0$	$mean\_merchant\_amount < 1000$
$list\_bankid\_b == 0$	$fuel\_list == 0$	
	$medic\_list == 0$	

RCS1	RCS2
$riskmerchant == 1$	$riskmerchant == 1$
$foreignbin == 1$	$trc == 1$