Iranian Journal of Numerical Analysis and Optimization
Vol. 13, No. 1, 2023, pp 19–37
https://doi.org/10.22067/ijnao.2022.73453.1073
https://ijnao.um.ac.ir/

How to cite this article
Research Article

Finding an efficient machine learning predictor for lesser liquid credit default swaps in equity markets

F. Soleymani

Abstract

To solve challenges occurred in the existence of large sets of data, recent improvements of machine learning furnish promising results. Here to propose a tool for predicting lesser liquid credit default swap (CDS) rates in the presence of CDS spreads over a large period of time, we investigate different machine learning techniques and employ several measures such as the root mean square relative error to derive the best technique, which is useful for this type of prediction in finance. It is shown that the nearest neighbor is not only efficient in terms of accuracy but also desirable with respect to the elapsed time for running and deploying on unseen data.

AMS subject classifications (2020): Primary 91G80; Secondary 62J05.

Keywords: Credit default swap (CDS); machine learning; prediction; liquidity; spread.

1 Introduction

1.1 Credit default swap

Using credit derivatives, participants of market can transfer credit risk for a portfolio of credits. The most important kind of credit derivative is the credit default swap (CDS) which consists of credit default index swap tranches, credit default index swaps, basket swaps, and swaps with single names, [6,

Received 8 November 2021; revised 22 February 2022; accepted 9 March 2022

Fazlollah Soleymani
Department of Mathematics, Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan 45137-66731, Iran. e-mails: fazlollah.soleymani@gmail.com, soleymani@iasbs.ac.ir

Chapter 1], [8]. To be more precise, CDS is a bilateral over-the-counter (OTC) derivative contract that enables two counterparties to buy and sell protection on the given reference entity. It inherits the traditional swap format and consists of two legs: a) premium leg and b) severity leg. The protection buyer pays regular fixed premium in return for receiving from the protection seller and the loss payment in case the reference entity defaults, see e.g. [22, Part 6], [26].

CDS is a basic building block for many other derivative contracts and methods. In this way, it is closely linked to debit valuation adjustment (DVA) and credit valuation adjustment (CVA). By assuming CDS contract as a continuous process/observation, it can be defined as follows [20]:

$$\text{CDS} = c\int_0^T \exp(t(-(h+r)))\ dt - h(1-R)\int_0^T \exp(-ht)\exp(-rt)\ dt, \quad (1)$$

where c, r, h, T and R are the CDS coupon, zero coupon interest rate, hazard rate, maturity of the contact and the recovery rate, respectively. The key driver of the CDS value is the hazard rate $h = -c/(R-1)$, which shows that the hazard rate is simply the CDS rate divided by the loss function.

CDS enables counterparties to manage and control credit exposure to a reference credit entity. Under the contract, on one side, the protection buyer (seller) pays (receives) a premium in return for credit loss compensation that is received (paid) when the reference entity defaults. In essence, the CDS is an insurance contract against reference entity default. Over the years, CDS has evolved in many directions and structural variations were proposed to adapt the contract to particular market needs, [4]. For example, the reference obligations can be a single entity, index, and baskets of a few names or larger pools.

To discuss about the applicability, a company proposes a strategic risk indicator, CDS spreads, for its risk dashboard. If the CDS rate, which is the price for insuring against the default of a client, went outside a specified range, then mitigation steps can be performed to deal with the client's increased risk, [12]. Recalling that CDS is basically a form of insurance that the buyer of say, a bond, buys from a financial institution, say a bank, against the bond's going "bad" (not paying in full.)

The CDS spread is the rate of payments that the buyer of the CDS makes to the seller each year. To discuss further, say the value of the bond was \$1,000. A bank might charge an annual amount \$10 per thousand if it felt that the bond had a slightly less than 1% chance of going bad in the coming year (because the \$10 would include a commission.) If the buyer paid \$10 or 1% (the credit default swap rate), the buyer would be purchasing the right to sell back the bond to the bank for \$1,000, no matter what the bond was really worth. This would be the "swap." And its purpose would be to protect against credit default. Note that in practice, CDS contracts pay in regular intervals - typically quarterly.

In addition, quanto CDSs are designated in a given currency to furnish protection when default of a certain entity, [23]. There are some instances, such as for systemically important companies or for sovereign entities, when an investor considers purchasing protection on a currency against the one, at which the reference entity' assets are denominated. It is known that in different currency denominations quanto CDS spreads are differences in CDS premiums of the same reference entity, [23].

1.2 Motivation

In this paper, predictive analysis based on machine learning (ML) [25] is discussed for some active groups of liquid CDS rates given for various daily rates, which are then employed to predict lesser liquid CDS. This is the main motivation of the paper since this application can mostly be observed in the equity or credit markets where the factor of liquidity drives specific tools into certain categories, [21], [28, Chapter 9].

1.3 ML

ML uses statistical methods to train machines from a given data set. After the learning, the systems produce optimized models that explain the data in the best way and restrict the potential biases, further enabling better assessments and decision making. Thus, such models are also broadly employed for predictions. ML is based on acquiring a habit in terms of learning. In fact, learning is considered as a process of progressive adaptation and the ability to produce the right patterns in response to a given set of inputs, [15, Chapter 6].

Here, an application of ML approach in financial mathematics is discussed, see the book [13, Part 18.8] for some background. It is shown how we can predict lesser liquid CDS spreads over a list of actively traded and liquid CDS spreads over several years of daily spreads.

ML is useful in finding patterns in contrast to traditional linear models, [16, Chapter 1]. The methods and tools like the nearest neighborhood (NeN), neural networks (NN), or decision trees (DT) suggest better flexibility in finding complex relationships. In fact, prediction as a technique to approximate outcome from supporting features basically recommends practical solutions to economics and finance where the approximation could be very invaluable. Inflation prediction, marketing campaign model testing, growth rates forecast, or market data generation, are just few instances where ML becomes a necessary tool in making decisions, [5, 27]. Additionally, reasonable CPU times and prediction ability too make ML a promising approach than the traditional regression-like methods.

1.4 Contribution

Here, we contribute by providing an ML-based model by comparing and finding several well-known ML methods when there are many features for the model which are CDS rates. As a matter of fact, we attempt in finding the best model when the number of CDS rates are 5, 8, 10 over a period of 5 to 10 years. ML is used because of the existence of large sets of data. It is discussed and illustrated that the nearest neighbor prediction method performs the best when the size of the original set of data is becoming larger and larger. In the presence of such a complex large set of financial data, it will be observed that the regression-type methods which are classical statistical tools cannot anymore be employed to tackle such problems. This paper also follows the recent works [21, 30]. In this paper, we show that ML furnishes an efficient method to solve this challenge in finance.

The advantages of this study comprise:

- Considering many CDS spreads as features for the ML techniques to do the prediction.
- By implementing and imposing several well-known ML predictions, we obtain a method for predicting CDS rates.
- Furnishing insights into the applicability and accuracy of ML-based economic models for predicting CDS rates.

1.5 Structure of the paper

After having an introductory discussion regarding the issues with CDS rates and ML in this section, the remaining sections of this article are structured as follows. In Section 2, predictive analytics (PA) is introduced, which comprises different statistical methods from ML, predictive modeling, and data mining which investigate historical and current facts to forecast the forthcoming unknown events. Section 3 furnishes how the proposed procedure can be applied on a large set of financial data. The sample size of the series is more than thousands of observations. It is shown by way of illustration that the proposed solution method for prediction is useful and provides promising results. Several numerical experiments are investigated with implementation details in Section 4 to confirm the applicability of the ML methods and to compare with several well-known and state-of-the-art methods in literature for prediction. Lastly, several concluding summaries are made in Section 5.

2 Predictive analytics

PA is a term mostly employed in analytical and statistical methods, [25], which forecasts the future by investigating the historical and current data. The forthcoming occurrences and behavior of variables can be predicted by the PA's models and a score is furnished. A lower score shows a lower likelihood of occurrence of the event and a higher score shows a higher likelihood of occurrence of an event. Transactional and historical data patterns are evaluated by such techniques for finding out the solution to many scientific problems. These models are useful in recognizing the opportunities and risks for each manager, employee, or customer, [11, 19].

Statistically speaking, the problem of prediction simplifies in finding conditional distribution of a variable y considering other variables $x = (x_1, x_2, \ldots, x_n)$. Furthermore in the methodology of data science, variables x are named as features. For the calibrated conditional distribution, generally the prediction point y is the highest value (mean), [14].

It is well known that the most common tool is (linear) regression analysis. ML recommends a better set of tools that could summarize usefully different types of nonlinear relations in the economic data. A promising predictor includes deriving a function that minimizes an error function. Then, the target of prediction methods is to obtain promising out-of-sample approximations for unseen data. This process is not trivial and generally regressions are known to be weak around out-of-sample predictions. This leads to overfitting issues (especially for regression-like methods of higher orders), [7, Chapter 3].

Basically, the targets of a ML predictive modeling task are twofold: to return a high-performing predictive model for operational use and an approximate of its performance, [9]. The process basically consists of the following stages: (a) Tuning, at which various combinations of methods and their hyper-parameter values are calibrated, (b) Attaining an ultimate model trained on all existing data by the best configuration, and finally (c) Performance estimation.

3 Problem set-up

CDS indices are tradable products that permit investors to take short or long credit risk positions in certain equity markets or segments thereof. Here it is considered that the data are generally identically distributed and independent.

3.1 Data

Let us consider n number of CDSs which are served as features, see e.g. [31]. These CDS rates are fed from market but in this work we employ simulated values based on a correlation matrix as follows:

$$A = (a_{i,j})_{n+1\times n+1}\,, \quad a_{i,j} = a_{j,i}, \quad a_{i,i} = 1, \tag{2}$$

where its entries are obtained via uniform distributions subject to the *volatility* vector $V = (v_1, v_2, \ldots, v_{n+1})$, $0 < v_i < 1$. Then the simulated data are extracted from the covariance-variance matrix

$$\mathrm{CM} = (v_i v_j a_{i,j})_{n+1\times n+1}\,, \tag{3}$$

which is a symmetric positive definite (SPD) matrix. The financial data sets can be constructed in the software system Wolfram Mathematica [1, 2] as comes next:

```
SeedRandom[12];
volatility = Reverse@Sort@RandomReal[{0, 0.1}, n + 1];
crl = RandomReal[{0, 0.5}, {n + 1, n + 1}];
crl = (1/2) (crl + Transpose[crl]);
crl = crl.Transpose[crl];
Table[crl[[i, i]] = 1, {i, n + 1}];
cm = Table[volatility[[i]]*volatility[[j]]*crl[[i, j]],
   {i, 1, Length[volatility]}, {j, 1, Length[volatility]}];

initial = RandomReal[{0.5, 1.5}, n + 1];
max = Number of days in the time period;
mean1 = {0, 0, 0, 0};
mn = MultinormalDistribution[mean1, cm];
data2 = RandomVariate[mn, max];
data1 = Prepend[data2, initial];
data = Accumulate[data1];
```

We have chosen `SeedRandom`[12] on purpose to let readers reproduce the financial data set. The data follow a multivariate normal (Gaussian) distribution with mean vector 0 and variance matrix (3). We now divide the data into several different scenarios:

- $n = 5, 8, 10$ spreads.
- $T = 5, 10$ years which roughly indicates 1825 and 3650 days, respectively.

Financial set of data with higher dimensions and more features is a recent problem. Usually, many sets of data have features with similar information.

This can act as noise in the system and increase the complexity. Note that if the data are fat then it states more features relative to observations or tall, which states many observations relative to features. Figure 1 reveals a sample set of data for $n = 5, 8, 10$ spreads when $T = 5$.

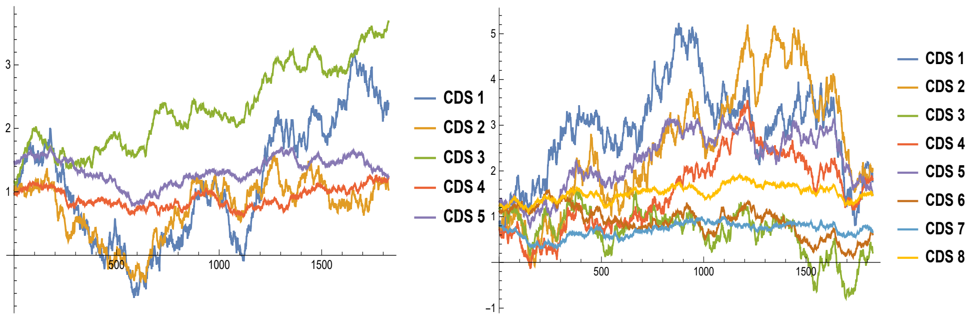


Figure 1: Five years CDS spreads for $n = 5, 8, 10$, in top, middle, and bottom, respectively.

3.2 The sub-sets

The aim of this subsection is to impose different prediction routines under the ML environment to obtain a model and then employ it for out-of-sample domain, [3, 21]. Here we first break the in-sample data (obtained from the CDS markets in practice) into three famous sets known as (i) training, (ii) validation and (iii) testing sets. The training, validation and testing subsets are roughly 60%, 20% and 20%, of the whole original set, respectively.

Here the data type is in numerical vector format and *only* the first three members of the training set is now given to illustrate how the prediction routine is going to be incorporated ($n = 10$:)

```
Training set = {
  {0.839768, 1.48679, 0.710729, 1.48696, 0.713219,
    0.923109, 1.46489, 1.43977, 1.23923, 1.29405} -> 1.2654,
  {0.961012, 1.48665, 0.78337, 1.54342, 0.696918, 0.958177,
   1.51974, 1.45154, 1.25072, 1.3013} -> 1.26877,
  {0.846854, 1.4823, 0.676574, 1.56933, 0.661563, 0.944591,
   1.51674, 1.4319, 1.23782, 1.28701} -> 1.26747
   };
```

3.3 Predictors

The classification methods are needed for our purpose and we focus on prediction methods. The predictor routines used for comparisons in this work are given as follows:

- RF: Random forest is an ensemble learning method in regression and classification that incorporates deriving a multitude of decision trees (DT). To understand this further, the prediction of the forest is attained by the mean-value tree predictions or getting the most common class. Every DT is trained on a random subset of the set of training and employs only a random subset of the features.

- DT: A DT is a flow chart-like network, in which each internal node shows a test on a feature, each branch shows the test's outcome, and each leaf shows a probability density, value distribution or class distribution.

- GBT: The Gradient boosting tree is a technique of ML for classification and problems that provides a prediction model as an ensemble of trees. The training on the trees is done sequentially with the aim of compensating for the weaknesses of previous trees. Basically the Light Gradient Boosting Machine framework is used in the back end.

- LR: The linear regression forecasts the computational value y employing a linear combination of numerical features $x = \{x_1, x_2, \ldots, x_n\}$. The conditional probability $P(y|x)$ is modeled based on

  $$P(y|x) \propto \exp\left(-\frac{(y - f(\theta, x))^2}{2\sigma^2}\right), \tag{4}$$

  with $f(\theta, x) = x\theta$.

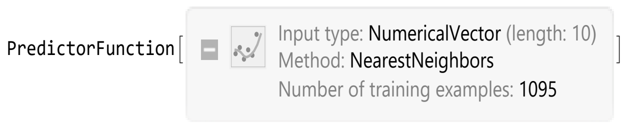
- NN: A neural network comprises stacked layers, each doing a simple calculation. Then, the information layer by layer is processed from the input layer to the output layer. A loss function is minimized to train the NN on the training set employing gradient descent.

- NeN: Nearest neighbors is a kind of instance-based learning and it chooses the averages of the values among the k nearest neighbors or the commonest class in its simplest form. To generate short term forecasts, similar patterns of behavior are located with respect to NeN by a distance measure that is normally the Euclidean distance. The time evolution of these NeNs is exploited to obtain the required forecast. Thus, the procedure employs only local information to predict and makes no effort to fit a model to the whole time series at once. The selection of the size (m) normally called the embedding dimension and of the number of neighbors (k) is a fundamental point of this routine, [10]. Note that there is no training step during NeN.

- GP: The Gaussian method is via the assumption of a Gaussian process for the model. This process is expressed by its covariance function and it will estimate the parameters of this covariance function in the training phase. Then, it is conditioned on the training data and employed to infer the value of a new instance by a Bayesian inference.

Having a large set of data, we assign 20% of the data in each scenario for the validation set. Typically this is employed when the data in the training set and the data that we want to forecast arise from various resources. Using this, the hyper-parameter selections are done by testing performance on data.

In addition, the predictors are set and trained on the target device CPU, while we use 1234 as random seeding whenever required inside the predictors. Our prediction routine can be obtained as a predictor function. As an instance for the case of NeN when $n = 10$, $T = 5$, it can be written in what follows:



```
PredictorFunction[ Input type: NumericalVector (length: 10)
                   Method: NearestNeighbors
                   Number of training examples: 1095 ]
```

To illustrate the results of comparisons, we furnish Table 1 *for one run* in the case $n = 5$, $T = 5$, which shows that number of training examples, validation set examples, and test set instances, would be 1095, 365, and 366, respectively. Here since we employ the built-in functions for the predictor routines in Mathematica, so all the hyper-parameters have been assigned automatically based on the built-in optimization techniques to choose the best hyper-parameters for the model corresponding to the training and validity sets.

Predictors are compared with respect to their CPU times (seconds) in Table 2. The time reported is based on the running CPU times for constructing the models. A critical bottleneck in using the NN is its high computational time for constructing the model. This restricts its applicability for the purpose of our financial application in predicting lesser liquid CDS rates. In other words, the larger the financial data set (of this type), the larger the CPU time is. Therefore, it becomes requisite to improve and rely on alternate predictors.

Meanwhile, the testing stage requires the comparison of the test vector to all existing data points in the data set which might take a significant amount of time.

4 Benchmarking

The implementations in this paper were performed by Mathematica 12.0 [17, Chapter 7] installed in a computer having Core i7-9750H with SSD memory and 16 GB RAM. It is important to mention that the general results and conclusion are obtained by shuffling the data set for 50 times and getting means whenever required.

4.1 Implementation details

We can apply several predictive routines defined in Subsection 3.3 to attain the forthcoming value in the out-of-sample domain (for unseen data). In ML, the validity set is used to tune our model parameter settings and a test set to evaluate the model's performance on unseen events. The procedure of prediction here using our ML methods on the list of large data sets is incorporating a prediction function on the unseen data.

Table 1: Information of compared predictors for $n = 5$, and $T = 5$.

	Sub-method, parameters	Single evaluation time (ms/ example)	Batch evaluation speed (example/ ms)	Loss	Model memory (kB)	Running time (s)
RF	Feature fraction=1/3, Leaf size = 4, Tree number = 100	8.4	12.9	-1.83 ± 0.01	605	1.3
DT	Feature fraction=1, Distribution smoothing = 1	1.14	487.	-1.43 ± 0.08	121	0.6
GBT	Leaves number=60, Learning rate = 0.2, Leaf size = 7	4.04	44.0	$+2.63 \pm 0.26$	723	10.93
LR	L1 regularization=0, L2 regularization = 100.0, Optimization method = Normal equation	1.31	362.	-2.61 ± 0.03	307	3.72
NN	Network depth=2, Max training rounds = 30	2.98	25.1	-2.95 ± 0.04	471	38.1
NeN	Neighbors number=2, Distribution smoothing = 0.5, Nearest method = k-D tree	1.11	194.	$+0.5 \pm 0.49$	174	0.6
GP	Estimation method =Maximum posterior, Search method = Simulated Annealing	3.83	9.7	-0.98 ± 0.26	6.3	5.6

Table 2: Comparisons of running times for various routines to construct the model.

n	T	RF	DT	GBT	LR	NN	NeN	GP
5	5	1.3	0.6	10.93	3.72	38.1	0.6	5.6
8	5	0.8	0.6	10.8	3.7	34.6	0.6	5.2
10	5	1.2	0.69	21.5	3.79	86.	0.7	5.9
5	10	1.4	1.29	13.0	4.42	85.	1.32	31.1
8	10	1.81	1.30	13.0	4.3	158.	1.40	33.8
10	10	1.87	1.31	17.2	4.06	419	1.41	31.3

To compare various routines for prediction, two different measures are employed as described below. The absolute error is calculated via

$$\varepsilon = \|p_{\text{predict}} - p_{\text{actual}}\|_\infty, \tag{5}$$

where p_{actual} and p_{predict} are the exact and predicted values, respectively. Also, the root mean square relative error (RMSRE) of $\mathcal{N}$ predicted values p_{predict} whose real values are p_{actual}, is defined by

$$\epsilon = \sqrt{\sum_{i=1}^{\mathcal{N}} \frac{1}{\mathcal{N}} \left| \frac{p^i_{\text{predict}} - p^i_{\text{actual}}}{p^i_{\text{actual}}} \right|^2}. \tag{6}$$

Here $\mathcal{N}$ is the length of each sub-sets. For this analysis, we compared the ε and ϵ for each training, validity and test sets in each scenario. The routines have not seen the test sets before and the accuracies that come from the incorporation of the test sets are important to find the most useful algorithm for prediction. The results based on a *mean of over 50 shuffles* on the original set containing the training, validity, and test (prediction) sets, each time, are gathered in Table 3. It is important to state that for some routines such as NeN, there is no training part and its model representation is the entire training dataset, [24, Chapter 7]. The training set in Table 3 for such methods is in fact the calibration set.

Remark 1. Here it is noted that RF is an ensemble DT model based on different feature selections and data set partitions that do not require cross-validation. However, it is used in a similar fashion just like the other models. This is mainly to check the robustness of the models when the input data change. The more general the model, the less susceptible it would be to data variation. And every model needs to be cross-validated and because of this as well as having fair comparisons, we compute mean over 50 shuffles on the original set.

Also note that KNN makes predictions using the training dataset directly.

Table 3: Comparison of mean results among different methods.

		ε			ϵ		
Scenario	Routine	Training set	Validity set	Test set	Training set	Validity set	Test set
$n = 5,\ T = 5$	RF	5.5×10^{-2}	4.4×10^{-2}	4.6×10^{-2}	9.3×10^{-3}	9.7×10^{-3}	9.8×10^{-3}
	DT	6.6×10^{-2}	6.5×10^{-2}	6.5×10^{-2}	1.0×10^{-2}	1.1×10^{-2}	1.1×10^{-2}
	GBT	2.9×10^{-2}	3.9×10^{-2}	3.9×10^{-2}	3.2×10^{-3}	5.6×10^{-3}	5.5×10^{-3}
	LR	5.2×10^{-2}	5.0×10^{-2}	5.0×10^{-2}	1.3×10^{-2}	1.3×10^{-2}	1.3×10^{-2}
	NN	2.6×10^{-2}	2.4×10^{-2}	2.6×10^{-2}	4.0×10^{-3}	4.8×10^{-3}	4.9×10^{-3}
	NeN	3.1×10^{-2}	3.3×10^{-2}	3.4×10^{-2}	2.6×10^{-3}	4.2×10^{-3}	4.3×10^{-3}
	GP	2.4×10^{-2}	2.7×10^{-2}	2.6×10^{-2}	3.3×10^{-3}	4.7×10^{-3}	4.7×10^{-3}
$n = 8,\ T = 5$	RF	2.6×10^{-2}	2.6×10^{-2}	2.6×10^{-2}	7.1×10^{-3}	7.2×10^{-3}	7.2×10^{-3}
	DT	5.4×10^{-2}	5.1×10^{-2}	5.0×10^{-2}	8.1×10^{-3}	8.9×10^{-3}	8.9×10^{-3}
	GBT	1.9×10^{-2}	2.6×10^{-2}	2.6×10^{-2}	2.1×10^{-3}	4.1×10^{-3}	4.1×10^{-3}
	LR	4.4×10^{-2}	4.3×10^{-2}	4.3×10^{-2}	1.1×10^{-2}	1.1×10^{-2}	1.1×10^{-2}
	NN	1.7×10^{-2}	1.6×10^{-2}	1.7×10^{-2}	2.7×10^{-3}	3.4×10^{-3}	3.4×10^{-3}
	NeN	1.6×10^{-2}	1.8×10^{-2}	1.8×10^{-2}	2.0×10^{-3}	3.1×10^{-3}	3.1×10^{-3}
	GP	1.7×10^{-2}	1.7×10^{-2}	1.7×10^{-2}	1.6×10^{-3}	3.3×10^{-3}	3.3×10^{-3}
$n = 10,\ T = 5$	RF	1.7×10^{-2}	1.7×10^{-2}	1.6×10^{-2}	6.0×10^{-3}	5.8×10^{-3}	5.8×10^{-3}
	DT	3.6×10^{-2}	3.2×10^{-2}	3.3×10^{-2}	6.2×10^{-3}	6.7×10^{-3}	6.7×10^{-3}
	GBT	1.2×10^{-2}	1.4×10^{-2}	1.3×10^{-2}	1.1×10^{-3}	2.1×10^{-3}	2.2×10^{-3}
	LR	1.8×10^{-2}	1.7×10^{-2}	1.7×10^{-2}	4.6×10^{-3}	4.7×10^{-3}	4.7×10^{-3}
	NN	7.0×10^{-3}	7.1×10^{-3}	7.3×10^{-3}	1.1×10^{-3}	1.5×10^{-3}	1.5×10^{-3}
	NeN	1.0×10^{-2}	1.0×10^{-2}	1.1×10^{-2}	1.4×10^{-3}	2.1×10^{-3}	2.1×10^{-3}
	GP	1.7×10^{-2}	1.3×10^{-2}	1.5×10^{-2}	2.0×10^{-3}	2.3×10^{-3}	2.4×10^{-3}
$n = 5,\ T = 10$	RF	5.0×10^{-2}	4.8×10^{-2}	5.0×10^{-2}	9.5×10^{-3}	9.7×10^{-3}	9.7×10^{-3}
	DT	7.0×10^{-2}	6.6×10^{-2}	6.5×10^{-2}	9.9×10^{-3}	1.0×10^{-2}	1.0×10^{-2}
	GBT	3.5×10^{-2}	4.1×10^{-2}	4.2×10^{-2}	4.0×10^{-3}	5.6×10^{-3}	5.6×10^{-3}
	LR	6.8×10^{-2}	6.6×10^{-2}	6.7×10^{-2}	1.5×10^{-2}	1.5×10^{-2}	1.5×10^{-2}
	NN	2.7×10^{-2}	2.7×10^{-2}	2.8×10^{-2}	4.0×10^{-3}	4.7×10^{-3}	4.7×10^{-3}
	NeN	1.9×10^{-2}	2.9×10^{-2}	2.8×10^{-2}	1.8×10^{-3}	3.8×10^{-3}	3.4×10^{-3}
	GP	2.7×10^{-2}	2.8×10^{-2}	2.9×10^{-2}	3.2×10^{-3}	4.4×10^{-3}	4.4×10^{-3}
$n = 8,\ T = 10$	RF	3.7×10^{-2}	3.9×10^{-2}	3.9×10^{-2}	1.2×10^{-2}	1.2×10^{-2}	1.2×10^{-2}
	DT	7.2×10^{-2}	7.2×10^{-2}	6.9×10^{-2}	1.1×10^{-2}	1.1×10^{-2}	1.1×10^{-2}
	GBT	2.5×10^{-2}	3.0×10^{-2}	3.0×10^{-2}	3.4×10^{-3}	4.9×10^{-3}	4.9×10^{-3}
	LR	6.5×10^{-2}	6.3×10^{-2}	6.3×10^{-2}	1.6×10^{-2}	1.6×10^{-2}	1.6×10^{-2}
	NN	1.9×10^{-2}	2.1×10^{-2}	2.0×10^{-2}	3.3×10^{-3}	4.0×10^{-3}	3.9×10^{-3}
	NeN	1.3×10^{-2}	1.9×10^{-2}	2.0×10^{-2}	1.6×10^{-3}	3.0×10^{-3}	3.0×10^{-3}
	GP	2.0×10^{-2}	2.0×10^{-2}	2.1×10^{-2}	2.5×10^{-3}	3.7×10^{-3}	3.7×10^{-3}
$n = 10,\ T = 10$	RF	3.1×10^{-2}	3.1×10^{-2}	3.2×10^{-2}	1.1×10^{-2}	1.1×10^{-2}	1.1×10^{-2}
	DT	7.7×10^{-2}	7.1×10^{-2}	6.7×10^{-2}	8.9×10^{-3}	9.6×10^{-3}	9.5×10^{-3}
	GBT	2.0×10^{-2}	2.1×10^{-2}	2.3×10^{-2}	1.7×10^{-3}	2.7×10^{-3}	2.7×10^{-3}
	LR	2.9×10^{-2}	2.8×10^{-2}	2.8×10^{-2}	8.0×10^{-3}	8.1×10^{-3}	8.0×10^{-3}
	NN	1.3×10^{-2}	1.1×10^{-2}	1.2×10^{-2}	1.7×10^{-3}	2.0×10^{-3}	2.0×10^{-3}
	NeN	8.6×10^{-3}	1.1×10^{-2}	1.1×10^{-2}	1.1×10^{-3}	2.0×10^{-3}	2.0×10^{-3}
	GP	1.1×10^{-2}	1.0×10^{-2}	1.0×10^{-2}	1.7×10^{-3}	1.9×10^{-3}	1.9×10^{-3}

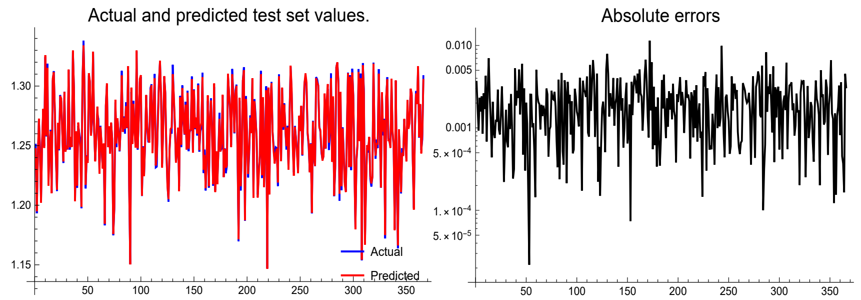


Figure 2: NeN results on the test sub-set for *one run* of the scenario $n = 10$, and $T = 5$ in left and its associated log absolute errors in right. See the online colorful version for further clarification.

4.2 Quantitative results

The large data sets considered for checking and comparing the presented models have 1825 and 3650 members each comprising 5, 8 and 10 elements respectively.

The results of comparisons for the unseen CDS rates (the test sets) are provided in Table 3. We observe that all the predictors replicated the training sets quite well. But it is observable that the best one by considering both the numerical accuracy and CPU time is mostly NeN. In order to save space and avoid providing repeated similar figures of comparisons, Figure 2 is provided only for the scenario $n = 10, T = 5$ to reveal that the NeN furnishes promising prediction on the test (prediction) sub-set.

We visualize the scatter plot of the test values as a function of the predicted values in Figure 3 for one run. This illustrates again the point that the size of the input financial data and their types have clear effect on the choice of the predictors in ML. We have illustrated the prediction by CDS data and revealed the application of non-regression tools as better techniques in PA.

It is well known that data correlation is the way at which one data set can correspond to another data set, [29]. And in ML, we can think of how the features correspond with the output. Data visualization and correlation may help decide, which ML method to use. Accordingly, Table 4 provides the means of correlations for 50 runs of the shuffled data among the actual and predicted values of different methods under several scenarios to also help in obtaining the best method for our CDS problem. Noticing that small values may not necessarily represent a bad correlation as long as the set of data has a large statistically significant correlation. When we have datasets with many features, the data correlation would be of clear importance.

Considering all the results given in Tables 1-4 reveal that NeN is the best ML routine that can be considered for prediction of lesser liquid CDS rates under the format of the financial data described in Section 3 both in terms of accuracy and the elapsed CPU time. NeN is a non-parametric non-linear forecasting routine, which is mainly via pieces of time series, in the past, that may have a resemblance to pieces in the future. This is useful in finance, see also [18]. For NeN, no learning of is needed and all of the work happens at the time a prediction is requested. Recalling that the number of neighbors selected, the length of the series and the embedding dimension to perform predictions are indicated automatically by the built-in routines inside our programming language.

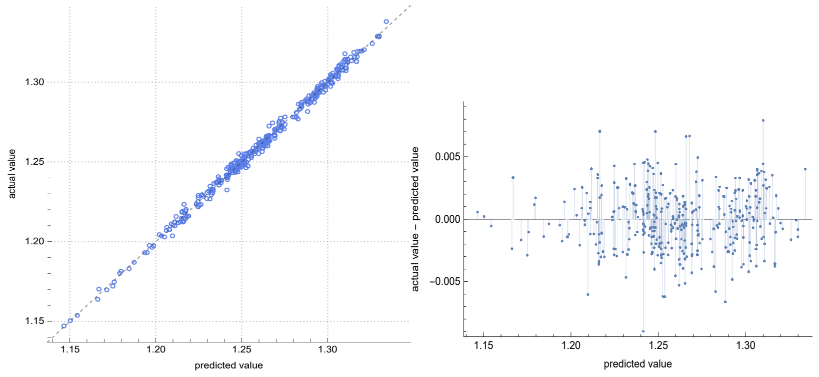


Figure 3: Comparison of perfect prediction line and predictions (left) and the residual plot (right) in NeN routine for the scenario $n = 10$, and $T = 5$.

Table 4: Correlation between the actual and predicted values.

n	T	RF	DT	GBT	LR	NN	NeN	GP
5	5	0.97	0.96	0.99	0.95	0.99	0.99	0.99
8	5	0.97	0.96	0.99	0.93	0.99	0.99	0.99
10	5	0.97	0.97	0.99	0.98	0.99	0.99	0.99
5	10	0.97	0.96	0.99	0.92	0.99	0.99	0.99
8	10	0.97	0.97	0.99	0.95	0.99	0.99	0.99
10	10	0.97	0.98	0.99	0.98	0.99	0.99	0.99

Computational pieces of evidence from Table 3 reveal that NN, NeN and GP have the best performance on the unseen test sets over 50 shuffles. However, accuracy is not the main factor in choosing the best routine when the computational CPU times are different. Clearly the lower the CPU time of training, the more useful method we have when the accuracies are almost the same. Due to this and based on Table 2, the best performance belongs to NeN in terms of computational time. Thus, we have found the NeN as our model that could be performed automatically without any theoretical assumptions through the process of progressive adaptation.

This subsection is ended by pointing out that some other models such as RNN and LSTM can also be used for comparisons that implicitly fit well with time series data. For such routines, we remind that, e.g., LSTMs take longer to train, require more memory to train, are easy to overfit, are sensitive to different random weight initializations and dropout is much harder to implement in LSTMs. Besides, our data are not stock prices and the NeN already performed best in terms of running time as well as the accuracy. Thus, comparisons to other solvers is no longer necessary.

4.3 Results on unseen data

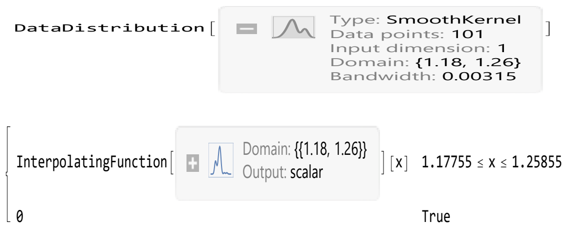
Finally in this section, it is necessary to illustrate how the NeN as an efficient ML technique for predicting lesser liquid CDS rates can be imposed on totally new unseen data. Considering the NeN routine as the predictor, saved as `prediction` previously and obtained when $n = 10$, and $T = 10$, then the data for this verification are simulated as follows:

```
n2 = 10;
SeedRandom[12345];
volatility2 = Reverse@Sort@RandomReal[{0, 0.1}, n2];
crl2 = RandomReal[{0, 0.5}, {n2, n2}];
crl2 = (1/2) (crl2 + Transpose[crl2]);
crl2 = crl2.Transpose[crl2];
Table[crl2[[i, i]] = 1, {i, n}];
cm2 = Table[volatility2[[i]]*volatility2[[j]]*crl2[[i, j]],
   {i, 1, Length[volatility2]}, {j, 1, Length[volatility2]}];
initial2 = RandomReal[{0.1, 2.0}, n2];
max2 = 100;
mean2 = ConstantArray[0, n2];
mn2 = MultinormalDistribution[mean2, cm2];
data22 = RandomVariate[mn2, max2];
data12 = Prepend[data22, initial2];
dataTest = Accumulate[data12];
dataTest2 = RandomSample[dataTest];
```

Now we impose the prediction model from NeN on unseen data:

```
pdataTest = prediction[dataTest2];
```

Here for 101 unseen data, we obtain the results of predictions based on NeN and plot them in Figure 4 (left), while the distribution for this unseen data can be obtained as follows:
with the following probability density of the predicted values:
and illustrated in Figure 4 (right).



```
DataDistribution[ Type: SmoothKernel
                  Data points: 101
                  Input dimension: 1
                  Domain: {1.18, 1.26}
                  Bandwidth: 0.00315 ]

{ InterpolatingFunction[ Domain: {{1.18, 1.26}}
                         Output: scalar ][x]   1.17755 ≤ x ≤ 1.25855
  0                                            True
```

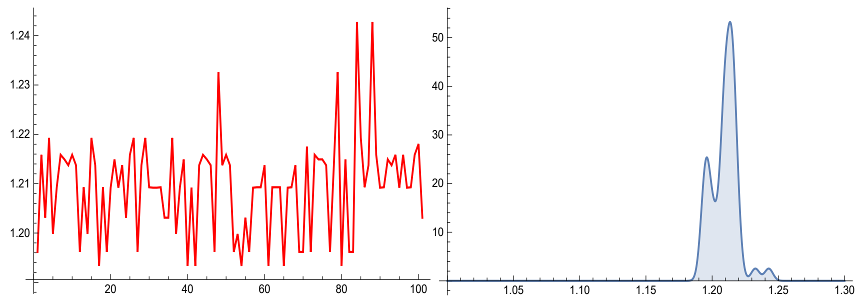


Figure 4: Predicted values (left) and their data distribution (right) in NeN routine for unseen data.

5 Conclusions and future work

Recently, there has been a proliferation of ML techniques and growing interest in their applications in finance, where they have been applied to sentiment analysis of news, trend analysis, and portfolio optimization. This paper has explored the potential of ML to enhance the investment process. A prediction model based on ML was discussed in equity markets when the numbers of predictors/features which are CDS rates are high over a larger period of time. This is especially relevant to finance where the ability of data groups to forecast the values of lesser liquid tools is of high interest. The results obtained in this work are useful as a model for predicting lesser liquids and can be employed for further investigation of the dynamic relationship between the VIX index and the CDS markets. When dealing with multidimensional set of data, it is requisite to filter out non-correlated features. Instead, it is better to use fewer highly correlated features to train a model. Taking into account such consideration may help improve the robustness of the NeN method, which is under study for further work in our team.

Acknowledgements

The author is grateful to two anonymous referees for their constructive comments and corrections on an earlier version of this work.

References

[1] R. Adhikari, *Foundations of computational finance*, The Mathematica J., 22 (2020), 1-59.

[2] R. Adhikari, *Selected financial applications*, The Mathematica J., 23 (2021), 1-33.

[3] A. Antonov, *Variable importance determination by classifiers implementation in Mathematica*, Lecture Notes, Florida, 2015.

[4] G.O. Aragon, L. Li, J. Qian, *The use of credit default swaps by bond mutual funds: Liquidity provision and counterparty risk*, J. Finan. Econ., 131 (2019), 168-185.

[5] J. Bao, S. Franco, Y.-H. He, E. Hirst, G. Musiker, Y. Xiao, *Quiver mutations, Seiberg duality, and machine learning*, Phys. Rev. D., 102 (2020), Art. ID: 086013.

[6] T.R. Bielecki, M.R. Rutkowski, *Credit Risk: Modeling, Valuation and Hedging*, Springer, New York, 2004.

[7] C.M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, NY, 2006.

[8] D. Brigo, N. Pede, A. Petrelli, *Multi currency credit default swaps*, Int. J. Theor. Appl. Finan., 22 (2019), 1950018.

[9] S. Carbo-Valverde, P. Cuadros-Solas, F. Rodríguez-Fernández, *A machine learning approach to the digitalization of bank customers: Evidence from random and causal forests*, Plos One, 15 (2020), Art. ID: e0240362.

[10] G.H. Chen, D. Shah, *Explaining the success of nearest neighbor methods in prediction*, Found. Trends Mach. Learn., 10 (2018), 337-588.

[11] K. Cortez, M. Rodríguez-García, S. Mongrut, *Exchange market liquidity prediction with the K-nearest neighbor approach: Crypto vs. fiat currencies*, Mathematics, 9 (2021), Art. ID: 56.

[12] P.P. da Silva, I. Vieira, C. Vieira, *M&A operations: Further evidence of informed trading in the CDS market*, J. Multi. Fin. Manag. 32-33 (2015), 116-130.

[13] M.L. De Prado, *Advances in Financial Machine Learning*, Wiley, New Jersey, 2018.

[14] W. E, *Machine learning and computational mathematics*, Commun. Comput. Phys., 28 (2020), 1639-1670.

[15] F.J. Fabozzi, S.M. Focardi, P.N. Kolm, *Trends in Quantitative Finance*, The Research Foundation of CFA Institute, USA, 2006.

[16] G. Gan, C. Ma, J. Wu, *Data Clustering: Theory, Algorithms, and Applications*, SIAM, Philadelphia, 2007.

[17] N.L. Georgakopoulos, *Illustrating Finance Policy with Mathematica*, Springer International Publishing, Cham, Switzerland, 2018.

[18] D. Guégan, N. Huck, *On the use of nearest neighbors in finance*, Finance, 26 (2005), 67-86.

[19] B.M. Henrique, V.A. Sobreiro, H. Kimura, *Literature review: Machine learning techniques applied to financial market prediction*, Expert Sys. Appl., 124 (2019), 226-251.

[20] I. Hlivka, *Credit default swap valuation*, Lecture Notes, London, Quant Solutions Group, (2014), 1-2.

[21] I. Hlivka, *Predictive analytics in finance: Patterns detection for outcome prediction*, Lecture Notes, London, Quant Solutions Group, (2015), 1-14.

[22] A. Itkin, A. Lipton, D. Muravey, *Generalized Integral Transforms in Mathematical Finance*, World Scientific Publishing, Toh Tuck, Singapore, 2021.

[23] A. Itkin, V. Shcherbakov, A. Veygman, *New model for pricing quanto credit default swaps*, Int. J. Theor. Appl. Fin., 22 (2019), Art. ID: 1950003.

[24] M. Kuhn, K. Johnson, *Applied Predictive Modeling*, 1st ed., Springer Science + Business Media, New York, 2013.

[25] V. Kumar, M.L. Garg, *Predictive analytics: A review of trends and techniques*, Int. J. Comput. Appl., 182 (2018), 31-37.

[26] R. Mohamadinejad, A. Neisy, J. Biazar, *ADI method of credit spread option pricing based on jump-diffusion model*, Iran. J. Numer. Anal. Optim., 11 (2021), 195-210.

[27] A. Mosavi, Y. Faghan, P. Ghamisi, P. Duan, S.F. Ardabili, E. Salwana, S.S. Band, *Comprehensive review of deep reinforcement learning methods and applications in economics*, Mathematics, 8 (2020), Art. ID. 1640.

[28] H. Ni, X. Dong, J. Zheng, G. Yu, *An Introduction to Machine Learning in Quantitative Finance*, World Scientific Publishing Europe Ltd., London, 2021.

[29] L. Sandoval Junior, *Correlation of financial markets in times of crisis*, Phys. A: Stat. Mech. Appl., 391 (2012), 187-208.

[30] J. Sirignano, A. Sadhwani, K. Giesecke, *Deep learning for mortgage risk*, J. Finan. Econometrics, 19 (2021), 313-368.

[31] Y. Son, H. Byun, J. Lee, *Nonparametric machine learning models for predicting the credit default swaps: An empirical study*, Expert Sys. Appl., 58 (2016), 210-220.

How to cite this article
Soleymani, F., Finding an efficient machine learning predictor for lesser liquid credit default swaps in equity markets. *Iran. j. numer. anal. optim.*, 2023; 13(1): 19-37. https://doi.org/10.22067/ijnao.2022.73453.1073.