Ecole CentraleSupélec
Option: Applied Mathematics, 2019-2020

# Application of machine-Learning techniques to XVA

Achraf Dridi

Supervisor: Jeremy POIROT



**Acknowledgements**

I would like to express my gratitude to my supervisor Jeremy Poirot for his help and his guidance during the different stages of my internship. His willingness to give his time so generously has been very much appreciated.

I would also like to extend my thanks to my fellow colleagues from the MRM team at Natixis, and more particularly the XVA team members, for their continuous support and their interest in my research subject.

# Contents


**1 Introduction** **4**
- 1.1 Introduction to xVA . . . 4
- 1.2 Context of the internship . . . 4

**2 Credit Value Adjustment** **5**
- 2.1 Definitions and formulas . . . 5
- 2.2 CVA calculation in practice . . . 6
- 2.3 CDS and default probabilities . . . 7
  - 2.3.1 Credit default swaps . . . 8
  - 2.3.2 Bootstrapping the survival probability function . . . 9

**3 CDS rate construction** **10**
- 3.1 Shortage of Liquidity problem and need for a new proxy . . . 10
- 3.2 A classification problem . . . 12
  - 3.2.1 Features and data . . . 12
  - 3.2.2 Classification . . . 16
  - 3.2.3 Backtesting . . . 22

**4 Neural network for CVA: Learning Future Values** **25**
- 4.1 Context and model description . . . 25
- 4.2 Model methodology . . . 25
  - 4.2.1 Standard Bermudan Swaptions . . . 26
  - 4.2.2 Description of the proposed short rate model: Vasicek model . 26
  - 4.2.3 Future-Mark-to-Market distribution calculation . . . 27
  - 4.2.4 Neural Networks . . . 28
- 4.3 Model implementation . . . 30

**5 Conclusion** **37**

# 1 Introduction

## 1.1 Introduction to xVA

The valuation of derivative products on the financial markets has long been based on the absence of arbitrage opportunities and complete markets assumptions, but these hypotheses are rarely realistic, especially over short-term periods, "The Law of One Price" is then starting to be seriously questioned by practitioners. More particularly concerned are OTC products, characterized often by the absence of a central counterparty (CCP).

During the financial crisis of 2008 many financial institutions failed, leaving their counterparties with claims on derivative contracts that were paid only in part. Therefore counterparty credit risk must also be considered in derivatives valuation, and the risk neutral value is then adjusted correspondingly.

The current valuation framework is therefore more and more detached from the assumptions of this law of unique price. It aims to take into account a wider range of risks to be minimized by setting up valuation adjustments (xVAs) in order to make pricing more realistic and broader than the risk-neutral framework. The purpose of these adjustments is twofold: primarily to hedge for possible losses due to other parties' failures to pay amounts due on the derivative contracts; but also to determine (and hedge) the amount of capital required under the bank capital adequacy rules.

In a nutshell, we can write the **Full economic value** of a product as follows:

- Full economic value = Reference value + xVAs
- Where Reference value = Classical pricing theory assuming neither the issuing bank nor the counterparty can default, and that the issuing bank may freely borrow and lend at some specified reference rate of interest.

This Framework is still in the process of being constructed, although definitions and methods of calculating adjustments such as Credit Value Adjustment (CVA), Debit Value Adjustment (DVA) and Funding Value Adjustment (FVA) tend to converge towards a stable state and acquire an importance and unanimous legitimacy among the actors.

## 1.2 Context of the internship

The Valuation Model Validation group is responsible for independently validating the quantitative financial models that are used in Natixis. The work will involve

theoretical mathematical analysis, model implementation and coding and documentation of the work for management review.
The work would consists in a review of the different applications of machine learning to quantitative finance with an emphasis on application to XVA. This work focuses on the calculation of the CVA and concerns two main topics: **CDS rate construction methods by Machine Learning Technique** and **Neural network for CVA: Learning Future Values**. These topics are described in the following sections.

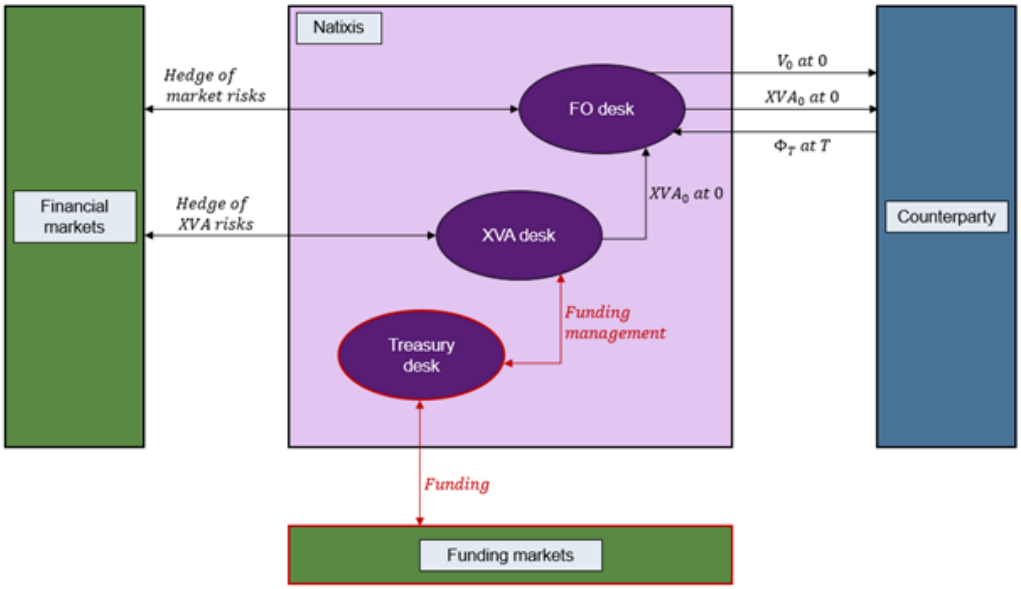


Figure 1: FO structure for a derivative contract

# 2 Credit Value Adjustment

In this section, I will try to relate the subjects that I worked on to the XVA's calculation context. To do this, I will make use of some information extracted from a book of Andrew Green, "XVA: Credit, Funding and Capital Valuation Adjustments" [1].

## 2.1 Definitions and formulas

Let's consider the following context:

- Natixis enters into a long position on a payoff $\Phi_T$ at T.
- The deal is collateralised with a collateral amount $C_t$, $\forall t \in [0, T]$.
- The collateral remuneration rate is $r_t^c$, $\forall t \in [0, T]$.

- The reference value of the deal is $V_t$, $\forall t \in [0, T]$.
- All amounts are considered from Natixis' point of view, i.e. positive when received and negative when paid.
- The default time of the counterparty (resp. Natixis) is $\tau^c$ (resp. $\tau^n$).
- The recovery rate of the counterparty (resp. Natixis) is $R^c$ (resp. $R^n$).

By definition:

$$\begin{aligned} CVA_0 = & \text{ spot pricing adjustment to include the counterparty's default risk} \\ & = price_0(\text{considering the counterparty' s default risk }) – price_0(\text{risk free}) \\ & = price_0 \text{ (Losses due to the counterparty' s default)} \end{aligned}$$

On the other hand:
Losses due to the counterparty' s default $= -(1 - R^c).(V_{\tau^c} - C_{\tau^c})_+.1_{\tau^c<T}$ at $\tau^c$

By conditioning w.r.t the default time variable $\tau^c$, the CVA formula is given by:

$$CVA_0 = -(1 - R^c) \cdot \int_0^T B(0,t) \underbrace{E^{Qt}\left[(V_t - C_t)_+\right]}_{\text{Expexted positive exposure}} \cdot dQ\left(\tau^c \leq t\right)$$

Comments on the formula:

- Independence between the counterparty's default and the remaining risk factors.
- Constant recovery.
- The CVA is negative and, hence, is a cost.
- Perfect collateralisation $\rightarrow \forall t \in [0, T] : C_t = V_t \rightarrow CVA_0 = 0$.
- Counterparty of a high credit quality $\rightarrow$ low CVA cost.

## 2.2 CVA calculation in practice

If we consider $R^c$, the rate curves and the credit curves as given. The challenge is to compute the expected positive exposures ($EE_t^+$ ) for all dates $t \in [0, T]$.

- Oftentimes, $C_t = f(V_t)$, where f is a deterministic function.
- Therefore, the CVA calculation boils down to the estimation of the distribution of the future prices $V_t$ (also called 'mark-to-futures') for all dates $t \in [0, T]$.

The integral term is discretised as follows:

$$CVA_0 = -\left(1 - R^c\right) . \sum_{k=1}^{N} B\left(0, t_k\right) \cdot \underbrace{E^{Q_{t_k}}\left[\left(V_{t_k} - f\left(V_{t_k}\right)\right)_+\right]}_{\text{Expexted positive exposure}} \cdot Q\left(t_{k-1} \leq \tau^c \leq t_k\right)$$

The main calculation approach in practice is the Hybrid Monte Carlo framework:

- Cross asset diffusion of risk factors (e.g. HW2F for interest rates).
- For each scenario, and each exposure date $t_k$, the price $V_{t_k}$ is computed:
  - Closed formulas for simple payoffs (e.g. European swaptions).
  - Numerical approximations for complex payoffs to avoid nested Monte Carlo schemes (e.g. Longstaff-Schwartz regression for Autocalls).
- Estimation of $EE^+_{t_k}$ and hence of the CVA.

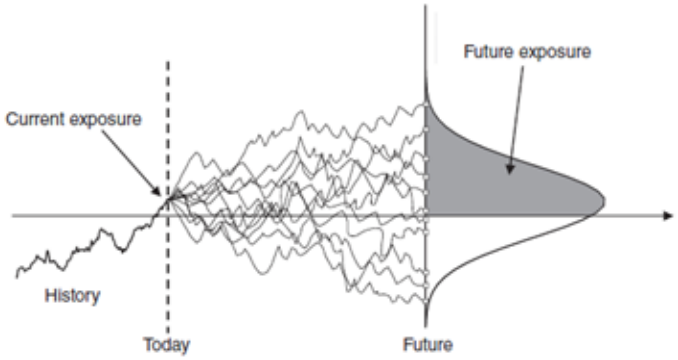


Figure 2: Hybrid Monte Carlo framework for future exposure

In the above description, we assumed that the different default probabilities are given, however, these latter are built using the CDS curves. The principle of this framework will be explained in the next subsection.

## 2.3 CDS and default probabilities

The survival probability function is central to the calculation of CVA and this chapter presents a description of how this function can be obtained. In general, for CVA calculations the survival probability function is obtained from credit default swap spreads. The standard reduced form model for the default process is as the first jump of a Poisson process with a deterministic hazard rate $\lambda(t)$. Hence under this model the survival probability is given by

$$P(\tau > T) = \exp\left(-\int_t^T \lambda(s)\mathrm{d}s\right)$$

, and the default probability is given by

$$P(\tau \leq T) = 1 - \exp\left(-\int_t^T \lambda(s)\mathrm{d}s\right)$$

### 2.3.1 Credit default swaps

Credit default swaps are insurance-like derivatives that are designed to offer protection against the default of a reference obligation. They allow the owner of a bond to buy default protection so that in the event of a default by the bond issuer they are compensated by the protection seller.

The CDS consist of two legs, a premium leg and a default leg. The premium leg consists of a stream of payments from the protection buyer to the protection seller until the default of the reference obligation or maturity. The protection (or default) leg consists of a compensating payment $[(1 - RecoveryRate) \times notional]$ from the protection seller to the protection buyer on the default of the reference obligation. The recovery rate is usually set through an auction process administered by ISDA. These flows are illustrated in Figure 3.

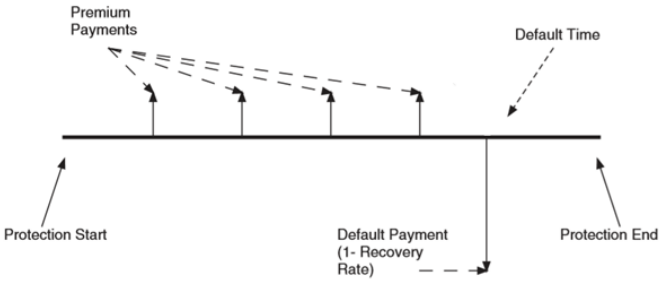


Figure 3: The cash flows of a credit default swap

The premium or fee leg is simply the value of the fee conditioned on the survival of the reference entity to the payment date of the fee,

$$V^i_{\text{premum}}(t_0) = \mathbb{E}\left[\exp\left(-\int_{s=t_0}^{t_i} r(s)ds\right) C_i \alpha_i 1_{\{\tau > t_i\}} \mid \mathcal{F}_t\right]$$

where $C_i$ is the premium paid at time $t_i$ and $\alpha_i$ is the day-count fraction (normally 30/360).
Under the market standard assumption of independence between rates and credit, the above expression reduces to

$$V^i_{\text{premium}}(t_0) = C_i \alpha_i B(t_0, t_i) P(\tau > t_i)$$

where $B(t_0, t_i)$ is the value of the discount bond maturity at time $t_i$. Hence, the value of the premium leg is given by

$$V_{\text{premium}}(t_0) = \sum_{i=1}^{n} C_i \alpha_i B(t_0, t_i) P(\tau > t_i)$$

The protection or default leg pays $(1 - R)$ on the default of the reference name so the value is given by

$$V_{prot}(t_0) = \mathbb{E}\left[\exp\left(-\int_{t_0}^{\tau} r(s)ds\right)(1-R)1_{\{\tau<T\}}\right]$$
$$V_{prot}(t_0) = (1-R)\int_{t_0}^{T} \lambda(s)\exp\left(-\int_{t_0}^{s}(\lambda(v)+r(v))dv\right)ds$$

To evaluate the expectation we adopt an approach where the interval [t, T] is partitioned into small segments such that $t = u0 \leq u1 \leq \ldots \leq uk \leq \ldots \leq up = T$, giving

$$V_{prot}(t_0) = (1-R)\sum_{k=1}^{p}\int_{u_{k-1}}^{u_k} \lambda(s)\exp\left(-\int_{t_0}^{s}(\lambda(v)+r(v))dv\right)ds$$

Assuming the CDS short and hazard rates are constant over each time partition segment, the previous equation is simplified to a sum:

$$V_{\text{prot}}(t_0) = (1-R)\sum_{k=1}^{p} B(t_0, u_k) P(\tau > u_k) \lambda_k \left[-\frac{\exp(-(\lambda_k + r_k)(v - u_k))}{(\lambda_k + r_k)}\right]_{u_{k-1}}^{u_k}$$
$$= (1-R)\sum_{k=1}^{p} \frac{\lambda_k}{\lambda_k + r_k}\left[B(t_0, u_{k-1}) P(\tau > u_{k-1}) - B(t_0, u_k) P(\tau > u_k)\right]$$

Hence the value of a credit default swap from the perspective of the protection buyer is given by

$$V_{CDS} = V_{premium} - V_{prot}$$

### 2.3.2 Bootstrapping the survival probability function

Having obtained an expression for the value of a credit default swap in terms of the survival probability function, $P(\tau > t)$, it is possible to use this to bootstrap this function from a set of CDS spreads. Typically CDS spreads will be available with maturities at 6M, 1Y, 2Y, 3Y, 5Y, 7Y, 10Y, 20Y, 30Y, although liquidity is frequently very limited in longer maturity CDS contracts. Once the form of the survival probability function is chosen it can be calibrated using equation of the CDS value.

Given the standard form of the survival probability function, $P(\tau > T) = \exp\left(-\int_t^T \lambda(s)\mathrm{d}s\right)$, CDS and Default Probabilities parametrisation depend on the choice of functional form for the hazard rate $\lambda(t)$. Three choices of parametrisation present themselves:

- Piecewise constant
- Piecewise linear
- Cubic spline

The standard model most frequently used in the CDS market is the piecewise constant hazard rate model. This approach has been adopted for a number of reasons: Piecewise constant hazard rates have been found to be numerically stable and behave better than alternatives such as piecewise linear. There is no information on the hazard rate between CDS maturities. Piecewise constant hazard rates provide the simplest possible assumption on behaviour between market observations. The segments of the hazard rate function are chosen to match the maturity dates of the CDS spreads in the curve, so for the maturities listed above the segments would be, 0–6M, 6M– 1Y, 1Y–2Y, 2Y–3Y, 3Y–5Y, 5Y–7Y, 7Y–10Y, 10Y–20Y, 20Y–30Y. Each segment can be fitted using successive CDS spreads, with the hazard rate segment chosen to ensure the CDS has zero value.

# 3 CDS rate construction

## 3.1 Shortage of Liquidity problem and need for a new proxy

As we have already seen, the calculation of the CVA requires the Credit Default Swap quotes to bootstrap the default probabilities of the different counterparties. However, according to the European Banking Authority's survey (EBA, 2015), over 75 % of the counterparties do not have liquid CDS quotes. The Shortage of Liquidity problem is defined as the problem of assessing the default-risk of counterparties without liquid CDS quotes and to solve this problem we need CDS Rate Construction Methods, which are methods used to construct the missing quotes based on available market data, and the CDS data obtained for the non-liquid counterparty is called a proxy. According to regulators' publications, any CDS Proxy Method needs to satisfy the following criteria:

1. The algorithm used for the CDS Proxy Method must discriminate at least three types of variables: Credit Quality (e.g., rating), Industry Sector and Region (BCBS, 2015).

2. Both the Observable Counterparties used to construct a CDS proxy spread for a given Nonobservable and the Nonobservable itself have to come from the same Region, Sector and Credit Quality group (BCBS, 2015).

3. According to (EBA, 2013), the proxy spread must reflect available credit default swap spreads. In addition, the appropriateness of a proxy spread should

be determined by its volatility and not by its level. This criterion highlights the regulators' requirement that CDS proxy rates should include counterparty-specific components of counterparty default risk.

Actually, there are two types of CDS Proxy Methods that are currently used by the finance industry:

- The Credit Curve Mapping Approach: This method consists in creating Region/Sector/Rating buckets of single-name CDS contracts with underlying counterparties from the same regions and sectors and with the same ratings, and then proxies the CDS rates of a Nonobservable from a given Region/Sector/Rating bucket by the mean or median of the spreads of the single-name CDS rates within that bucket.

- The Cross-sectional Regression Approach: this CDS Proxy Method explains a counterparty i's CDS rate $S_i$ by the following log-linear regression equation:

$$\log(S_i) = \beta_0 + \sum_{m=1}^{\text{Regions}} \beta_m^R I_{i,m}^R + \sum_{m=1}^{\text{Sectors}} \beta_m^S I_{i,m}^S + \sum_{m=1}^{\text{ratings}} \beta_m^r I_{i,m}^r + \sum_{m=1}^{\text{seniorities}} \beta_m^s I_{i,m}^S + \epsilon_i$$

  where the I's are dummy or indicator variables for, respectively, sector, region, rating class and seniority, as specified in the CDS contract. To get the regression-coefficients, we apply the Ordinary Least Squares method on Observable counterparties data and then we will be able to predict the rate of a CDS-contract of a Nonobservable in a given region and sector with a given rating and seniority.

As we have seen above, current proxy methods seek to directly construct the missing rates, based on Region, Sector and Ratings data. The two approaches clearly satisfy criteria 1 and 2 but assumes the counterparty default risks for all counterparties coming from the same Region, Sector and Rating bucket to be homogeneous and ignores the idiosyncratic default risk specific to individual counterparties. Thus, they ignore the CDS-spread volatility across counterparties within a bucket, thereby failing to meet criteria 3.

One of the motivations of this work is to provide an alternative to the few currently existing CDS Rate Construction Methods that can meet the criteria 3. One recommended solution-method is to map a counterparty lacking liquidly quoted CDS spreads to one for which a liquid market for such quotes exists, and which, according to criteria based on financial market data, best resembles the non-quoted party. The CDS data of the liquid counterparty, which is called a proxy of the non-liquid counterparty, can subsequently be used to obtain estimates for the risk-neutral default probabilities of the latter. This approach treats the Shortage of Liquidity problem

as a Machine Learning problem and construct CDS Proxies using classification algorithms. This work is based on the results of a previous work done by Raymond Brummelhuis and Zhongmin Luo [2], which investigated different classifiers on a number of US entites' data, and we are going to follow the same approach with Natixis counterparties.

## 3.2 A classification problem

### 3.2.1 Features and data

#### 3.2.1.1 Features presentation :

$$\mathbf{x} = \left(PD_{6m}, PD_{1y}, PD_{2y}, PD_{3y}, PD_{4y}, PD_{5y}, \sigma^{imp}_{3m}, \sigma^{imp}_{6m}, \sigma^{imp}_{12m'}\sigma^{imp}_{18m'}\sigma^{h}_{1m'}\sigma^{h}_{2m'}\sigma^{h}_{3m'}\sigma^{h}_{4m'}\sigma^{h}_{6m}\right)$$

- $PD_{6m}, PD_{1y}, PD_{2y}, PD_{3y}, PD_{4y}, PD_{5y}$ denote the counterparty's probabilities of default over, respectively, a 6-month, 1-year, 2-year, 3-year, 4-year and 5-year time-horizon.
  These default probabilities are obtained from BloombergTM which uses a model that we don't know a lot about. This model integrates different types of inputs, it takes into consideration counterparties' accounting data for example.

- $\sigma^{imp}_{3m}, \sigma^{imp}_{6m}, \sigma^{imp}_{12m'}\sigma^{imp}_{18m'}$ are the counterpart's at-the-money implied volatility as computed from European call options on its equity with, respectively, a 3-month, 6-month, 12-month and 18-month maturity.
  In fact, studies[3] showed that option-implied volatility is an important determinant of CDS spreads. More importantly, the volatility risk premium embedded in option prices covaries with the CDS spread.

- $\sigma^{h}_{1m'}\sigma^{h}_{2m'}\sigma^{h}_{3m'}\sigma^{h}_{4m'}\sigma^{h}_{6m}$ denote the historical volatilities estimated from 1-year historical equity price returns for terms of 1-month, 2-month, 3-month, 4-month and 6-month respectively.

#### 3.2.1.2 Data collection :

We constructed our data sample based on the observations taken during the period going from 02/03/2020 to 08/07/2020 in a "stressed" economic environment. Our aim is to assess the validity of the proposed CDS-proxy method in such an environment.

A difficult part of the work was to collect the data. In fact, in order to import the data from Bloomberg we need to have for each entity, a Corp Ticker to get the default probabilities and an Equity ticker to get the volatilities. After some cleaning

and manual verifications, we went from 1600 counterparties to 150 ones that have all the necessary information and constructed the following data base.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | CodeTiersNatixis | Libellé Bloomberg | CorpTicker | EquityTicker |
| 2 | DEUTBAUFT | Aareal Hyp AG - Covered | AARB CorpTicker | ARL GR Equity |
| 3 | ABERTISBA | Abertis Infraestructuras SA | ABESM CorpTicker | ABE SM Equity |
| 4 | ABNAMRO2 | Fortis Bank Nederland NV | ABNANV CorpTicker | ABN NA Equity |
| 5 | ACCORPA | Accor SA | ACFP CorpTicker | AC FP Equity |
| 6 | FORTISAG | AG Insurance SA | AGSBB CorpTicker | AGS BB Equity |

Figure 4: Tickers data base

Hence, we were able to obtain the features values for each of the 150 entities, which made it 93 lines (93 business days) and 15 columns (number of features) for each entity.

However, the study is conducted only on the liquid counterparties (which are 51 entities) as the data available for the non-liquid ones are not useful but for the final prediction.
The next step is then the baskets construction. Here we define the baskets within 2 criteria: the geographical area and the industrial sector so that 51 entities are distributed on 11 baskets which are the following:

| REGION_SECTOR |
|---|
| Europe_ConsumerServices |
| Europe_Industrials |
| AsiaExclJapan_Financials |
| Europe_Financials |
| NorthAmerica_Industrials |
| Europe_ConsumerGoods |
| Europe_Utilities |
| Europe_Government |
| Europe_TelecommunicationsServices |
| Europe_Energy |
| Europe_Healthcare |

Figure 5: Region$_s ectorbaskets$

Among the 51 entities, 22 are in the **Europe-Financials** basket so we decided that we will focus on this basket and the study will be based on its entities.

#### 3.2.1.3 Data visualization and preprocessing :

The first step in a machine learning problem is to take a look on the data we have. We began by visualizing some scatter plots of the features.

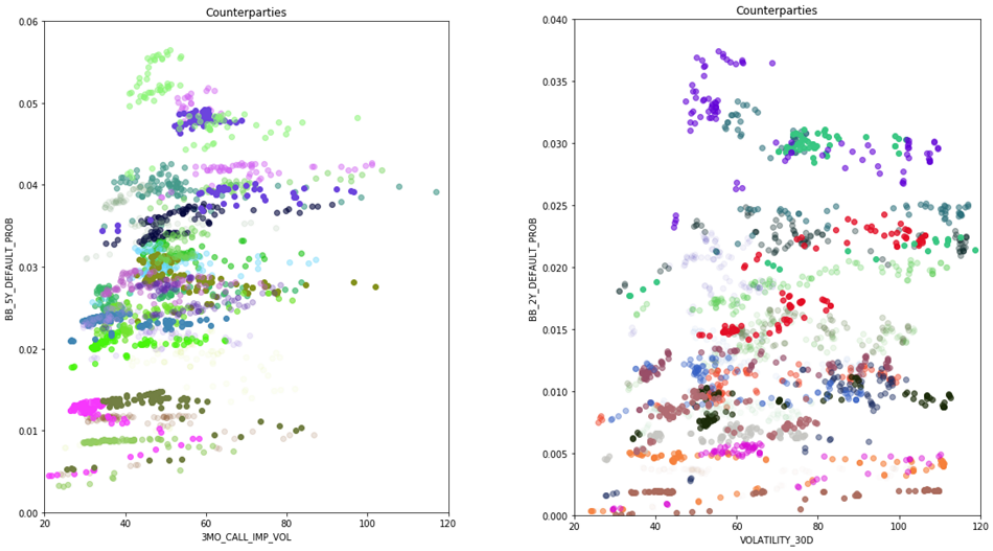


Figure 6: Scatter plots of features

We can see that we can differentiate with the naked eye between the different classes which strengthens our motivation to treat the topic as a classification problem.

We plotted then the distribution of the different features and we noticed that the volatilities distributions are close to the gaussian distribution, however, this is not the case with the default probabilities distributions even after applying some classical transformations of the data.

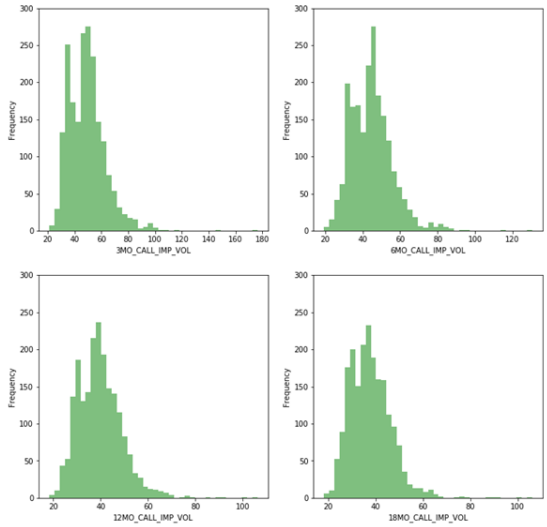


Figure 7: Implied volatilities distributions

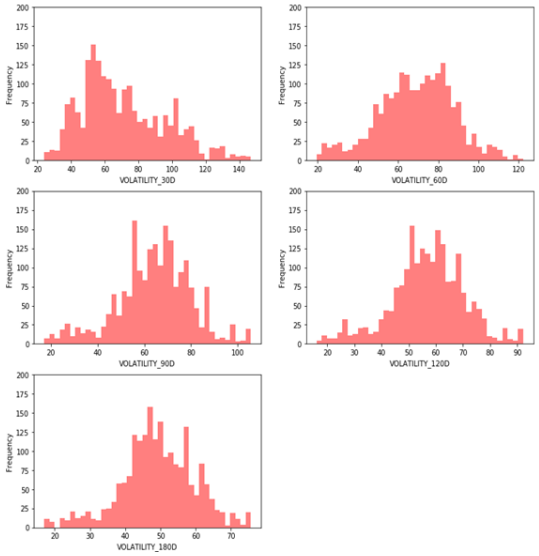


Figure 8: Historical volatilities distribution

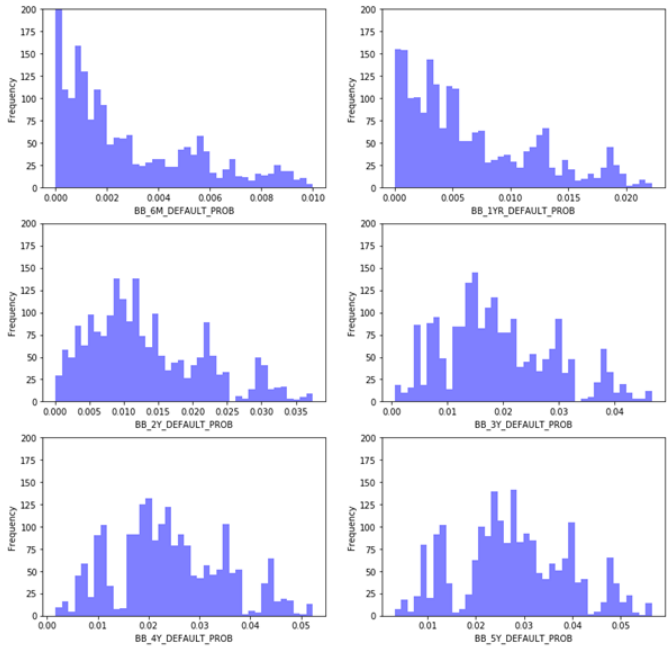


Figure 9: Default probabilities distributions

Some machine learning algorithms require all variables to be in the same range to function properly, or they will tend to pay more attention to certain variables rather than others. An example of such algorithms are distance-based algorithms. In normalizing the entities at the same distance, we make sure that the algorithm treats them with the same importance.

Thus, we decided to standardize features by removing the mean and scaling to unit variance. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. To do this we used the StandardScaler function available in the sklearn python library.

### 3.2.2 Classification

In this section, we are going to test a number of classification algorithms that we can use for our CDS Proxy construction. Based on the results of [2], we chose to work with 3 different classifiers which are: Random Forest, Support Vector Machine and K- Nearest Neighbours.

#### 3.2.2.1 Classification Models :

As Classification Techniques are not well-known in the Finance industry, we will begin by presenting these classifiers and their basic concepts.

- **Random Forest:**
  The "Random Forest" algorithm (sometimes also translated as a forest of decision trees ) is a classification algorithm that reduces the variance of forecasts from one decision tree alone, thus improving their performance. For this, it combines many decision trees in a bagging-type approach.
  In its most classic formula, it performs parallel learning on multiple decision trees randomly constructed and trained on different data subsets. The ideal number of trees, which can go up to several hundred or even more, is an important parameter: it is very variable and depends on the problem. Concretely, each tree of the random forest is trained on a random set of data according to the bagging principle, with a random subset of features. The predictions are then used for a majority vote in the case of classification trees. The random forest algorithm is known to be one of the most efficient "Out-of-the-box" classifiers (i.e. requiring little data preprocessing).

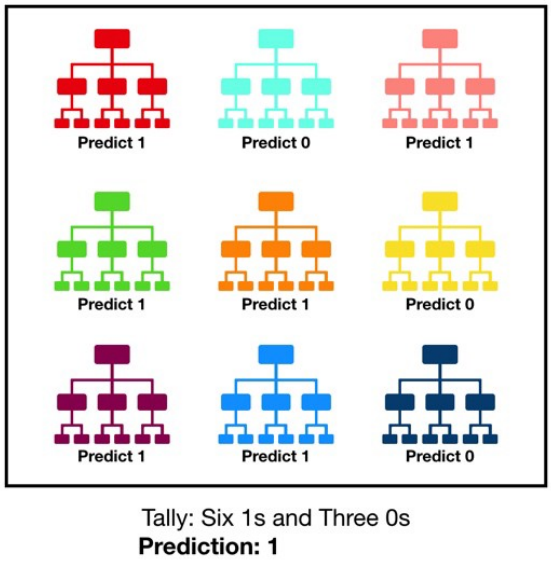


Figure 10: Random Forest illustration

Among the most important parameters of Random Forest that we have chosen to adjust:

1. **n-estimators:** number of trees in the forest.
2. **max-features:** maximum number of features taken into account for splitting a node.
3. **max-depth:** maximum number of levels in each decision tree.

- **Support Vector Machine:**
  The objective of the support vector machine algorithm is to find an hyperplane in an N-dimensional space(N is the number of features) that distinctly classifies the data points.
  To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.
  Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier.

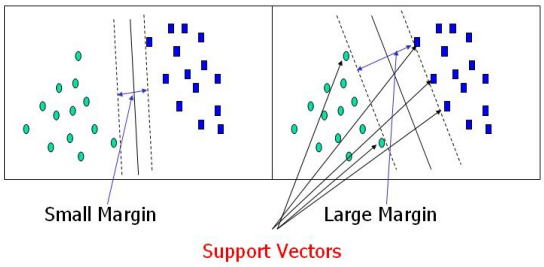


Figure 11: Support Vector Machine illustration

- **k-Nearest Neighbours:**
  In k-NN classification, an object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small).
  Thus, for a given feature vector x we compute all the distances $d(x, x_i)$ for $(x_i, y_i) \in D^T$, where the metric d can be any metric of one's choice on the feature space $R^d$, such as the Euclidean metric. Then we rank order the distances and select the k nearest neighbours of $x$ amongst the $x_i$ and finally we classify $x$ to that element which occurs most often amongst the $y_i$ for which $x_i$ is a neighbour.
  A useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor.

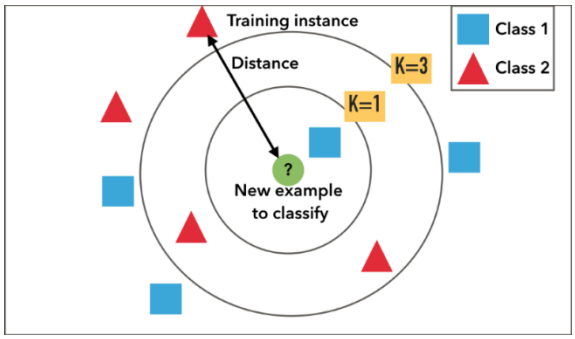


Figure 12: k-Nearest Neighbours illustration

#### 3.2.2.2 Cross-Validation :

To assess the quality of our model in relation to over-fitting, it is necessary to carry out a cross-validation protocol which will quantify its robustness. This protocol, associated with a grid search, allows to have for each classifier the best parameters to use.

We used the K-fold Cross Validation method. In this method, the data is divided into k subsets. The hold method is repeated k times, so that one of the k subsets is used as a test set and that the other k-1 subsets are grouped together to form a learning set. The error estimate is averaged over all k trials to obtain the total efficiency of our model. Hence, a tuning of the parameters will give us the best ones to use.

We took a value of 5 for the coefficient K.

```
RandomizedSearchCV(cv=StratifiedShuffleSplit(n_splits=5, random_state=42, test_size=0.2,
            train_size=None),
                   error_score=nan,
                   estimator=RandomForestClassifier(bootstrap=True,
                                                    ccp_alpha=0.0,
                                                    class_weight=None,
                                                    criterion='gini',
                                                    max_depth=None,
                                                    max_features='auto',
                                                    max_leaf_nodes=None,
                                                    max_samples=None,
                                                    min_impurity_decrease=0.0,
                                                    min_impurity_split=None,
                                                    min_samples_...
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [10, 20, 30, 40, 50, 60,
                                                      70, 80, 90, 100, 110,
                                                      None],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 4],
                                        'min_samples_split': [2, 5, 10],
                                        'n_estimators': [200, 400, 600, 800,
                                                         1000, 1200, 1400, 1600,
                                                         1800, 2000]},
                   pre_dispatch='2*n_jobs', random_state=42, refit=True,
                   return_train_score=False, scoring=None, verbose=2)
```

Figure 13: Grid Search with Cross Validation, Random Forest

#### 3.2.2.3 Results and evaluation :

First, we present a brief comparison between the results of the 3 classifiers using a number of classification scores.

| | model_name | accuracy_score | precision_score | recall_score | f1_score |
|---|---|---|---|---|---|
| 0 | Support Vector Machine | 0.967066 | 0.969562 | 0.967449 | 0.967023 |
| 1 | Random Forest | 0.916168 | 0.926937 | 0.916764 | 0.916287 |
| 2 | Decsision Tree | 0.892216 | 0.901106 | 0.892669 | 0.891424 |

Figure 14: Classification scores

Then we used the sklearn function classification-report to present, for each classifier, a report that gives an idea on the true and false predictions of each class. This report displays the following specified classification metrics:

- **Precision:** Accuracy of positive predictions.
  Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive.

- **Recall:** Fraction of positives that were correctly identified.
  Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives.

- **F1 score:** What percent of positive predictions were correct?
  The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.
  F1 Score = 2*(Recall * Precision) / (Recall + Precision)

KNN Classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.80 | 0.89 | 30 |
| 1 | 0.90 | 0.90 | 0.90 | 30 |
| 2 | 0.91 | 0.65 | 0.75 | 31 |
| 3 | 0.72 | 0.93 | 0.81 | 30 |
| 4 | 0.97 | 1.00 | 0.98 | 30 |
| 5 | 0.90 | 0.87 | 0.88 | 30 |
| 6 | 1.00 | 1.00 | 1.00 | 31 |
| 7 | 1.00 | 0.84 | 0.91 | 31 |
| 8 | 0.83 | 1.00 | 0.91 | 30 |
| 9 | 0.93 | 0.81 | 0.86 | 31 |
| 10 | 1.00 | 0.71 | 0.83 | 31 |
| 11 | 0.83 | 0.80 | 0.81 | 30 |
| 12 | 0.83 | 1.00 | 0.91 | 30 |
| 13 | 0.97 | 0.97 | 0.97 | 30 |
| 14 | 0.79 | 0.97 | 0.87 | 31 |
| 15 | 1.00 | 0.87 | 0.93 | 30 |
| 16 | 0.88 | 0.97 | 0.92 | 30 |
| 17 | 0.87 | 0.90 | 0.89 | 30 |
| 18 | 0.93 | 0.90 | 0.92 | 30 |
| 19 | 0.86 | 0.97 | 0.91 | 31 |
| 20 | 0.84 | 0.90 | 0.87 | 30 |
| 21 | 0.88 | 0.90 | 0.89 | 31 |
| accuracy | | | 0.89 | 668 |
| macro avg | 0.90 | 0.89 | 0.89 | 668 |
| weighted avg | 0.90 | 0.89 | 0.89 | 668 |

SVM Classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 1.00 | 0.98 | 30 |
| 1 | 1.00 | 1.00 | 1.00 | 30 |
| 2 | 0.96 | 0.74 | 0.84 | 31 |
| 3 | 0.78 | 0.97 | 0.87 | 30 |
| 4 | 0.97 | 1.00 | 0.98 | 30 |
| 5 | 1.00 | 1.00 | 1.00 | 30 |
| 6 | 1.00 | 1.00 | 1.00 | 31 |
| 7 | 0.97 | 0.97 | 0.97 | 31 |
| 8 | 1.00 | 1.00 | 1.00 | 30 |
| 9 | 1.00 | 0.97 | 0.98 | 31 |
| 10 | 1.00 | 0.90 | 0.95 | 31 |
| 11 | 1.00 | 1.00 | 1.00 | 30 |
| 12 | 0.94 | 1.00 | 0.97 | 30 |
| 13 | 1.00 | 1.00 | 1.00 | 30 |
| 14 | 1.00 | 1.00 | 1.00 | 31 |
| 15 | 1.00 | 0.97 | 0.98 | 30 |
| 16 | 0.91 | 0.97 | 0.94 | 30 |
| 17 | 0.91 | 1.00 | 0.95 | 30 |
| 18 | 0.97 | 0.93 | 0.95 | 30 |
| 19 | 1.00 | 0.97 | 0.98 | 31 |
| 20 | 0.97 | 0.97 | 0.97 | 30 |
| 21 | 1.00 | 0.94 | 0.97 | 31 |
| accuracy | | | 0.97 | 668 |
| macro avg | 0.97 | 0.97 | 0.97 | 668 |
| weighted avg | 0.97 | 0.97 | 0.97 | 668 |

RF Classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 0.97 | 0.97 | 30 |
| 1 | 0.76 | 0.93 | 0.84 | 30 |
| 2 | 1.00 | 0.61 | 0.76 | 31 |
| 3 | 0.71 | 1.00 | 0.83 | 30 |
| 4 | 0.94 | 1.00 | 0.97 | 30 |
| 5 | 1.00 | 0.90 | 0.95 | 30 |
| 6 | 1.00 | 1.00 | 1.00 | 31 |
| 7 | 0.96 | 0.84 | 0.90 | 31 |
| 8 | 0.86 | 1.00 | 0.92 | 30 |
| 9 | 0.93 | 0.90 | 0.92 | 31 |
| 10 | 0.96 | 0.77 | 0.86 | 31 |
| 11 | 1.00 | 0.97 | 0.98 | 30 |
| 12 | 0.82 | 0.93 | 0.87 | 30 |
| 13 | 0.94 | 0.97 | 0.95 | 30 |
| 14 | 0.97 | 1.00 | 0.98 | 31 |
| 15 | 0.96 | 0.90 | 0.93 | 30 |
| 16 | 0.97 | 0.97 | 0.97 | 30 |
| 17 | 0.93 | 0.90 | 0.92 | 30 |
| 18 | 1.00 | 0.93 | 0.97 | 30 |
| 19 | 1.00 | 0.90 | 0.95 | 31 |
| 20 | 0.91 | 0.97 | 0.94 | 30 |
| 21 | 0.91 | 0.94 | 0.92 | 31 |
| accuracy | | | 0.92 | 668 |
| macro avg | 0.93 | 0.92 | 0.92 | 668 |
| weighted avg | 0.93 | 0.92 | 0.92 | 668 |

Figure 15: Classification reports

Moreover, in sklearn the Random Forest classifier has a features-importance method that rank the features by their contributions in the classification task.

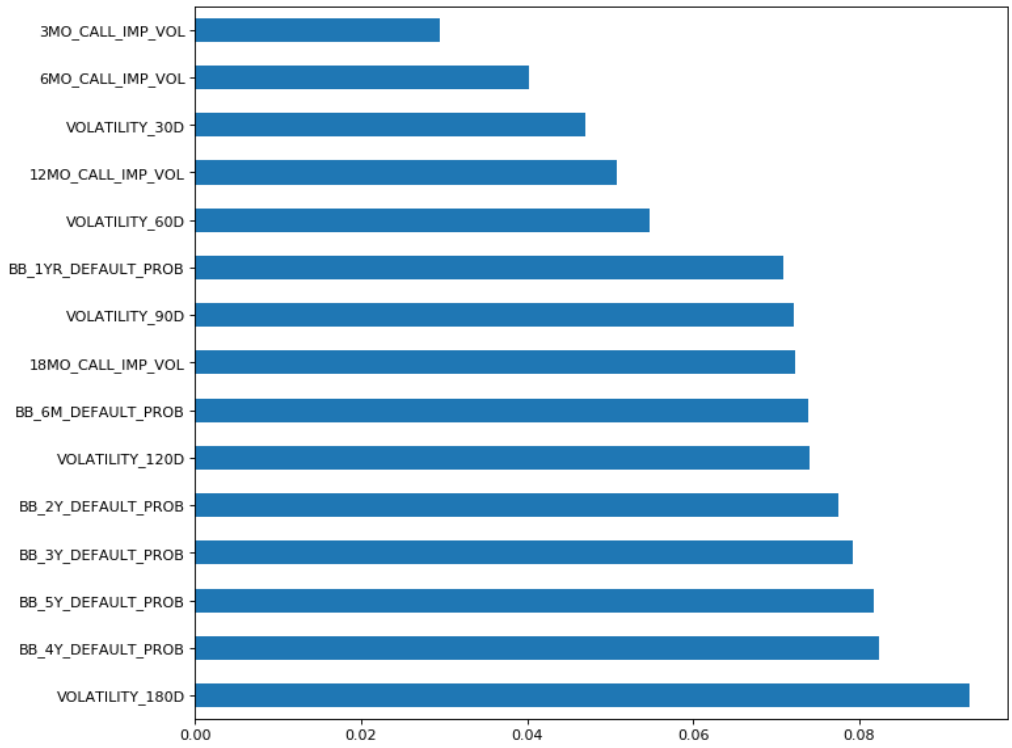


Figure 16: Random Forest features importance

As we have seen, the 3 algorithms provide very good classification scores using the observable counterparties data. However, we can't say anything about the CDS proxy method till now. In the next section, we will backtest the method and see whether it outperforms the Credit Curve Mapping Approach or not.

### 3.2.3 Backtesting

To assess the method's results, we should find a benchmark and define a metric to quantify the performances. Hence, we decided to iterate over the observable counterparties list and do the following steps:

- Consider that counterparty as non-observable and train the classifiers on the remaining 21 entities data.
- Obtain the predictions with the trained classifier and calculate a CDS proxy for this counterparty.
- Compare the CDS proxy results with the real CDS spreads.

As we have 92 samples for each counterparty, we didn't know which one to use for the prediction, so we chose to predict the corresponding class for each sample and

see if we obtain always the same class. We present here an example of classification summary:

| | A | B |
|---|---|---|
| 1 | | Classif_summary |
| 2 | AXASAPA | {'GFCPA': 85, 'SCORPA': 7} |
| 3 | BBVMA | {'BNPPA': 89, 'CNCAPA': 3} |
| 4 | ALTAEBANCO | {'DEUFT': 33, 'COMZFT': 24, 'CNCAPA': 21, 'SGPA': 7, 'DANSKCO': 7} |
| 5 | INTERMA | {'INTMI': 45, 'CRITECAIXA': 32, 'MEDIBCAMI': 8, 'SVENSK': 4, 'RZBVI': 3} |
| 6 | BNPPA | {'BBVMA': 88, 'INTMI': 3, 'INTERMA': 1} |
| 7 | CRITECAIXA | {'INTERMA': 49, 'RZBVI': 27, 'BBVMA': 15, 'INTMI': 1} |
| 8 | COMZFT | {'DEUFT': 50, 'CREDIMI': 36, 'ALTAEBANCO': 6} |
| 9 | CNCAPA | {'BBVMA': 49, 'SGPA': 39, 'BNPPA': 3, 'ALTAEBANCO': 1} |
| 10 | DANSKCO | {'DEUFT': 32, 'CNCAPA': 20, 'COMZFT': 19, 'ALTAEBANCO': 10, 'SGPA': 8, 'RZBVI': 2, 'BNPPA': 1} |
| 11 | DEUFT | {'BBVMA': 51, 'ALTAEBANCO': 29, 'COMZFT': 11, 'DANSKCO': 1} |
| 12 | GFCPA | {'AXASAPA': 42, 'SCORPA': 41, 'KLEPIERRE': 9} |
| 13 | HANRUECK | {'AXASAPA': 58, 'SCORPA': 34} |
| 14 | INTMI | {'BNPPA': 55, 'INTERMA': 31, 'MEDIBCAMI': 5, 'RZBVI': 1} |
| 15 | KLEPIERRE | {'SCORPA': 75, 'GFCPA': 9, 'MEDIBCAMI': 5, 'INTERMA': 2, 'SWEDSK': 1} |
| 16 | MEDIBCAMI | {'INTERMA': 78, 'SVENSK': 8, 'INTMI': 6} |
| 17 | RZBVI | {'CRITECAIXA': 44, 'SWEDSK': 31, 'INTERMA': 10, 'SVENSK': 6, 'INTMI': 1} |
| 18 | SCORPA | {'GFCPA': 72, 'AXASAPA': 13, 'KLEPIERRE': 7} |
| 19 | SEBSK | {'SWEDSK': 66, 'SVENSK': 17, 'MEDIBCAMI': 9} |
| 20 | SGPA | {'COMZFT': 48, 'CNCAPA': 29, 'ALTAEBANCO': 7, 'CREDIMI': 3, 'DEUFT': 3, 'DANSKCO': 2} |
| 21 | SVENSK | {'SWEDSK': 75, 'MEDIBCAMI': 6, 'RZBVI': 5, 'INTMI': 3, 'INTERMA': 3} |
| 22 | SWEDSK | {'SVENSK': 56, 'SEBSK': 31, 'RZBVI': 5} |
| 23 | CREDIMI | {'COMZFT': 82, 'ALTAEBANCO': 10} |

Figure 17: Classification summary of the SVM algorithm

To explain the above summary: using the 92 samples of "AXASAPA", the Support Vector Machine predicted "GFCPA" 85 times and "SCORPA" 7 times. Following these results, we decided to redefine the proxy in order to take into account the heterogeneity of the classification results. Therefore, the CDS proxy is now a weighted average which gives:

$$Proxy.CDS('AXASAPA') = (85 * CDS('GFCPA') + 7 * CDS('SCORPA'))/92$$

Then, following this approach we calculated the different CDS-proxies for each counterparty and using the different classifiers, and compared these results with the real CDS spreads and the CCMP proxy spreads (Credit Curve Mapping Approach).

**Reminder:** The Credit Curve Mapping Approach proxies the CDS rates of a Nonobservable from a given Region/Sector/Rating bucket by the mean of the spreads of the single-name CDS rates within that bucket.

| | 6M | 1Y | 2Y | 3Y | 4Y | 5Y | 7Y | 10Y | 15Y | 20Y | 30Y |
|---|---|---|---|---|---|---|---|---|---|---|---|
| BBVMA | 15,7804 | 19,544 | 28,033 | 35,494 | 43,0502 | 50,8817 | 65,8394 | 77,0843 | 85,1131 | 90,067 | 92,9428 |
| BBVMA_proxy | 16,473525 | 21,13078152 | 28,18456413 | 33,9877837 | 40,50779239 | 46,82200978 | 55,73899674 | 63,40305435 | 67,80495978 | 70,41292065 | 73,09936848 |
| BBVMA_CCMP_proxy | 26,32214836 | 32,06829447 | 45,63417849 | 57,46747619 | 68,76452857 | 79,7379381 | 95,02869424 | 107,0976026 | 115,9460368 | 120,6345861 | 125,8809924 |
| BBVMA_proxy_error | 0,043923158 | 0,081190213 | 0,005406633 | -0,042435801 | -0,059056813 | -0,079786843 | -0,15340971 | -0,177484204 | -0,203354598 | -0,21821621 | -0,213501546 |
| BBVMA_CCMP_proxy_error | 0,668027956 | 0,640825546 | 0,627873524 | 0,619075793 | 0,597310316 | 0,567124096 | 0,44334083 | 0,389356881 | 0,362258416 | 0,339387191 | 0,354392082 |

Figure 18: Example of SVM proxy results

where, for each maturity, a proxy error is the relative error calculated with respect to the real CDS spread.

Finally, we defined our metric, the proxy total error, as:

$$Proxy.total.error = \sum_{maturities} Proxy.error(maturity)^2$$

and we summarized the final results in the following table:

| Counterparty | CCMP_proxy_total_error | knn_Proxy_total_error | SVM_Proxy_total_error | RF_Proxy_total_error |
|---|---|---|---|---|
| AXASAPA | 14,58744879 | 1,408439155 | 19,54674462 | 14,09688042 |
| BBVMA | 3,032966529 | 0,152828411 | 0,20979427 | 0,176156558 |
| ALTAEBANCO | 1,212159604 | 3,425261222 | 1,876335642 | 2,531570202 |
| INTERMA | 1,205384879 | 0,125323519 | 0,1177877 | 0,100824128 |
| BNPPA | 4,896182374 | 0,540250663 | 0,442335117 | 1,06605612 |
| CRITECAIXA | 1,731644838 | 0,406668493 | 0,575348233 | 0,7685315 |
| COMZFT | 0,437206861 | 0,214441389 | 4,466317397 | 0,471261462 |
| CNCAPA | 15,50333155 | 6,052689729 | 1,330366694 | 32,65759692 |
| DANSKCO | 7,399289521 | 19,68328467 | 6,639086339 | 1,598772268 |
| DEUFT | 0,686386183 | 1,344988243 | 0,897594631 | 0,741607668 |
| GFCPA | 0,709632386 | 3,472945056 | 1,202638748 | 2,176495944 |
| HANRUECK | 32,02608045 | 2,75271499 | 2,465534353 | 2,375535153 |
| INTMI | 1,091036596 | 0,410256535 | 1,206934084 | 0,470648081 |
| KLEPIERRE | 4,238863167 | 4,603341528 | 4,811689404 | 2,222049647 |
| MEDIBCAMI | 1,409663999 | 0,142943198 | 0,14485307 | 0,317558055 |
| RZBVI | 0,534202515 | 0,922692877 | 0,147054614 | 0,176868958 |
| SCORPA | 2,341406904 | 20,8721839 | 4,232113522 | 13,75324752 |
| SEBSK | 12,79160609 | 0,297794181 | 0,206590304 | 0,045976837 |
| SGPA | 6,867269499 | 3,906554376 | 1,946927919 | 8,364151227 |
| SVENSK | 48,88275011 | 4,406382154 | 10,12837619 | 8,721655864 |
| SWEDSK | 16,01173 | 0,282783717 | 0,039021087 | 0,11970927 |
| CREDIMI | 1,299571503 | 1,206157731 | 1,640142524 | 0,739524106 |
| Total average error | 8,131627925 | 3,483223897 | 2,921526657 | 4,258758087 |
| | proxy_error < CCMP_proxy error | | proxy_error > CCMP_proxy error | |

Figure 19: Proxies final metrics

The results seem promising, however, we can't affirm that the classification proxy is a better proxy because in reality the CCMP proxy is computed with all the available CDS spreads but here we only used the CDS spreads of the entities mentioned above and that to maintain the validity of the comparison. As a future direction, the aim is to search for more data or even try to replace for example the implied volatilities with a proxy that may approximate the latters.

# 4 Neural network for CVA: Learning Future Values

## 4.1 Context and model description

This model will be used to produce Bermudan swaptions price distribution at some pre-specified exposure dates. These future Mark-to-Market distributions will then feed the XVA engine and produce the relevant XVA metrics. While the intention is to use it mainly for Bermudan swaptions, the methodology is applicable to most of Interest Rates products.

The pricing of Bermudan options amounts to solving a dynamic programming principle, in which the main difficulty, especially in high dimension, comes from the conditional expectation involved in the computation of the continuation value. These conditional expectations are classically computed by regression techniques on a finite dimensional vector space.

The proposed approach, once validated, will leverage on the existing FO pricing system and use the same short rate diffusion (1 Factor Hull-White model). However, for the moment, we will use a simpler short rate model which is Vasicek model to test the approach.
At each exposure date, for each path of the Monte-Carlo simulation, we have access to the simulated short rate and a "Backward Monte-Carlo" technique is used to compute the discounted future cash-flows on that path and then estimate the price at each exposure date for each short rate value.
As Bermudan swaptions are callable instruments, the calculation of future prices distribution requires additional care to handle the fact that potentially the product has been called/exercised before the exposure date we are looking at.

## 4.2 Model methodology

In this section, we first introduce the main notations and give a quick overview of the different payoffs looked at and the corresponding valuation function. Then, we briefly introduce the Vasicek model used for the short rate. Finally, we explain the

Future-Mark-to-Market distribution calculation as well as the neural networks used for the computation of the continuation values.

### 4.2.1 Standard Bermudan Swaptions

A Standard Bermudan Swaption is an option for the holder to enter into a fixed-floating swap on any of its fixing dates. The Swaption is similar to a Bermuda option, which offers the same predetermined schedule of potential exercise dates where the exercise values in this case are the vanilla swap values. We will recall then the Vanilla Swap value.
Let's consider the following notations:

- $(\mathrm{T_n})_{0\leq \mathrm{n}\leq \mathrm{N}}$ corresponds to the swap dates or underlying swap in the case of swaptions and $\tau_n = T_{n+1} - T_n$. For clarity's sake, it is assumed in the payoff description that float and fixed legs have the same schedule.
- $\mathrm{L}\left(\mathrm{t}, \mathrm{T_n}, \mathrm{T_{n+1}}\right)$ is the index forward rate defined by:

$$\mathrm{L}\left(\mathrm{t}, \mathrm{T_n}, \mathrm{T_{n+1}}\right) = \frac{1}{\tau_{\mathrm{n}}}\left(\frac{1}{\mathrm{P}^f\left(\mathrm{t}, \mathrm{T_n}, \mathrm{T_{n+1}}\right)} - 1\right)$$

For a fixed rate payer swap, the payoff at time $T_{n+1}$ is given by:

$$\tau_n \left(L\left(T_n, T_n, T_{n+1}\right) - k\right)$$

Hence, the value of a fixed-floating swap is given by:

$$V_{\mathrm{Swap}}\left(t\right) = \sum_{n=0}^{N-1} \tau_n P\left(t, T_{n+1}\right)\left(L\left(t, T_n, T_{n+1}\right) - k\right)$$

### 4.2.2 Description of the proposed short rate model: Vasicek model

The model specifies that the instantaneous interest rate follows the stochastic differential equation:

$$dr_t = a\left(b - r_t\right)dt + \sigma dW_t$$

where:

- $W_t$ is a standard Brownian motion under the risk neutral probability of the forecast economy.
- $b$ : "long term mean level". All future trajectories of $r$ will evolve around a mean level b in the long run.
- $a$ : "speed of reversion". $a$ characterizes the velocity at which such trajectories will regroup around $b$ in time.
- $\sigma$ : "instantaneous volatility", measures instant by instant the amplitude of randomness entering the system.

### 4.2.3 Future-Mark-to-Market distribution calculation

We consider a Bermudan swaption and denote :

- $(t_i)_{0\leq i\leq n}$ the CVA dates i.e. dates at which we need the exposure
- $(T_j)_{0\leq j\leq m}$ the exercise dates
- $(\tau_k)_{0\leq k\leq l}$ the coupon payment dates
- $(X(\tau_k))_{0\leq k\leq l}$ the coupon paid on the coupon payment dates
- $I\left(T_j\right) = \mathbf{1}_{\tau>T_j}$ the exercise indicator where $\tau$ is the exercise date. It is one if the option has been exercised (i.e. enter into the underlying swap) before or at $T_j$, and zero otherwise.
- $M(t)$ is the conditional expectation of a Bermudan swaption to the filtration t and $I(Tj) = 0$ where $T_j \leq t \leq T_{j+1}$

$$M(t) = \mathbb{E}_t\left[\sum_{\tau_k>t} D\left(t,\tau_k\right) X\left(\tau_k\right) \mid I\left(T_j\right)\right]$$

  Given that we generally stay within the framework of Markovian processes, the dependence on time before $t$ is captured by the state variable(s) at time $t$, meaning that $M$ is actually a function of the state variable(s)

- We recall the definition of the following exercise functions

$$C_j = 1_{S(t_i)>M(t_i)}$$

$$I(t) = I_j$$

$$I_j = 1 - \prod_{i=1}^{j}\left(1 - C_i\right)$$

- $V(t)$ is the resulting mark-to-market of a bermudan swaption at time $t$ i.e. we have $V(t) = (1 - I(Tj))M(t) + I(Tj)S(t)$ for $T_j \leq t \leq T_{j+1}$ Where $S(t)$ is the price of the underlying swap at t.

The challenging part is then to efficiently compute this conditional expectation. The proposed approach to approximate the latter is similar to the Longsta-Scwartz technique but with neural networks.
The Longstaff-Schwartz algorithm is based on what is called the dynamic programming principle. The idea behind this method is that a decision at a given point in time will depend on two factors only : the immediate situation, and a valuation which accounts for all subsequent optimal solutions. The algorithm will then

run backward, as the decision at the expiration time becomes a standard optimisation problem. At the heart, the algorithm is based on a Monte-Carlo method, a temporal discretization (effectively solving a Bermudan swaption) and polynomial least-square regression.
The algorithm is computed following these steps:

- Generate N short rate trajectories following the dynamic described in the Vasicek model
- Compute the underlying swap prices $S(t)$ for each path
- At maturity, $M(t)$ is known since it is equal to the payoff function $h() = max(0, S(t))$. At any exercise date $T_j$, $M(T_j)$ is determined by the value of the short rate $r_{T_j}$ as follows:

$$M(T_j, r_{T_j}) = max(h(r_{T_j}), CV(T_j, r_{T_j}))$$

  where $CV$ denotes the continuation value of the option:

$$CV(T_j, r_{T_j}) = \mathbb{E}_{T_j, r_{T_j}} \left[ \exp(-\int_{T_j}^{T_{j+1}} r_t dt) M(T_{j+1}, r_{T_{j+1}}) \right]$$

  $CV(T_j, r_{T_j})$ can be approximated using a neural network $NN_{T_j}(r_{T_j})$.
  To obtain $NN_{T_j}(r_{T_j})$ we train our model using samples $(x^i, y^i)$ where $x^i = r^i_{T_j}$ and $y^i = (\exp(-\int_{T_j}^{T_{j+1}} r_t dt) M(T_{j+1}, r_{T_{j+1}})^i$ with $i$ is the path index.

- At any exposure date $t_i$ where $T_j \leq t_i \leq T_{j+1}$,

$$M(t_i, r_{t_i}) = \mathbb{E}_{t_i, r_{t_i}} \left[ \exp(-\int_{t_i}^{T_{j+1}} r_t dt) M(T_{j+1}, r_{T_{j+1}}) \right] = NN_{t_i}(r_{t_i})$$

### 4.2.4 Neural Networks

An artificial neural network is a system whose design was originally schematically inspired by the functioning of biological neurons, and which subsequently approached statistical methods. Neural Networks represent a learning process by a network of stylised (mathematical models of) single neurons, organised into an Input Layer, and Output Layer and one or more intermediate Hidden Layers. Each single "neuron" transforms an input vector $\mathbf{z} = (z_1, ..., z_p)$ into a single output u by first taking a linear combination $\sum w_i z_i$ of the inputs, adding a constant or bias term $w_0$, and finally applying a non-linear transformation f, called also an *activation function*, to the result:

$$u = f\left(\sum w_i z_i + w_0\right) = f\left(\mathbf{w}^T \mathbf{z} + w_0\right)$$

where the weights $w_i$ of all of the neurons will be "learned" through some global optimisation procedure.

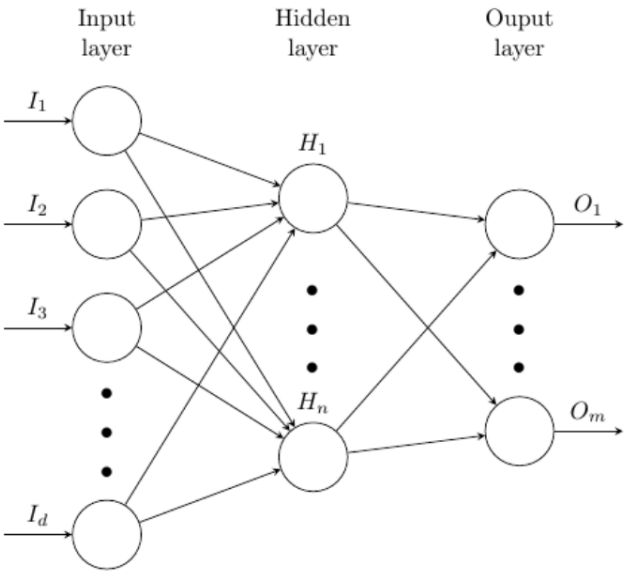


Figure 20: An Illustration for A Simple Neural Network

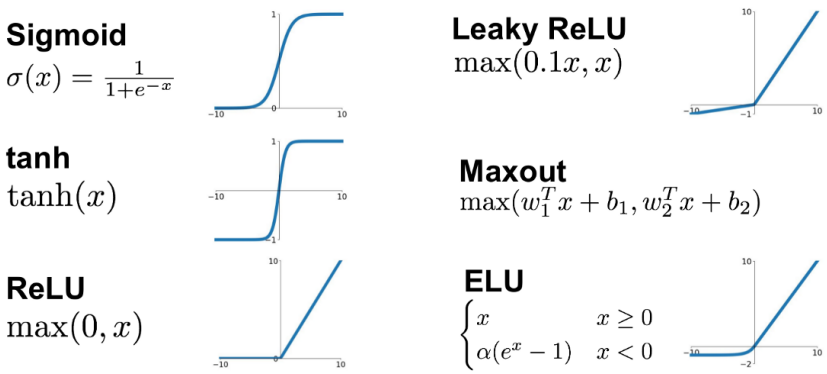


Figure 21: Activation Functions for Neural Network

The use of neural networks as function approximations is justified by the fundamental results of Hornik [1991].

**Theorem (Universal Approximation Theorem)**

Assume that the function $\sigma_a$ is non constant and bounded. Let $\mu$ denote a probability measure on $\mathbb{R}^r$, then for any $L \geq 2, \mathcal{NN}_\infty$ is dense in $L^2(\mathbb{R}^r, \mu)$

For more explanations, please refer to [4] and [5]

## 4.3 Model implementation

We first simulated the short rate paths for 10 years with a daily frequency following the dynamic:

$$dr_t = a\,(b - r_t)\,dt + \sigma dW_t$$

with the chosen parameters :

- $b = 0.03$
- $a = 0.07$
- $\sigma = 0.005$

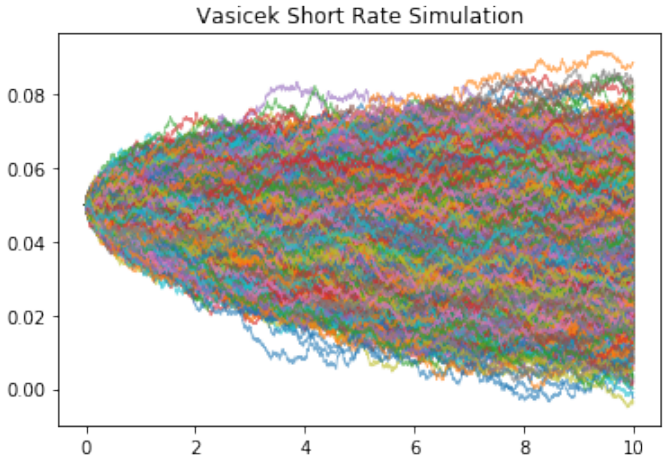


Figure 22: 5000 paths

We consider a vanilla swap on the Euribor 6m starting in 5Y and ending in 10Y ie $(\tau_k = 5 + 0.5 * k)_{1 \leq k \leq 10}$ and a Bermudan swaption on the underlying swap with the following exercise dates: $T_j = 0.25 + j * 0.5$ for $0 \leq j \leq 18$.
The exposures are computed monthly: $t_i = i * (1/12)$ for $0 \leq i \leq 116$.
We simulated the short rates with a different number of simulations and we computed at each time the swap prices for the different paths. Then, we plotted at each time the average of the swap trajectories in order to study its convergence.

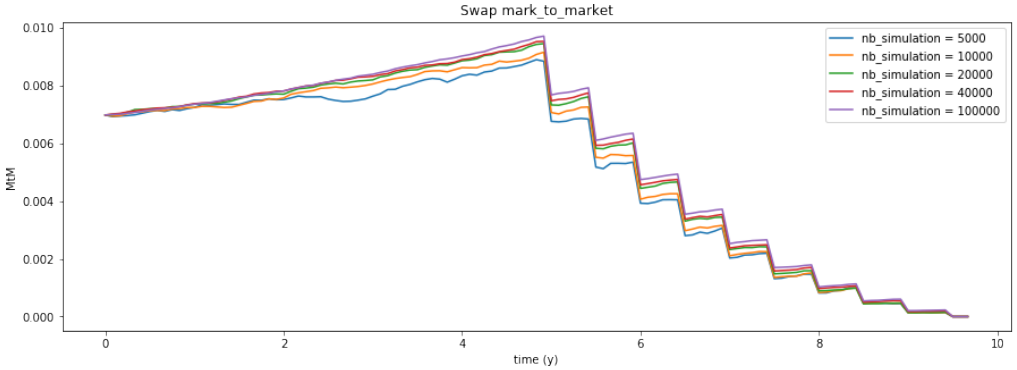


Figure 23: Swap mark-to-future convergence

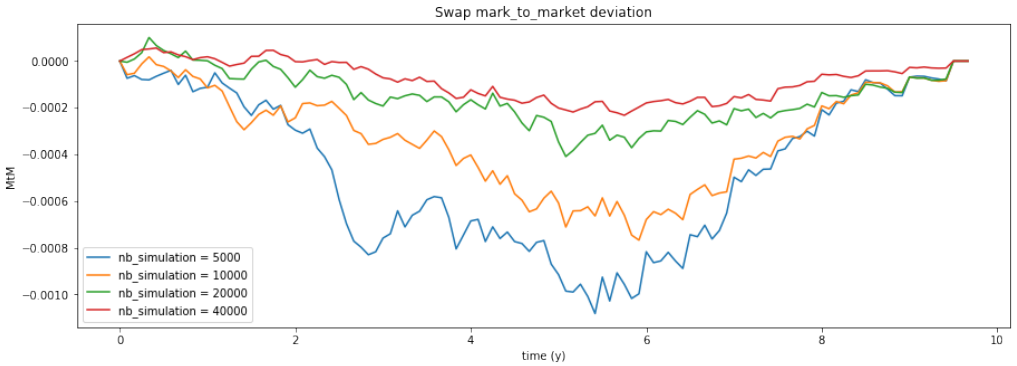


Figure 24: Swap mark-to-future deviation from the mtf with 100000 paths

Then, we proceeded to the swaption pricing using the neural networks. In the following parts, we used 5000 paths to conduct the studies for memory and computation time reasons.

After trying different parameterizations, in order to ensure stability, convergence and a minimal error , we chose to use a neural network with:

- 1 hidden layer
- 32 units per layer
- **relu** as an activation function
- **Adam** optimizer with a learning rate of 0.005

```python
initializer = initializers.RandomNormal(mean=0., stddev=1., seed=300)
def initialize_model(n_unities=32, n_layers=1, lr=0.005):
    # Define the model architecture
    model = Sequential()

    # Add an input layer
    model.add(Dense(n_unities, activation='relu', kernel_initializer='normal', input_shape=(1,)))


    # Add hidden layers
    for i in range(n_layers):
        model.add(Dense(n_unities, activation='relu', kernel_initializer='normal'))


    # Add an output layer
    model.add(Dense(1, kernel_initializer='normal'))

    # Compile model
    optimizer = Adam(lr=lr)
    model.compile(loss="mse", optimizer=optimizer)
    return model
```

Figure 25: The neural network model

In order to study the stability of the method, we computed the swaption MTF array several times and plotted at each time the average of the trajectories. In fact, the random neural network initialization may cause an instability in the swaption pricing.

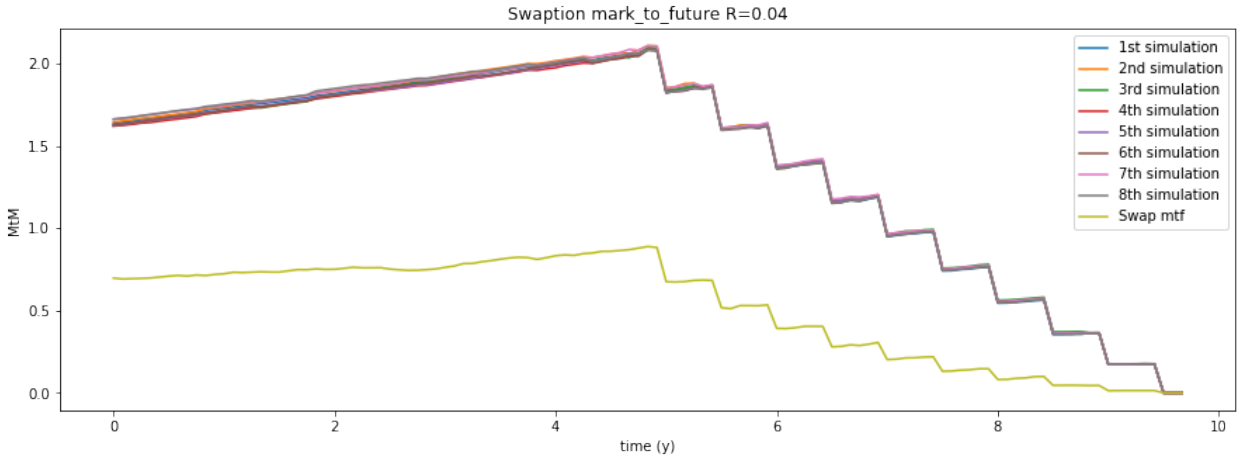


Figure 26: Bermudan Swaption pricing stability

With the chosen parameterization, we were able to ensure the stability of the method because we ensured the convergence of the optimizer. To verify this, we plotted the history of the training at each time:

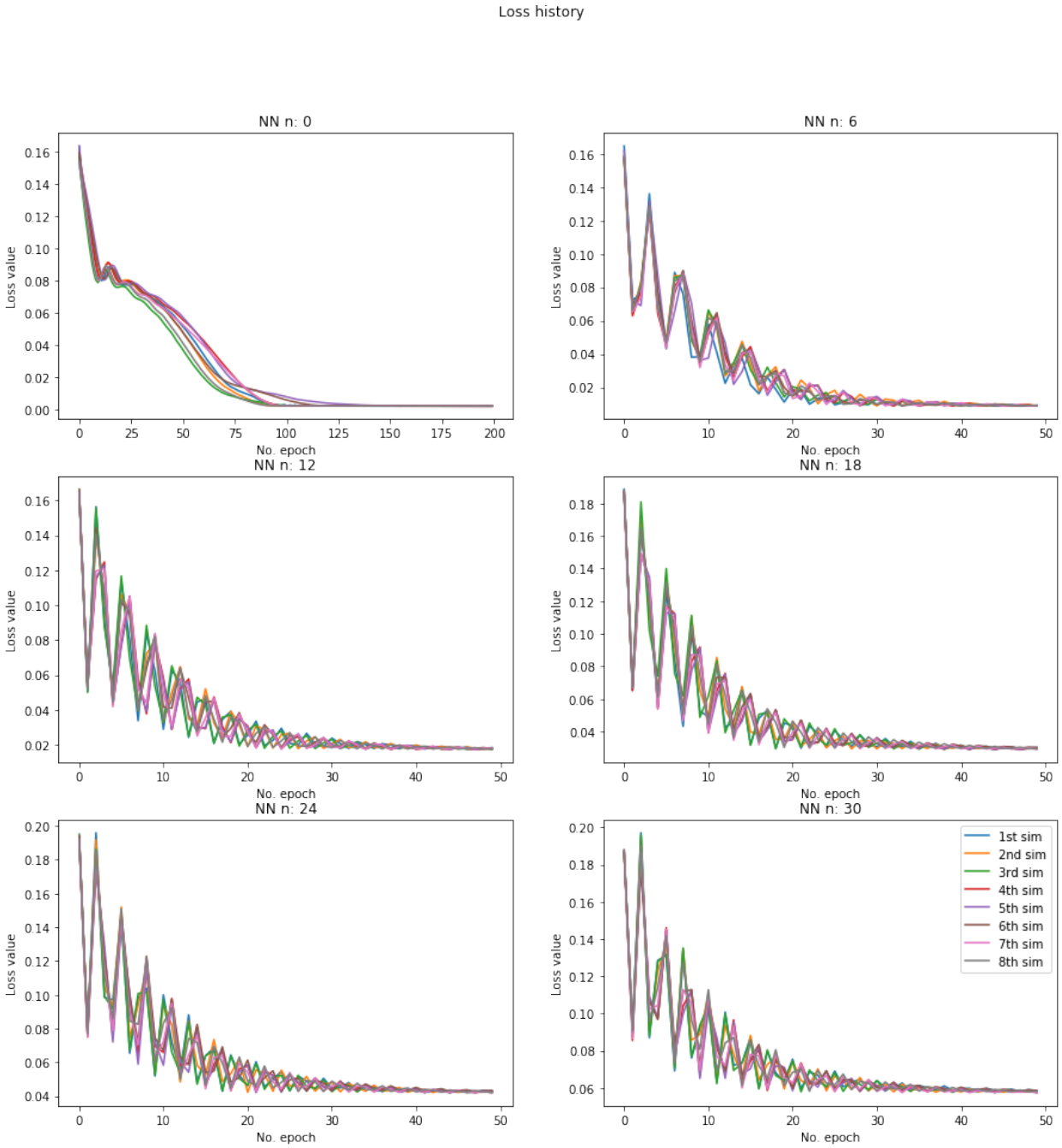


Figure 27: Neural network convergence

In order to visualize few effects, we have displayed scatter plots of the sum of discounted cash-flows for each path as a function of the state variable (i.e. $r_t$) paths with a plot of neural network regression applied to compute the continuation. Then we did the same thing with a polynomial regression and we noticed that neural

networks fit better the scatter plots. In fact, by accumulating bias, the polynomial regression has impacted the shape of the discounted cash flows distribution which looks usually like the one with neural networks.

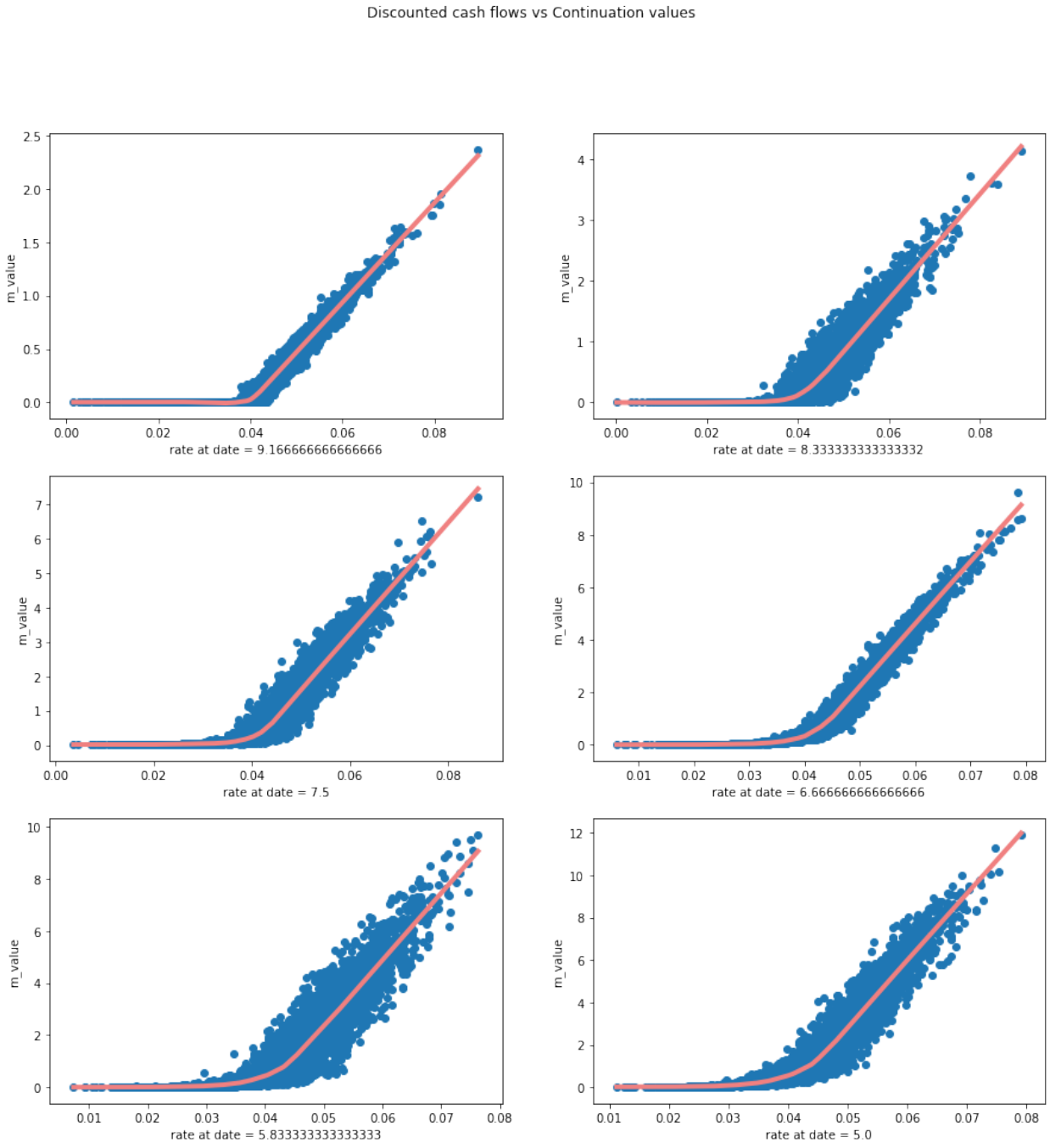


Figure 28: Neural network regression

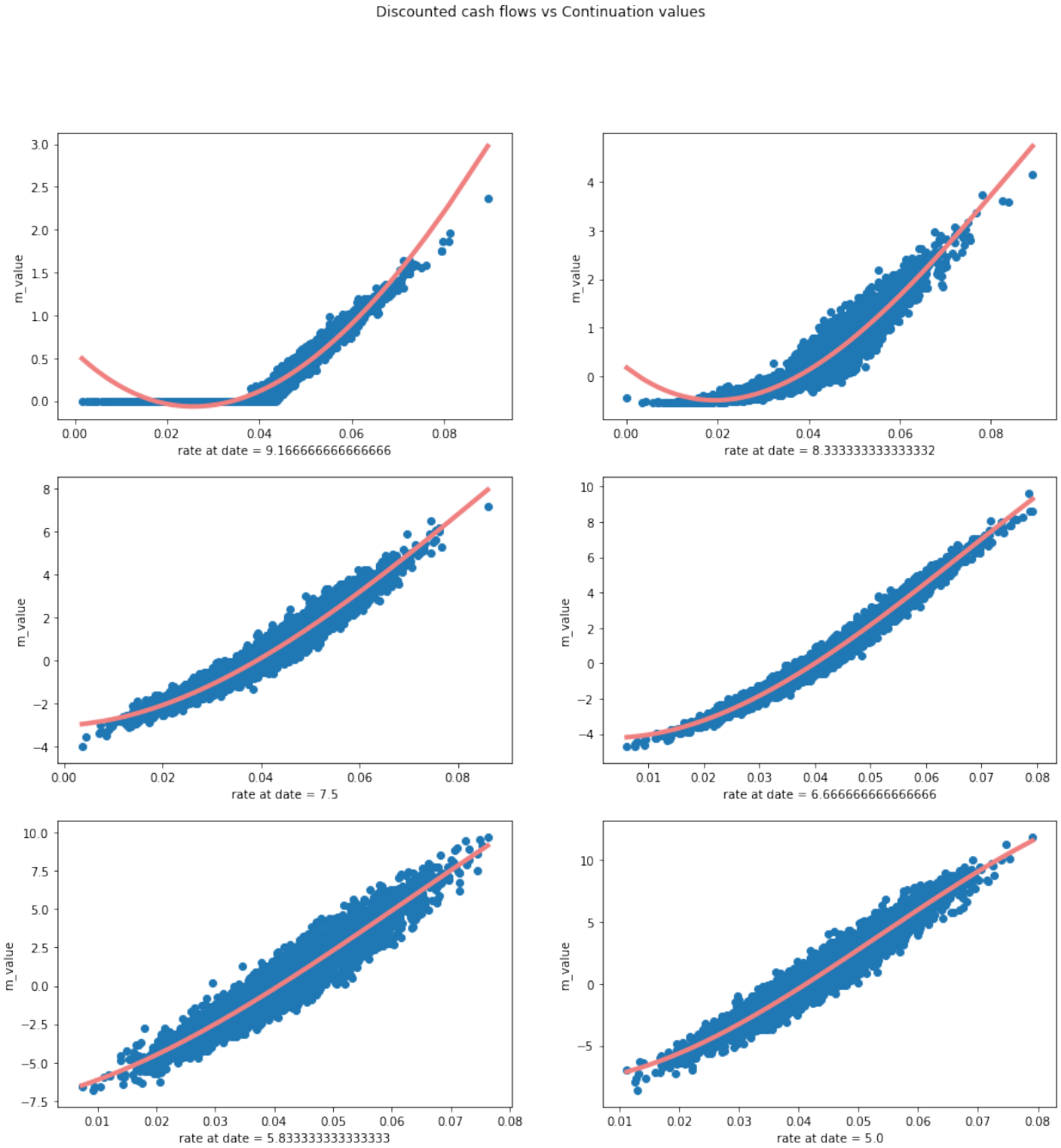


Figure 29: Polynomial regression

Finally, we plotted the future values distributions for several exposure dates:

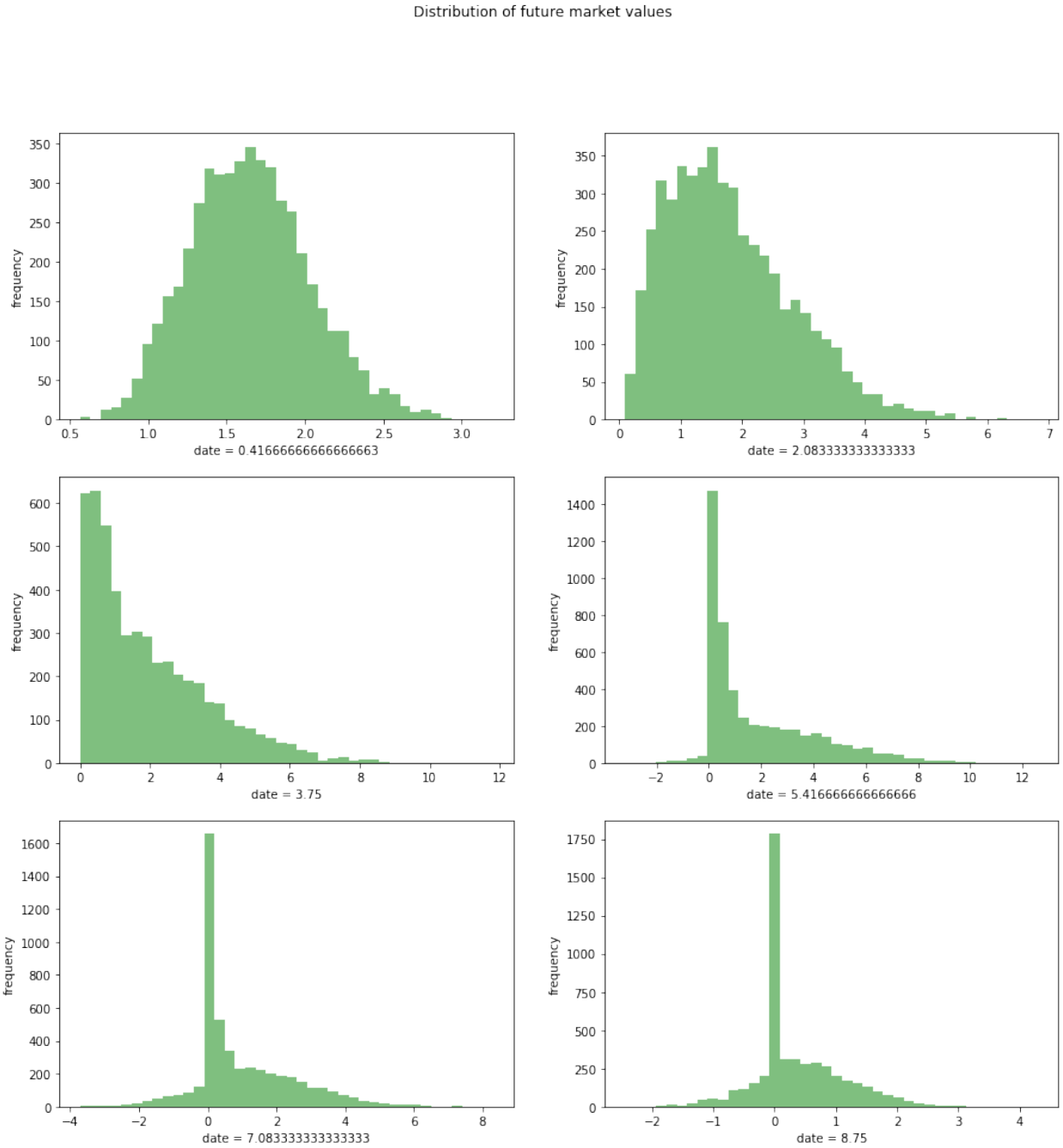


Figure 30

The negative values indicate that the swaption has been exercised earlier and the values correspond to negative swap values. We noticed also the presence of a peak at the zero level at several late dates, this peak corresponds also to negative value swaps but the swaption hasn't been exercised this time.

# 5 Conclusion

Nowadays, banks are facing a very strict regulatory environment, which will impact the way they manage their counterparty credit risk (CCR) through CVA and they ensure that they are generating sufficient return on capital. Thus, the challenge of studying the Credit Value Adjustment requires modelling the future values of a portfolio and the default probabilities of the counterparties.

For the purpose of bootstrapping the default probabilities, we created a CDS proxy that constructs the CDS rates of the non-liquid counterparties with machine learning techniques. We treated the topic as a classification problem and we trained our data using several algorithms (Random Forest, k-Nearest Neighbours and Support Vector Machine) in order to associate observable entities to non-observable ones and finally used the classification results to construct the CDS curves and compared the results with an existing method. Although we had promising results, we can't validate the method and efficiently test it because we didn't manage to collect the data for all the counterparties, hence, a future work should be done on data collection.

Later in this work, we focused on learning a Bermudan Swaption future values using neural networks to compute the continuation values. We managed to build a solid and stable model but we couldn't test it efficiently. In fact, front office prices are computed under the Hull and White 1 Factor short rate model and we used the Vasicek model to compute our prices, so we couldn't have a benchmark to do the several tests. The future perspective is to implement the Hull and White 1 Factor model simulation and then compare the prices computed with the method we implemented with front office prices.

# References


[1] Andrew Green. *XVA: Credit, Funding and Capital Valuation Adjustments*. (The wiley finance series, 2015.

[2] Zhongmin Luo Raymond Brummelhuis. Cds rate construction methods by machine learning techniques. May 22 2017.

[3] Zhaodong Zhong Charles Cao, Fan Yu. The information content of option-implied volatility for credit default swap valuation. August 2010.

[4] Jérôme Lelong Bernard Lapeyre. Neural network regression for bermudan option pricing. 2019.

[5] Dan Grecu Jian-Huang She. Neural network for cva: Learning future values. Nov 6 2018.