Journal of Empirical Finance 64 (2021) 144–159

Contents lists available at ScienceDirect

Journal of Empirical Finance

journal homepage: www.elsevier.com/locate/jempfin

Machine learning loss given default for corporate debt

Luke M. Olson [a], Min Qi [b], Xiaofei Zhang [b], Xinlei Zhao [b,*]

[a] Office of Financial Research, USA
[b] Office of the Comptroller of the Currency, USA

ARTICLE INFO

JEL classification:
G21
G28
C14
C52

Keywords:
Loss given default
Machine learning
Bagging
Shuffled K-fold cross-validation
Unshuffled K-fold cross-validation
Sequential blocked cross-validation

ABSTRACT

We apply multiple machine learning (ML) methods to model loss given default (LGD) for corporate debt using a common dataset that is cross-sectional but collected over different time periods and shows much variation over time. We investigate the efficacy of three cross-validation (CV) schemes for hyper-parameter tuning and bootstrap aggregation (Bagging) in preventing out-of-time model performance deterioration. The three CV methods are shuffled K-fold, unshuffled K-fold and sequential blocked, which completely destroys, keeps some and completely retains the chronological order in the data, respectively. We find that it is important to keep the chronological order in the data when creating the training and testing samples, and the more the chronological order that can be retained, the more stable the out-of-time ML LGD model performance. By contrast, although bagging improves out-of-time fit in some cases, its effectiveness is rather marginal relative to that from the unshuffled K-fold and sequential blocked CV methods. Substantial uncertainty in relative out-of-time performance remains, however, thus ongoing model performance monitoring and benchmarking are still essential for sound model risk management for corporate LGD and other ML models.

1. Introduction

Modeling loss given default (LGD) for corporate debt has attracted interest in the academic literature in recent years due to increased use of LGD as a critical component of risk management operations among financial institutions, such as credit underwriting, pricing, risk management, loss forecasting, and capital calculation. LGD is a main input when pricing various financial products, such as loans, bond and credit default swaps. It is also one of the key parameters for loss forecasting, stress testing, and capital modeling. From this perspective, LGD models are closely related to both banks' lending decisions and profitability of the portfolios the models are applied to. Consequently, developing good LGD models and fine-tuning such models has become a central function of risk management operations in many financial institutions.

One major challenge for corporate debt LGD models is the unusual distribution: corporate LGDs typically demonstrate a bi-modal distribution with modes close to the boundary values of 0 and 1. Studies have shown that nonparametric machine learning (ML) methods tend to perform better than traditional parametric models of corporate debt LGD.[1] However, financial markets can have unexpected jolts, and a model that is well-trained based on a development sample and performs decently out-of-sample may still perform poorly out-of-time. Moreover, complex data-driven ML models can be especially sensitive to market disruptions or structural changes over time. Although the importance of out-of-time testing is self-evident and well recognized in the literature (e.g., Hand, 2009), most of the existing studies on ML models for corporate debt LGD are based on in-sample and out-of-sample analysis without rigorously testing model performance out-of-time.[2] The limited number of studies that report out-of-time results investigate only

* Correspondence to: Office of the Comptroller of the Currency, 400 7th St. SW, Washington DC 20219, USA.
E-mail address: Xinlei.Zhao@occ.treas.gov (X. Zhao).

[1] See, for example, Bastos (2010, 2014), Qi and Zhao (2011), Loterman et al. (2012), Hartmann-Wendels et al. (2014), Yao et al. (2015), Nazemi et al. (2017) and Hurlin et al. (2018).

[2] See, for instance, Bastos (2014), Qi and Zhao (2011), Loterman et al. (2012), Hartmann-Wendels et al. (2014), Yao et al. (2015), Nazemi et al. (2017), and Hurlin et al. (2018).

https://doi.org/10.1016/j.jempfin.2021.08.009
Received 15 February 2021; Received in revised form 5 August 2021; Accepted 29 August 2021
Available online 7 September 2021
0927-5398/Published by Elsevier B.V.

one or two types of ML methods (e.g., regression tree in Bastos, 2010, regression tree and support vector regression in Tobback et al., 2014), use data that contain a limited number of observations (e.g., Bastos, 2010, and Tobback et al., 2014) or do not exhibit a bi-modal pattern (e.g., Tobback et al., 2014). Thus, model risk associated with various types of ML models for corporate LGD is still not well understood. In particular, there is a lack of research that evaluates the effectiveness of the typical approaches such as cross-validation (CV) schemes for hyper-parameter tuning and bootstrap aggregation (Bagging), in preventing out-of-time model performance deterioration in the context of LGD modeling.

Our primary contribution to the literature is the investigation of the efficacy of alternative CV schemes in reducing out-of-time performance deterioration for ML LGD models. CV and hyper-parameter tuning are key to model building and reducing the risk of overfitting when constructing ML models. We first investigate the shuffled K-fold CV, the standard CV method in the ML literature for selecting the hyper-parameters.[3] This method randomly shuffles the development sample thus completely eliminates any chronological order in the data for training and out-of-sample testing.

To examine whether it is important to retain some information embedded in the time dimension when selecting hyper-parameters, we then investigate the unshuffled K-fold CV, in which the chronological order is retained when creating the training and out-of-sample data.[4] However, in the unshuffled K-fold CV, the left-out fold of the training sample can be the first, second, ..., Kth fold of the training sample, so this method does not completely retain the chronological order of the data. For the third CV scheme, we adopt a sequential blocked CV method that has been used when developing ML models for time-series data (e.g., Bergmeir and Benitez, 2012; Bergmeir et al., 2018) to study how important it is to strictly maintain the chronological order of the data. All three CV schemes are described in greater detail in Section 2.2.2.

As far as we are aware of, ours is the first study that applies the unshuffled CV and sequential blocked CV to a cross-sectional dataset consisting of observations from different time periods. We find that ML models built with shuffled K-fold CV provide excellent fit in the training and out-of-sample, consistent with the prior literature. In fact, results using the shuffled K-fold CV outperform those using the unshuffled K-fold CV in the training and out-of-sample for all 12 ML methods. Out-of-sample results from the unshuffled K-fold CV are particularly poor, which is not surprising, as these results are essentially out-of-time because of the way the samples for the unshuffled K-fold CV method are constructed. However, the ML models developed using the unshuffled K-fold CV tend to outperform those using the shuffled K-fold CV out-of-time in our study. In addition, the sequential blocked CV in general outperforms both the shuffled and the unshuffled CV methods out-of-time.

We next conduct additional analysis to better understand the reasons behind the above findings. We first explore the relation between the most important explanatory variables and find a dramatic change in some of these relations from in-sample and out-of-sample, to out-of-time sample for the shuffled K-fold method. Second, we compare the hyper-parameters chosen by the three different CV methods. We find that the sequential blocked CV generally yields hyper-parameters that are more parsimonious than the other two methods, while the hyper-parameters selected by the shuffled CV are the least parsimonious. Finally, we run the models on a rolling window over time and find that the sequential blocked CV performs the best in general, while the unshuffled K-fold typically outperforms the shuffled K-fold out-of-time.

We thus draw two conclusions from the above analyses. First, it is important to keep at least some degree of the chronological order of a cross-sectional dataset collected over time when constructing training and testing samples for ML methods if the data show substantial over time variation. Second, the more chronological order of the data we can retain in the CV process, the more stable the ML LGD models' out-of-time performance.

These two findings might be applicable beyond LGD modeling. Consequently, unshuffled K-fold and, especially sequential blocked CV methods deserve more exploration in the ML literature in the future when modeling data with substantial variation overtime.

Our second contribution to the literature is the examination of the efficacy of bootstrap aggregation (bagging) in ML models for corporate debt LGD. Homogeneous ensemble techniques, such as bagging (Breiman, 1996, 2000), have been proposed to potentially address the overfitting problem by reducing the variance of single model predictions and have led to the introduction of random forests (Breiman, 2001, 2004). It is uncertain whether such a method adds value to other types of ML models in predicting LGD, and we attempt to provide insight on this open question as well. Compared with the different CV methods, the benefit from bagging is very limited, primarily restricted to simplistic ML methods (such as regression tree) when shuffled K-fold CV is used.

Finally, no ML model consistently outperforms the simple Tobit model out-of-time. Therefore, although unshuffled K-fold and sequential blocked CV can be useful, there is no guarantee that they will always result in the best model for large corporate debt LGD model out-of-time. The unstable relationship between LGD and explanatory variables remains a key challenge for corporate debt LGD modeling, thus ongoing model performance monitoring and benchmarking are essential for sound model risk management.

The remainder of the paper is organized as follows. In Section 2, we discuss the data and methodology. We present empirical results in Section 3 and draw conclusions in Section 4.

2. Data and methodology

2.1. Data

Our data is based on Moody's Ultimate Recovery Database (URD), which covers U.S. corporate default events with over $50 million in debt at the time of default. We constructed the data at the end of 2014. Table 1 shows the number of observations and

[3] See, for example, Witten et al. (2017).

[4] We would like to thank an anonymous referee and the associate editor for suggesting us to include this CV method.

Table 1
Summary statistics of Loss Given Default (LGD).

Year	Counts	LGD		
		Mean	Median	Standard deviation
1985	2	0.8561	0.8561	0.0023
1987	39	0.3304	0.3750	0.3127
1988	54	0.6103	0.7947	0.3578
1989	72	0.6164	0.6918	0.3047
1990	179	0.5650	0.6111	0.3719
1991	256	0.4086	0.3530	0.3718
1992	103	0.4370	0.4115	0.3975
1993	96	0.4529	0.5797	0.3954
1994	58	0.3554	0.3306	0.3626
1995	110	0.3602	0.3052	0.3705
1996	66	0.3742	0.2100	0.3866
1997	60	0.3747	0.2030	0.3948
1998	73	0.5886	0.6420	0.3420
1999	180	0.4574	0.3976	0.3904
2000	281	0.4964	0.5432	0.3905
2001	573	0.5225	0.6421	0.4033
2002	688	0.5517	0.6818	0.3467
2003	335	0.3311	0.2600	0.3619
2004	175	0.2571	0.0175	0.3380
2005	182	0.2538	0.1255	0.3055
2006	68	0.2660	0	0.3548
2007	35	0.1916	0	0.2859
2008	66	0.5226	0.5277	0.3683
2009	713	0.3563	0.1874	0.3861
2010	128	0.3920	0.2420	0.4083
2011	101	0.2932	0	0.3743
2012	67	0.3914	0.3490	0.3794
2013	33	0.3777	0.4002	0.3328
Overall	4793	0.4321	0.3989	0.3861

the mean LGD by each default year.[5] There are a total of 4793 observations from 1985 to 2013 in the full sample. Default counts peaked in 1991, 2002 and 2009.[6] LGD peaked around 1989, 1998, 2002 and 2008. The remaining columns of Table 1 suggest that LGDs show large variations within each year and from year to year.

We use 22 explanatory variables (the same as in Qi and Zhao, 2011), and the summary statistics are listed in Table 2. Explanatory variables observed in year t-1 are used to predict LGD in year t.

Fig. 1 shows the distribution of the LGDs. It is worth noting that 30 percent of the observations in the sample have LGD values equal to 0, and roughly 15 percent of the observations have LGD values equal to 1, clearly showing a strong bi-modal distribution.

2.2. Methodology

2.2.1. ML models

The ML methods we investigate in this paper include nearest neighbor (unweighted and distance weighted), regression tree (RT), random forest (RF) (Breiman, 2001, 2004; Ernst and Wehenkel, 2006; Genuer et al., 2008), stochastic gradient boosting (XGBoost) (Friedman, 2001, 2002; Chen and Guestrin, 2016), neural network (NN) (Bishop, 1995), and support vector regression (SVR) (Vapnik, 1995; Suykens et al., 2003; Wang and Hu, 2005). A brief summary of the ML methods is provided in Appendix.

We use Scikit-learn (Pedregosa et al., 2011) to estimate the ML models.[7] We have also checked some results with R packages, and the R results are not qualitatively different. We use grid search to find the hyper-parameters. Table A.1 lists the important hyper-parameters and other modeling choices included in grid search for each model, as well as pre-determined modeling choices that are not part of the grid search.

2.2.2. Hyper-parameter tuning

We employ three methods to construct the CV samples in order to select hyper-parameters for the ML models.

[5] URD has three alternative approaches of calculating recovery: the settlement method, the trading price method, and the liquidity event method. The database also provides Moody's preferred method, which varies for each default and is the one Moody's consider the most representative of the actual recovery. The most common, preferred method is the settlement method. We use the LGD numbers from the preferred method for all of our analysis. To obtain discounted ultimate recoveries, each nominal recovery in URD is discounted back to the last time when interest was paid using the instrument's pre-petition coupon rate.

[6] Table 1 shows that the number of observations for 2012 and 2013 is rather low, possibly because the economy was in recovery and/or a significant proportion of the defaults that occurred in 2012–2013 might not have been resolved by 2014, the year when we constructed the data (our subscription of the data stopped in 2014).

[7] The gradient boosting algorithm we use, XGBoost (Chen and Guestrin, 2016), is available as a separate Python package. This package is able to interface with Scikit-learn, which allows us to seamlessly integrate XGboost into our analysis.

Table 2
Summary statistics of explanatory variables.

	Mean	Median	Standard deviation
Seniority index	0.4920	0.5000	0.2574
Debt seniority types			
Revolver	0.2035	0	0.4026
Term loan	0.1924	0	0.3942
Senior secure	0.1229	0	0.3284
Senior unsecure	0.2813	0	0.4497
Senior subordinated	0.1070	0	0.3091
Junior	0.0149	0	0.1212
Collateral types			
Capital stock	0.0379	0	0.1910
Equipment	0.0264	0	0.1604
Guarantees	0.0028	0	0.0526
Intellectual	0.0013	0	0.0357
Inter-company debt	0.0002	0	0.0146
Inventory, accounts receivable and cash	0.0373	0	0.1895
Other assets	0.0121	0	0.1095
Unsecured	0.4973	0	0.5000
Second lien	0.0439	0	0.2049
Third lien	0.0075	0	0.0860
Industry distance to default	0.1254	0.1216	0.0715
Aggregate default rate in the economy	0.2275	0.2021	0.1472
Stock market returns	−0.0219	−0.0081	2.1229
T-bill rate	0.2973	0.2190	0.2246
Utility dummy	0.0609	0	0.2392

Table 3
Fit statistics of learning methods under shuffled five-fold cross-validation.

	R-squared (R^2)			Mean squared error (MSE)			Spearman correlation		
	Left-out-folds	Out-of-sample	Out-of-time	Left-out-folds	Out-of-sample	Out-of-time	Left-out-folds	Out-of-sample	Out-of-time
Tobit	0.398	0.388	0.544	0.088	0.094	0.075	0.654	0.643	0.765
Nearest neighbor	0.516	0.541	0.411	0.071	0.071	0.098	0.732	0.746	0.674
Distance weighted nearest neighbor	0.704	0.716	0.543	0.043	0.044	0.076	0.833	0.840	0.791
Regression tree	0.528	0.572	−0.634	0.069	0.066	0.272	0.750	0.772	−0.140
Random forest	0.704	0.737	0.454	0.043	0.041	0.091	0.837	0.850	0.766
Stochastic gradient boosting: XGBoost	0.712	0.745	0.407	0.042	0.039	0.099	0.841	0.856	0.681
Neural network	0.536	0.565	0.467	0.068	0.067	0.089	0.743	0.758	0.766
Support vector regression	0.525	0.562	0.084	0.070	0.067	0.153	0.725	0.747	0.431
Nearest neighbor (bagged)	0.565	0.589	0.506	0.064	0.063	0.082	0.758	0.769	0.752
Weighted nearest neighbor (bagged)	0.701	0.709	0.577	0.044	0.045	0.070	0.835	0.838	0.803
Regression tree (bagged)	0.644	0.683	0.509	0.052	0.049	0.082	0.806	0.822	0.769
Neural network (bagged)	0.552	0.573	0.459	0.066	0.066	0.090	0.752	0.761	0.797
Support vector regression (bagged)	0.520	0.557	0.066	0.070	0.068	0.156	0.727	0.748	0.437

This table presents the results from the shuffled five-fold cross-validation using 1985–2009 development sample and 2010 out-of-time sample.

2.2.2.1 Shuffled K-fold CV The first method is a standard shuffled K-fold cross-validation. We first construct a training sample and an out-of-sample validation sample by completely and randomly reshuffling the entire data from the estimation period and then partitioning it further. For example, when we estimate the models using the 1985–2009 data, we first completely reshuffle all 4464 observations from 1985–2009 using the function train_test_split() in the Scikit-learn (Pedregosa et al., 2011) python package and setting the shuffle option to the default value "True" (i.e., to shuffle the data). We then use 75% of the observations as the training sample and the remaining 25% as out-of-sample. We select the hyper-parameters using five-fold cross-validation on the 75% training sample data (further partitioning the training sample into five folds and iteratively using four folds as in-sample (80%) and one fold as the left-out cross-validation sample (20%)) using the function RepeatedKFold().[8] Hyper-parameters are chosen by minimizing the average mean squared error (MSE) across the 5 left-out folds. We then evaluate model performance using the out-of-sample

[8] We use the function RepeatedKFold() instead of the function KFold() in Scikit-learn since the RepeatedKFold() generates more stable hyper-parameters. Since the 75% sample are already shuffled, RepeatedKFold() is similar to the KFold() function, regardless of whether the shuffle option in KFold() is set to 'True' or 'False'. We have tried the KFold() function at this step and the models using hyper-parameters yielded by KFold() show similar performance as those based on RepeatedKFold().

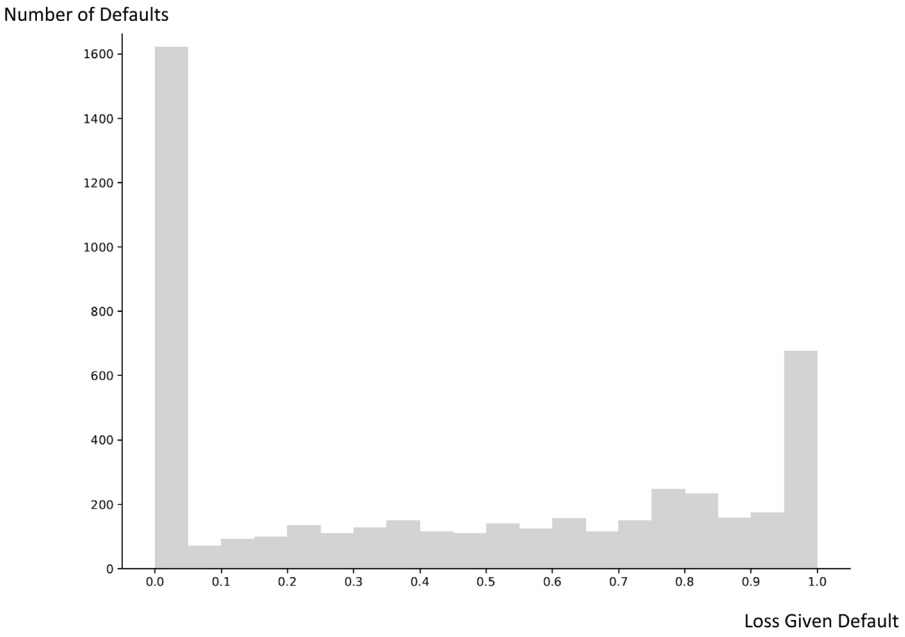

Fig. 1. Distribution of LGD, 1985–2013.

and out-of-time data.[9] This approach, which completely breaks the chronological order in the data when constructing training and out-of-sample testing data, is the standard approach in the ML literature and common industry practice.

2.2.2.2 Unshuffled K-fold CV In the unshuffled K-fold CVs, we set the shuffled option in the train_test_split() function to be "False". As a result, we maintain the chronological order in the data when constructing the training and out-of-sample data. For example, when we estimate the models using the 1985–2009 data, the 75% training sample is from the period 1985–2004 and the 25% out-of-sample is from the period 2004–2009. Therefore, the out-of-sample in this CV method is essentially out-of-time. The 75% is then partitioned into 5 folds for hyper-parameter selection using the function KFold(). We use the default setting "unshuffled" so that each of the five folds retains its own chronological order. However, the chronological order of the training data is not strictly maintained, since the left-out fold could be the first, second, third, fourth or fifth fold of the data from the 1985–2004 period for the five-fold CV. Hyper-parameters are again selected by minimizing the average MSE across the 5 left-out folds.

2.2.2.3 Sequential blocked CV In the third CV method, we always use data from the later periods as the left-out sample in hyper-parameter selection. We call this method sequential blocked CV and implement it using the function TimeSeriesSplit () in Scikit-learn. This method breaks the estimation sample (for example, 1985–2009) into five folds chronologically with roughly the same number of observations in each fold.[10] The model is first estimated in the first fold (i.e., from the earliest years) and validated in the second fold. The model is then estimated using data from the first and second folds and validated in the third fold and so on until the fifth fold is used for validation. Hyper-parameters are chosen by minimizing the average MSE across the 4 left-out folds. By appending subsequent folds of the sample to the first fold which is always used to train and tune the model, the sequential blocked cross-validation approach may better accommodate the time-varying relation between the dependent variable and the explanatory variables.

The sequential blocked CV might show improvements from the unshuffled K-fold CV. First, each left-out fold in the sequential blocked 5-fold CV uses 20% of the in-sample, which is more than the 15% in the shuffled or unshuffled CV (20% of the 75% training sample, i.e., 15%, of the in-sample due to the 25% for out-of-sample testing). More importantly, the sequential blocked CV keeps the chronological order of the data completely as it always use data from the later periods as the left-out sample, whereas in the unshuffled K-fold CV, the left-out fold can be from anywhere in the training sample period. As a result, the sequential blocked CV might handle over-time variations better than the unshuffled K-fold.

2.2.3 Bagging

Bagging stands for bootstrap aggregation. It is a meta-estimator which trains any estimator repeatedly on bootstrapped samples. These estimators are used individually to predict an outcome for a new observation and the individual estimates are aggregated as means into a final prediction. In this study, we use 500 bootstrap samples to train bagged models. We do not employ bagging on random forest or stochastic gradient boosting as these are themselves ensemble methods, composed of many simpler learners.

[9] Hyper-parameters are selected once for each model-sampling structure. The 1985–2009 data is used for these parameter searches. Given these selected parameters, models are then trained and evaluated on the corresponding data sets.

[10] All observations from the training sample and out-of-sample datasets described for the shuffled five-fold cross-validation are included in this partition.

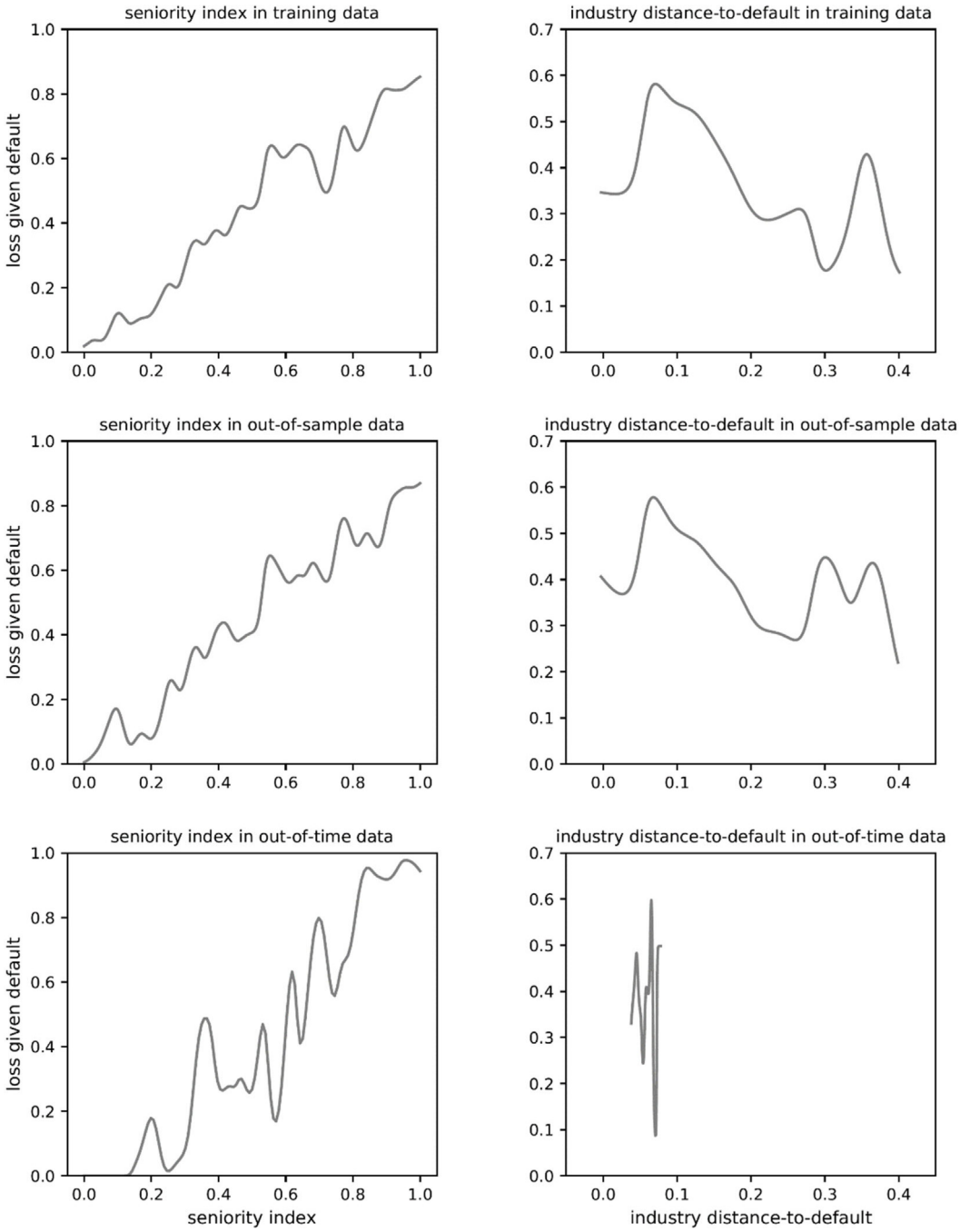

Fig. 2. Kernel regressions of LGD on the two most influential variables in different samples.

3 Empirical results

We report performance measures along two dimensions: predicative accuracy and rank ordering. For predictive accuracy, we use R-squared (R^2) and MSE, and we use Spearman correlation to measure rank ordering.

In Section 3.1, we report the training and out-of-sample results using data up to 2009 and then use 2010 as the out-of-time sample. In Tables 3 to 5 and Figs. 3–6, we report 13 sets of results from Tobit, seven ML models and five bagged ML models. Table 6 lists the values of the selected hyper-parameters. Robustness analysis based on alternative samples are reported and discussed in Section 3.2.

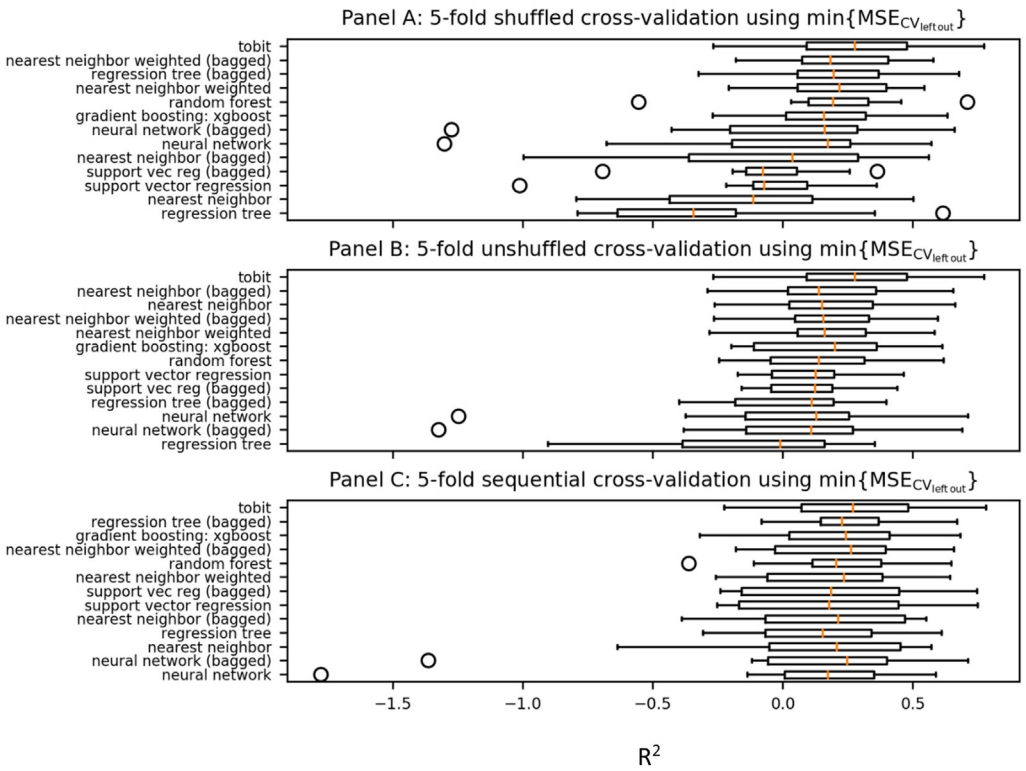

Fig. 3. Distribution of R^2 for each model from out-of-time samples, 2003–2013. Methods are ordered from the top to bottom by means of R-squared (R^2). In each box plot, we show the median (orange vertical line) and the 25th and 75th percentiles with the boxes. The end of the whiskers represents the values that are located within 1.5 times the inter-quartile range (IQR) beyond the upper and lower quartiles and the circles depict the values that are more extreme. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.1 Results on the development sample from 1985–2009

3.1.1 Shuffled K-fold CV

Results in Table 3 are based on the shuffled K-fold CV method as described in Section 2.2.2.1. The first column shows the average R^2 values of the left-out folds, while the second column reports the R^2 values of the out-of-sample. Comparing columns 1 and 2 of each fit statistic, we can see that the CV results are similar to the out-of-sample results, suggesting that the hyper-parameters we choose do not suffer from severe problems of over-fitting. All ML methods outperform Tobit by a large margin in both the training and out-of-sample testing data.

However, the out-of-time performance of these ML methods is not very good. The out-of-time R^2 for Tobit (0.544) actually improves from both the left-out folds (0.398) and out-of-sample (0.388), and we can see from the third column of Table 3 that Tobit has the second highest out-of-time R^2 among all methods. The method with the highest out-of-time R^2 is the bagged weighted nearest neighbor, while the weighted nearest neighbor is a close third. The relative ranking of the methods is similar based on MSE.

The relative ranking of the methods based on Spearman correlation is different from that based on R^2 and MSE. The bagged weighted nearest neighbor, the bagged neural network, and the weighted nearest neighbor make up the top three, while Tobit ranks seventh based on Spearman correlation.

Regression tree performs the worst out-of-time based on all three performance statistics in Table 3. This is not surprising, since regression tree is prone to over-fitting that cannot be adequately resolved by shuffled K-fold cross-validation.

From Table 3, we can clearly see that the efficacy from bagging varies across the ML methods. Bagging leads to significant lift in performance for models that are prone to over-fitting, such as regression tree. Results from bagged regression tree even surpass those from more complicated tree-based methods, such as random forest and stochastic gradient boosting. However, NN and SVR do not benefit from bagging.

To understand why some of the ML methods are not able to provide more accurate out-of-time prediction than Tobit in Table 3, we identify the two most important factors driving LGDs from the list of explanatory variables and plot the kernel regressions in Fig. 2 for the training, out-of-sample, and out-of-time data.[11] The seniority index is the most important driver of LGD, followed by

[11] Bandwidths of these kernel regressions is 0.5 for seniority index and 0.005 for the industry distance-to-default.

Table 4
Fit statistics of learning methods under unshuffled five-fold cross-validation.

	R-squared (R^2)			Mean squared error (MSE)			Spearman correlation		
	Left-out-folds	Out-of-sample	Out-of-time	Left-out-folds	Out-of-sample	Out-of-time	Left-out-folds	Out-of-sample	Out-of-time
Tobit	0.413	0.034	0.571	0.085	0.133	0.071	0.669	0.512	0.762
Nearest neighbor	0.424	0.101	0.543	0.083	0.124	0.076	0.662	0.531	0.758
Distance weighted nearest neighbor	0.528	0.108	0.527	0.068	0.123	0.079	0.734	0.521	0.763
Regression tree	0.371	−0.065	−0.060	0.090	0.147	0.177	0.630	0.410	0.457
Random forest	0.511	0.181	0.445	0.070	0.113	0.092	0.723	0.552	0.784
Stochastic gradient boosting: XGBoost	0.505	0.228	0.471	0.071	0.107	0.088	0.720	0.583	0.728
Neural network	0.444	−1.221	0.556	0.080	0.307	0.074	0.686	0.386	0.796
Support vector regression	0.435	0.089	0.239	0.081	0.126	0.127	0.676	0.397	0.675
Nearest neighbor (bagged)	0.428	0.074	0.527	0.082	0.128	0.079	0.664	0.527	0.768
Weighted nearest neighbor (bagged)	0.522	0.120	0.531	0.069	0.122	0.078	0.731	0.541	0.783
Regression tree (bagged)	0.450	0.014	0.111	0.079	0.136	0.148	0.685	0.433	0.507
Neural network (bagged)	0.452	−0.814	0.540	0.079	0.250	0.077	0.691	0.427	0.797
Support vector regression (bagged)	0.434	0.089	0.222	0.081	0.126	0.130	0.674	0.403	0.664

This table presents the results from the unshuffled five-fold cross-validation using 1985–2009 development sample and 2010 out-of-time sample.

Table 5
Fit statistics of learning methods under sequential blocked cross-validation.

	R-squared (R^2)		Mean squared error (MSE)		Spearman correlation	
	Left-out-folds	Out-of-time	Left-out-folds	Out-of-time	Left-out-folds	Out-of-time
Tobit	0.403	0.531	0.089	0.077	0.655	0.767
Nearest neighbor	0.370	0.509	0.094	0.082	0.619	0.719
Distance weighted nearest neighbor	0.892	0.641	0.016	0.060	0.944	0.813
Regression tree	0.465	0.512	0.080	0.081	0.686	0.766
Random forest	0.957	0.504	0.006	0.083	0.964	0.779
XGBoost	0.695	0.527	0.045	0.079	0.839	0.772
Neural network	0.565	0.477	0.065	0.087	0.760	0.785
Support vector regression	0.479	0.560	0.077	0.073	0.718	0.768
Nearest neighbor (bagged)	0.442	0.527	0.083	0.079	0.677	0.734
Weighted nearest neighbor (bagged)	0.918	0.657	0.012	0.057	0.950	0.834
Regression tree (bagged)	0.466	0.492	0.079	0.085	0.695	0.809
Neural network (bagged)	0.562	0.452	0.065	0.091	0.758	0.791
Support vector regression (bagged)	0.481	0.554	0.077	0.074	0.718	0.765

This table presents the results from the sequential blocked cross-validation using 1985–2009 development sample and 2010 out-of-time sample.

industry distance-to-default (Qi and Zhao, 2011) and these two risk drivers account for 34% of the variation in LGD based on OLS regression.[12] We can see that the relation between the seniority index and LGD is largely stable across the three samples, but there is substantial change in industry distance-to-default. It ranges between 0 to 0.4 in the training and out-of-sample but is no greater than 0.08 in the out-of-time sample. The relation between LGD and industry distance-to-default from the training/out-of-sample to out-of-time sample might have thus changed dramatically, resulting in the poor out-of-time performance for several ML models, such as the regression tree, support vector regression (with and without bagging).

3.1.2 Unshuffled K-fold CV

Table 4 reports results from the unshuffled K-fold CV as described in Section 2.2.2.2. We can see that Table 4 shows lower R^2, higher MSE, and lower correlation than Table 3 for all 12 ML methods in the left-out folds, and the out-of-sample performance measures in Table 4 are much worse than those in Table 3. Note that the out-of-sample in Table 4 is essentially out-of-time. As the results in Table 4 are unshuffled, and the out-of-sample is from the last one fifth of the 1985–2009 period, consisting largely of LGD observations from the 2007–2009 financial crisis. By contrast, the out-of-sample results in Table 3 are in-time results.

From Table 4, we can see this out-of-sample is particularly difficult to predict. Most of the R^2 in the second column of Table 4 are below 10%, with NN demonstrating the lower R^2 exceeding −100%, suggesting that it completely fails for this sample. Similarly, the out-of-sample MSEs typically double those from left-out-folds. None of out-of-sample correlation in Table 4 exceeds 0.6, while none of the out-of-sample correlations fall below 0.6 in Table 3.

However, the out-of-time results for most ML methods are better in Table 4 than in Table 3, with NN and SVR reaping the most benefit from the change in sample shuffling. Eight out of the 12 ML methods show higher R^2 and lower MSE in the out-of-time

[12] Seniority index and industry distance-to-default are also the most important variables for each of the ML models investigated in this study.

Table 6
Hyper-parameters selected in grid search.

	Shuffled 5-fold cross-validation	Unshuffled 5-fold cross-validation	Sequential 5-fold cross-validation
Nearest neighbor	4 closest neighbors; 17 best features included in distance calculation.	57 closest neighbors; 10 best features included in distance calculation.	90 closest neighbors; lasso feature selection, L1 penalty 0.21.
Nearest neighbor weighted	404 closest neighbors; lasso feature selection, L1 penalty 0.194.	137 closest neighbors; 10 best features included in distance calculation.	248 closest neighbors; lasso feature selection, L1 penalty 0.21.
Regression tree	At least 5 observations per leaf; tree depth of 42.	At least 15 observations per leaf; tree depth of 6.	At least 144 observations per leaf; tree depth of 9.
Random forest	At least 1 observation per leaf; 17 features randomly selected as candidates for each split decision.	At least 1 observation per leaf; 5 features randomly selected as candidates for each split decision.	At least 1 observation per leaf; 2 features randomly selected as candidates for each split decision.
XGBoost	All observations included in each tree; learning rate 0.05; tree depth of 8; at least 5 observations per leaf; 75% of features selected for each tree; 75% of these features selected for each depth level; gamma 0.0005; L1 regularization 0.075; L2 regularization 0.075.	All observations included in each tree; learning rate 0.025; tree depth of 8; 25% of features selected for each tree; 75% of these features selected for each depth level; gamma 0.0001; L1 regularization 0.05; L2 regularization 0.025.	25% of observations included in each tree; learning rate 0.05; tree depth of 5; at least 50 observations per leaf; 50% of features selected for each tree; 50% of these features selected for each depth level; gamma 0.000275; L1 regularization 0.3; L2 regularization 0.05.
Neural network	100 units in single hidden layer; L2 regularization 2.2.	75 units in single hidden layer; L2 regularization 8.0.	10 units in single hidden layer; L2 regularization 4.6.
Support vector regression	Variables standardized; radial basis function kernel; C penalty 0.5; epsilon tube 0.0715; kernel coefficient 2.	Variables standardized; radial basis function kernel; C penalty 0.15; epsilon tube 0.102; kernel coefficient 0.25.	Variables not standardized; radial basis function kernel; C penalty 0.1; epsilon tube 0.0575; kernel coefficient 0.035.

sample in Table 4 than in Table 3. Similarly, the Spearman correlation is higher for 8 out of the 12 ML methods in Table 4 than in Table 3. Therefore, the benefit from unshuffled K-fold CV for out-of-time performance is clear from Table 4. However, it is still hard for ML methods to outperform Tobit in Table 4.

Regression tree again shows the worst performance in Table 4. Except for regression tree, there is little enhancement from bagging. In addition, even with bagging, regression tree still clearly lags the other methods in Table 4.

3.1.3 Sequential blocked CV

Table 5 reports results based on the sequential blocked CV method. This approach uses all the data from 1985–2009 for hyper-parameter selection, so we report results from the left-out-folds and the out-of-time sample (i.e., 2010).

We can see that two-thirds of the 12 ML methods show improvement in R^2 from Tables 4 to 5, and R^2 for 10 out of the 12 ML methods improves from Tables 3 to 5. The out-of-time R^2 across all ML methods is within the range between 0.452 and 0.657, which is a major upgrade from Tables 3 and 4. Even regression tree shows decent R^2 out-of-time in Table 5, which indicates that this CV method can boost model stability. We can see similar lift in MSE and rank correlation. In this table, Tobit only ranks number 5 in terms of both R^2 and MSE and number 9 by Spearman correlation. Therefore, there is an overall increase in goodness-of-fit when the CV method changes from the shuffled K-fold to the unshuffled K-fold, to sequential blocked CV.

3.1.4 Comparison of the three CV methods

From Tables 3 to 5, we can clearly see overall improvements from shuffled K-fold to unshuffled K-fold to sequential blocked CV. We report the hyper-parameters chosen by the three CV methods for the 1985–2009 sample in Table 6 to illustrate where the improvements might originate. We can see from Table 6 that the hyper-parameters chosen by the unshuffled K-fold typically lead to more parsimonious models than those from the shuffled K-fold. For example, the unshuffled approach chooses 57 closest neighbors and 10 best features while the shuffled approach chooses 4 closest neighbors and 17 best features in distance calculations in nearest neighbor. As a result, the prediction function is smoother and less reactive and trading out a single observation for a new observation will have a much smaller impact on the prediction for the unshuffled K-fold CV method. In random forest, the unshuffled approach uses 5 features, while the shuffled approach uses 17 features randomly selected as candidates for each split decision. Fewer candidate variables allow for the selection of a covariate that results in a smaller reduction in the loss function than the "optimal" covariate but may result in larger reductions through interactions with other covariates further down the tree.

We next examine why the sequential blocked CV behaves differently from the unshuffled CV in out-of-time test for 2010 via comparison of the chosen hyperparameters. From Table 6 we can see that, compared to unshuffled CV, the sequential blocked CV generally selects hyper-parameters that are less likely subject to overfitting, for instance, larger values for L1 and L2 regularization in XGBoost, lower C penalty in SVR, more neighbors in nearest neighbor, more observations per leaf for regression tree, and fewer features in random forest. This is likely the primary reason why results from the sequential blocked CV seem more stable in general in Table 5 than those from Table 4.

Sequential blocked CV's advantage in neural network is uncertain. Although it has fewer units in the single hidden layer, its L2 regularization is lower than unshuffled CV. Not surprisingly, if we compare the out-of-time performance between unshuffled CV and

Table 7
Pairwise comparison of the three cross-validation methods.

	Proportion of ML methods where					
	Sequential blocked outperforms unshuffled CV		Unshuffled outperforms shuffled		Sequential blocked outperforms shuffled	
	R^2	Spearman correlation	R^2	Spearman correlation	R^2	Spearman correlation
2003	66.67%	83.33%	66.67%	41.67%	91.67%	50.00%
2004	100.00%	91.67%	50.00%	83.33%	100.00%	83.33%
2005	16.67%	16.67%	83.33%	66.67%	58.33%	33.33%
2006	75.00%	50.00%	66.67%	75.00%	83.33%	83.33%
2007	75.00%	58.33%	83.33%	83.33%	100.00%	91.67%
2008	50.00%	83.33%	58.33%	58.33%	41.67%	66.67%
2009	50.00%	83.33%	50.00%	75.00%	41.67%	75.00%
2010	66.67%	58.33%	66.67%	66.67%	83.33%	83.33%
2011	75.00%	91.67%	58.33%	41.67%	58.33%	91.67%
2012	75.00%	66.67%	33.33%	25.00%	75.00%	58.33%
2013	58.33%	75.00%	66.67%	41.67%	58.33%	66.67%

In this table, we present pairwise comparisons of the three CV methods in terms of R-squared (R^2) and Spearman correlation in the context of rolling out-of-time test. We start with the training sample and out-of-sample from 1985–2002, and we evaluate out-of-time performance for 2003. We then use 1985–2003 for the training sample and out-of-sample and predict out-of-time for 2004. We repeat and incrementally add one more year to estimate the model and test the model on data from the subsequent year. This exercise results in 11 sets of estimation and testing statistics from the 11 out-of-time years. The first column of the first row shows a number 66.67%, which says that among the 8 out of the 12 ML methods (66.67%) generate higher R^2 if sequential blocked CV instead of unshuffled CV is used.

sequential blocked CV in Tables 4 and 5, we can see that unshuffled CV outperforms sequential blocked CV for neural network for 2010.

Very interestingly, unshuffled CV outperforms sequential blocked CV for nearest neighbor in out-of-time fit in Tables 4 and 5. It might be due to different methods for feature selection between the two CV methods. Note that the difference in the number of closest neighbors selected between the two methods (57 versus 90) is not that large.

In summary, the sequential blocked CV, followed by the unshuffled K-fold CV chooses more coarse hyper-parameters. This is probably because preserving the chronological order can help prevent overfitting, especially when there is substantial variation in the relation between the risk drivers and LGDs over the different time periods.

Further, Table 6 shows that the impact from a change in CV method for hyper-parameter selection is larger than that from bagging. For example, bagging does not improve the model fit of support vector regression in Table 3; however, the R^2 increases from 0.084 in the third columns of Table 3 to 0.560 in the second column of Table 5 and a similar improvement related to the cross-validation methods is seen in the bagged support vector regressions. In addition, bagging improved the R^2 of weighted nearest neighbor by around 0.03 from 0.543 to 0.577 in Table 3. By contrast, sequential blocked CV improved the corresponding R^2 by 18% to 0.641 for weighted nearest neighbor in Table 5.

3.2 Robustness check: Rolling out-of-time model performance

The results presented in Section 3.1 are only from one sample period, which may not be representative of the rest of the sample periods. We next investigate model performance using alternative sample periods. We start with the training sample and out-of-sample from 1985–2002, and we evaluate out-of-time performance for 2003. We then use 1985–2003 for the training sample and out-of-sample and predict out-of-time for 2004. We repeat and incrementally add one more year to estimate the model and test the model on data from the subsequent year. This results in 11 sets of estimation and testing statistics from 11 out-of-time years, and we summarize the rolling out-of-time results in the boxplots in Figs. 3 to 6. Figs. 3 and 4 show the R^2 and the R^2 rank of various methods in each year, respectively. Similarly, Figs. 5 and 6 show the Spearman correlation and the ranking of different methods based on the Spearman correlation in each, respectively. In each figure, the top, middle, and bottom panel showing results based on the shuffled, unshuffled, and sequential blocked CV, respectively. When ranking the 13 methods, the best method in a given year is ranked 13 and the worst method is ranked 1.

In each box plot, we show the median (orange vertical line) and the 25th and 75th percentiles with the boxes. The end of the whiskers represents the values that are located within 1.5 times the inter-quartile range (IQR) beyond the upper and lower quartiles and the circles depict the values that are more extreme. In other words, the right whisker typically represents the maximum value. However, if the maximum value is more than 1.5 times the IQR beyond the 75th percentile, represented by the right whisker, then the maximum value is shown as a circle instead, as in the case of regression tree at the bottom of Panel A of Fig. 3. Similarly, the left whisker usually represents the minimum value. However, if the minimum value is less than 1.5 times the IQR below the 25th percentile, represented by the left whisker represents, then the minimum value is shown as a circle. Sometimes, there is no whisker for a box plot. For example, in Panel A of Fig. 4, there is no right whisker for Tobit, which indicates that the highest value is less than 1.5 times the IQR beyond the 75th percentile. In this case, Tobit produces the best out-of-time fit in more than three of the 11 years. In each graph, we sort the methods by the means. For example, Panel A of Fig. 3 shows that regression tree has the lowest mean R^2 of the 11 sets of out-of-time results from the shuffled K-fold cross-validation.

Several interesting patterns can be observed from these four figures. First, there is a major shift from Panel A to Panels B and C in both Figs. 3 and 5. A significant proportion of the out-of-time R^2s are negative in Panel A of Fig. 3, especially among support

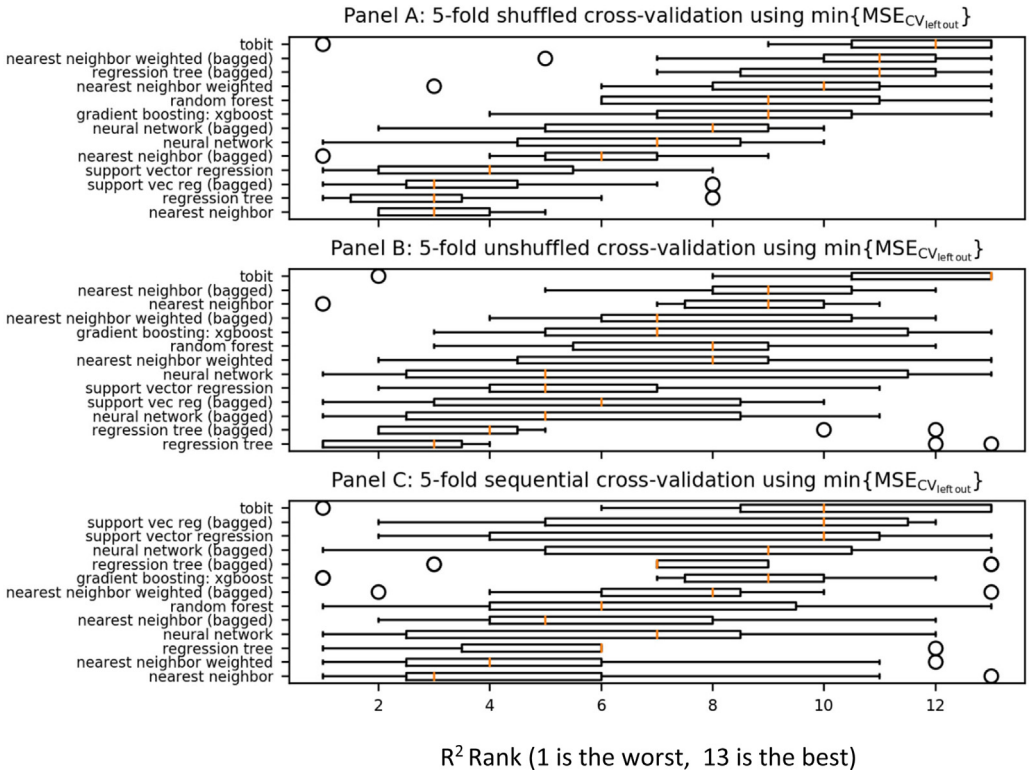

Fig. 4. Distribution of R^2 ranks for each model in out-of-time samples, 2003–2013. Methods are ordered from the top to bottom by mean R-squared (R^2) ranking. In each box plot, we show the median (orange vertical line) and the 25th and 75th percentiles with the boxes. The end of the whiskers represents the values that are located within 1.5 times the inter-quartile range (IQR) beyond the upper and lower quartiles and the circles depict the values that are more extreme. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

vector regressions (both bagged and not bagged), regression tree (not bagged), and nearest neighbor (bagged and not bagged); all four methods show negative R^2 for more than half of the out-of-time predictions. In Panels B and C, the out-of-time R^2 for all four of these methods shift significantly to the right, with their median R^2 well above zero. Further, with the change of the CV approach, we can observe in Fig. 3 a general shift of the boxplots to the right from Panel A to Panels B and C in Fig. 3, with all medians well in the range around 0.2–0.25 in Panel C. For most methods, this change leads to clear improvements. Even though the ranking of bagged weighted nearest neighbor drops from number two in Panel A to number seven in Panel C of Fig. 4, the distribution of the raw R^2 in Panel C of Fig. 3 still shows a rightward shift from Panel A of Fig. 3, and the median R^2 is higher in Panel C than in Panel A for this method.

We can again see from Figs. 5 and 6 that sequential blocked CV can generally boost the out-of-time Spearman rank correlation. Over half of the median Spearman correlations are below 0.6 in Panel A of Fig. 5, while all methods have median Spearman correlation above 0.6 in Panel C of Fig. 5.

From Figs. 3–6, we can clearly see the enhancement from Panel A to Panels B and C, suggesting that keeping the chronological order when splitting the testing samples can significantly boost model stability. There is improvement from Panel B to C, but the performance boost is not the most obvious. We thus further present in Table 7 pair-wise comparisons of the yearly rolling out-of-time R^2 and Spearman correlation among the three CV methods shown in Figs. 3 and 5.

In the first (second) column of Table 7, we calculate the fraction of the 12 ML methods that show higher R^2 (correlation) for sequential blocked CV than the unshuffled CV. The first column of the first line of Table 7 says that when we estimate the models using 1985–2002 data and forecast out-of-time for 2003, 8 out of the 12 (or 66.67%) ML methods yield higher R^2 when using the sequential blocked CV rather than the unshuffled CV to select hyper-parameters. Only one value (for year 2005) in the first column of Table 7 is below 50%. This number 16.67% suggests that, when using the 1985–2004 data and forecast out-of-time for 2005, only 2 ML methods yields higher R^2 using sequential blocked CV than using unshuffled CV, while the remaining 10 ML methods all have higher R^2 using the unshuffled K-fold CV.

Therefore, the sequential blocked CV outperforms or ties with the unshuffled CV in terms of R2 out-of-time in 10 out of the 11 years in our analysis. The second column of Table 7 shows similar results using Spearman correlation, and the sequential blocked CV outperforms unshuffled CV in Spearman correlation out-of-time in all years except 2005.

The third and fourth columns of Table 7 compares unshuffled versus shuffled K-fold CV, and we can see that the unshuffled CV generates R^2s that are higher than or equal to those by the shuffled CV in 10 out of the 11 years and higher Spearman correlations

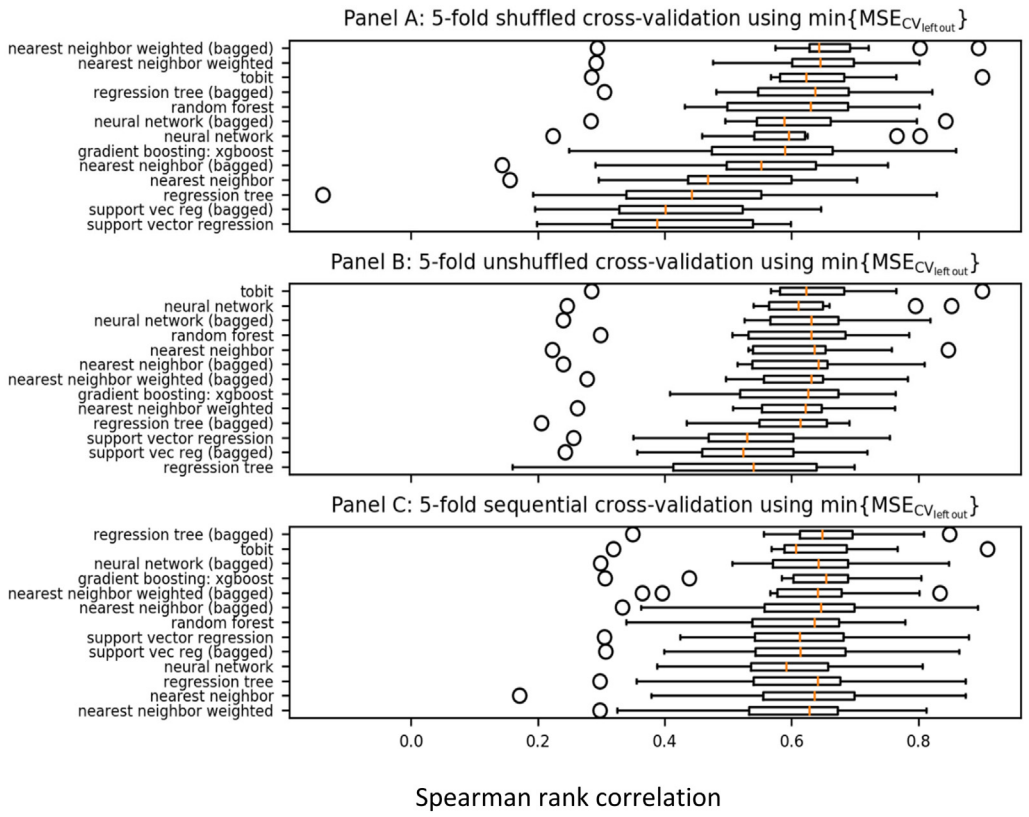

Fig. 5. Distribution of Spearman rank correlation, out-of-time samples, 2003–2013. Methods are ordered from the top to bottom by means of Spearman rank correlation. Methods are ordered from the top to bottom by means of Spearman correlation. In each box plot, we show the median (orange vertical line) and the 25th and 75th percentiles with the boxes. The end of the whiskers represents the values that are located within 1.5 times the inter-quartile range (IQR) beyond the upper and lower quartiles and the circles depict the values that are more extreme. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

in 7 out of the 12 years. In the last two columns of Table 7, we can see that the sequential blocked CV outperforms the shuffled K-fold CV 9 out of the 11 years based on R^2 and 10 out the 11 years based on Spearman correlation.

In summary, results in Table 7 render support to the earlier finding that the sequential blocked CV outperforms the unshuffled CV, and results from the shuffled K-fold CV are the most unstable. In addition, combining results from Figs. 3 and 5 and those in Table 7, we can see that the largest performance improvement is when we switch CV from the shuffled to the unshuffled K-fold CV. There is still enhancement from the unshuffled to the sequential blocked CV, but the degree of performance boost is less substantial. As a result, we conclude that it is important to keep at least some chronological order in the data, and the more chronological order we can keep, the better.

In addition, we can see from Figs. 3 to 6 that, for the same method, the bagged methods are typically ranked higher than the non-bagged ones. Therefore, bagging generally improves model fit. However, the impact from bagging varies depending on the ML method. For example, regression trees improved noticeably from bagging in all four figures, but neural network does not benefit much from bagging in Fig. 3. Therefore, the impact from bagging may vary substantially and it may not be of the first-order importance for most of these ML LGD methods.

Finally, we can see that the inter-quartiles and the whiskers of the graphs in Figs. 3 to 6 show wide ranges for most methods, and no method clearly stands out in these figures. This finding suggests that there is much uncertainty in both the absolute and relative model performance, and good performance of a model in one period does not guarantee decent performance in another period. As a result, performance instability poses a general challenge to LGD models, and LGD modelers can benefit from good model risk management practices of benchmarking and ongoing model performance monitoring.

4 Conclusions

We apply multiple machine learning methods to model loss given default using a common dataset from Moody's. The methods examined include nearest neighbor (with and without weighting), random forest, neural network, stochastic gradient boosting and support vector regression. We primarily focus on evaluating the efficacy of different cross-validation schemes for hyper-parameter tuning and bagging on out-of-time model performance.

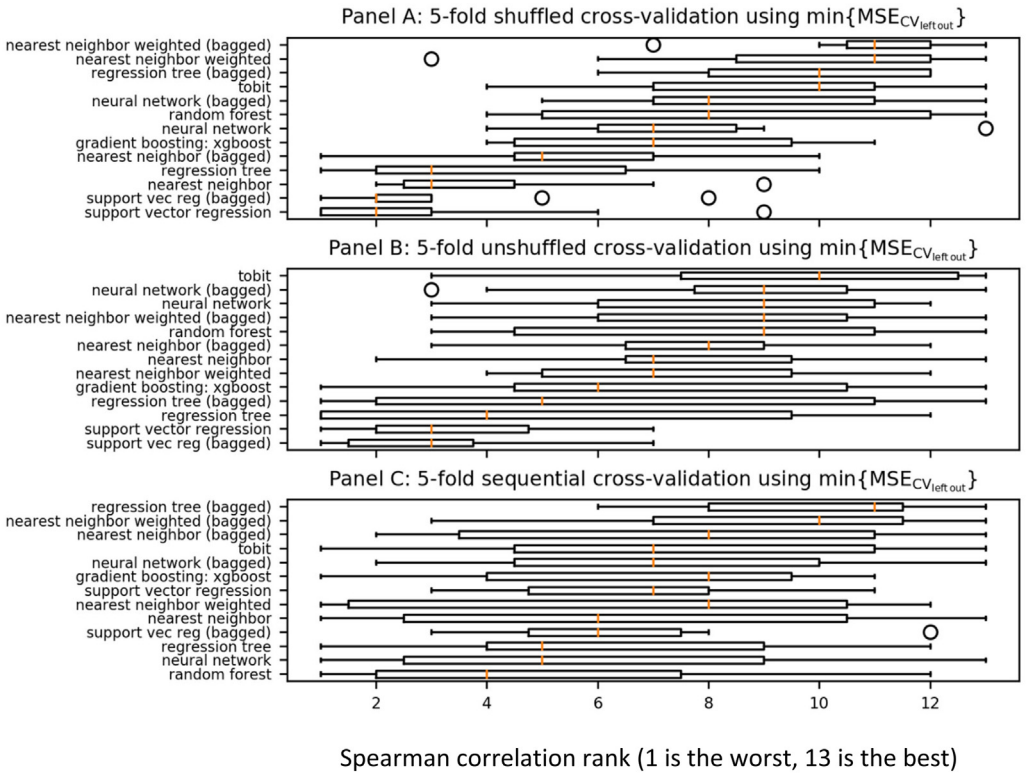

Fig. 6. Distribution of Spearman rank correlation ranks, out-of-time samples, 2003–2013. Methods are ordered from the top to bottom by mean Spearman rank correlation ranking. Methods are ordered from the top to bottom by means of Spearman correlation. In each box plot, we show the median (orange vertical line) and the 25th and 75th percentiles with the boxes. The end of the whiskers represents the values that are located within 1.5 times the inter-quartile range (IQR) beyond the upper and lower quartiles and the circles depict the values that are more extreme. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We find that it is important to retain the chronological order in the data, even though the LGD data is cross-sectional. By keeping some of the chronological order in the data, the unshuffled K-fold cross-validation shows substantial enhancement upon the shuffled K-fold cross-validation. Further, by keeping the chronological order of the data completely, the sequential blocked cross-validation can further improve upon the unshuffled K-fold cross-validation. Among the three cross-validation methods, the sequential blocked cross-validation generates the most parsimonious hyperparameters followed by the unshuffled cross-validation, and the shuffled cross-validation ranks the last in our use case. It is thus worth investigation in future research whether keeping the chronological order of the data is generally more effective than the standard shuffled K-fold cross-validation scheme when modeling cross-sectional data that demonstrates sizeable over-time variation.

We find that bagging can help in some cases (e.g., the regression tree with shuffled K-fold), but the impact from bagging is not consistent across different methods and bagging does not seem to be as effective as cross-validation methods that keep the chronological order in the data.

The primary challenge to model LGD is the unstable relation between the observed explanatory variables and LGD over different time periods. There is substantial uncertainty in relative out-of-time performance among ML and traditional models for large corporate debt LGD. Although sequential blocked CV, unshuffled K-fold CV, and bagging can help, ongoing model performance monitoring and benchmarking are still essential for sound model risk management.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We acknowledge the helpful comments from seminar participants at the Office of the Comptroller of the Currency (OCC). Most work was done when Luke was with the OCC. The views expressed in this paper do not necessarily reflect the views of the Office of the Comptroller of the Currency, the U.S. Department of the Treasury, or any federal agency and do not establish supervisory policy, requirements, or expectations. The authors take responsibility for any errors.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Appendix. Machine learning models

This section provides a brief background for each machine learning estimator used in the study as well as hyper-parameters relevant to each estimator. The values of some of these hyper-parameters are determined by the grid searches, while others are fixed at a level that we found to be reasonable. Table A.1 lists the hyper-parameters we consider for each algorithm. Note that we include choices as hyper-parameters that are not directly tied to the underlying estimator, but are important choices for the model predictions.[13] The values of hyper-parameters are jointly selected as those that minimize the mean squared error (MSE) of the sample left out for cross-validation. As there are multiple left-out folds in both the shuffled and sequential block cross-validation methods, we use the average MSE across these folds as the decisive value for selecting the set of hyper-parameters.

K nearest neighbors, in a regression context, predicts the outcome of an observation as an average of the outcomes of its k nearest neighbors. The concept of "nearest" is a metric, typically the Euclidean distance of the covariates (features). Prior to calculating the distance, the covariates are usually standardized in some manner.[14] As illustrated in Table A.1, our nearest neighbor grid search selects k, the number of neighbors as well as the covariates (features) used to calculate the distance. Three separate covariate selection methods are included in the grid search: the first one using all variables, the second one using a three-fold cross-validation lasso selection method, and the third using the p most important as determined by F statistics from univariate OLS regressions for each covariate.

The nearest neighbor method and the weighted nearest neighbor are similar, except that the former places equal weight to the matching observations, while the latter assigns higher weight to the closer observations, based on their distance in normalized covariate space. The domain of hyper-parameters searched over are identical. In general, the more neighbors that are selected, the more robust the results might be. However, selection of too many neighbors might not be able to capture the non-linearity sufficiently, leading to worse model performance. Another concern involves a loss in performance related to the inclusion of too many or unrelated covariates in the distance function. To reduce this risk, we allow for the restriction of available covariates. When lasso is chosen as the covariate selection method, a higher lambda results in the selection of fewer covariates.

Decision tree incrementally partitions observations in the training data by selecting covariates and cut points to roughly minimize a loss function based on prediction residuals. For regression tree, the loss function is typically the MSE. The resulting model can be represented by a tree with nodes labeled by the splitting covariate and its cut point or geometrically as a partition of the covariate space into hyper-rectangles. Prediction for a new observation occurs by "dropping the observation down the tree" and assigning its value to be the average value of the training data in the terminal node (leaf) in which the observation lands. Overfitting and hence high variance is a common problem with decision trees. As Table A.1 notes, we performed a grid search over the depth of the tree and the minimum number of observations per leaf. More observations per lead or smaller tree depth typically lead to a coarser but more stable tree, which reduces overfitting.

Random forest builds decision trees from multiple bootstrapped samples. A prediction is made for a new observation by dropping the observations down the constructed trees and averaging the predictions made by each tree. The averaging of predictions of individual trees reduces the variability associated with a prediction from a single tree. Random forest also provides a method to de-correlate predictions from pairs of trees, by randomly selecting a subset of the covariates as candidates in each split decision. Fewer candidate variables allow for the selection of a covariate that results in a smaller reduction in the loss function than the "optimal" covariate at that split point, but may result in larger reductions through interactions with other covariates further down the tree. Typically, each tree is allowed to grow to the depth that minimizes the loss function. In our study, we do not restrict tree depth and construct forests consisting of 5000 trees. We perform a grid search to select the number of covariates to include as candidates in a split decision as well as the minimum number of observations per leaf.

Stochastic gradient boosting is a general process of function construction that iteratively uses a simple learning tool to incrementally reduce prediction error on a set of training data. In each step a learner is trained on the residuals of the current model predictions and the resulting predictions are used to update the model. The most commonly employed simple learner is a shallow tree. A learning rate can be used to control the incremental influence of each additional learner on the current function output. The stochastic nature of this algorithm results from a random selection of a proportion of the training sample without replacement to train the current learner. XGBoost is a recent extension of this algorithm that allows for parallel learning and contains additional parameters to control overfitting. In this study, we used 1000 trees and performed a grid search over the learning rate, the depth of the trees, the minimum number of observations per leaf, and other hyper-parameters listed in Table A.1.

The neural network structure used in this paper can be conceived as a directed graph where nodes are organized in three layers. The initial layer of nodes is composed of the covariates and the final layer is the outcome. The middle layer is referred to as the hidden layer. Each node in a layer is connected to the all of the nodes in the previous layer by weights that are determined during the training process. The value of a node in the hidden layer is a function of the weighted sum of the values in the previous layer.

[13] Scikit-learn provides this functionality through the construction of pipelines where the raw dataset is incrementally transformed to finally feed to an estimator.

[14] In this study, features provided to nearest neighbors are normalized on a 0/1 scale. Initial analysis always resulted the choice of this normalization, so we hard-coded it into the data processing pipeline rather than including it as a choice in the formal hyper-parameter search.

Table A.1
Hyper-parameters and pre-processing decisions.

	Included in grid search	Pre-determined
Nearest neighbor and weighted nearest neighbor	Selection of the features to include in distance calculation: all variables, lasso, the p best features as determined by univariate F statistics. Number of closest observations to include in calculating the prediction.	Variables normalized between [0, 1] via the min–max formula. Early exploration always resulted in this as the normalization scheme.
Regression tree	Minimum observations per leaf and tree depth.	Minimizing MSE as the split criterion. All features considered in a split decision.
Random forest	Minimum observations per leaf and number of features randomly selected as candidates at a split point.	5000 trees grown. Trees allowed to grow to full depth. Minimizing MSE as the split criterion.
XGBoost	Learning rate, tree depth, percentage of observations selected without replacement to build the current tree, minimum number of observations per leaf, percentage of features selected as split candidates for each tree, percentage of these features selected as split candidates for each depth level, minimum decrease in loss required to implement a split, L1 regularization, L2 regularization.	1000 trees used as the base learners.
Neural network	Number of units included in hidden layer, L2 regularization.	Variables standardized (Z-score). Early exploration always resulted in this normalization scheme. The neural network algorithm in Sci-kit learn is limited to a single hidden layer. Rectified linear unit is the activation function in the hidden layer. A linear function is used for the final output. The lbfgs optimizer is used to determine the weights.
Support vector regression	Variables standardized or left as-is, C penalty parameter of the error term, epsilon tube radius, kernel coefficient.	Radial basis function kernel.

Once a network is trained, the outcome of a new observation is predicted by calculating the weighted sums for each node in the hidden layer, transforming these results, and then repeating this process with the next set of weights. It is commonly recommended to standardize the covariates prior to use in a neural network. In this study, we use standardized (z-scores) covariates, the reticulated linear unit as the function for each node in the single hidden layer. We perform a grid search to select the number of nodes in the hidden layer as well as the value for L2 regularization. A larger number of nodes allows the network greater flexibility in learning the training data, which can lead to overfitting. A larger value for the L2 parameter shrinks weight values and especially penalizes large values which encourages the use of the full network over a few dominant paths during the training process, discouraging overfitting.

Support vector regression is a method to solve non-linear problems by expanding the solution space into a higher dimension where the solution can be solved more cleanly. This expansion process involves the use of a kernel function to calculate the similarity of training observations while avoiding the direct expansion of the covariate space, often referred to as the "kernel trick". In this study, we use a radial basis function as the kernel, a function that allows for highly non-linear functions within the covariate space. We perform a grid search to allow the data to choose whether the covariates should be standardized, the coefficient of the RBF, the size of the epsilon tube or epsilon-insensitive error measure, and the C penalty. The coefficient of the RBF kernel reduces the similarity measure between two observations as it increases. The penalty parameter C provides a budget for overfitting, so that a smaller value of C reduces the ability to flexibly fit the training data. The epsilon tube provides a minimum distance between an observed value and its predicted value within which, no penalty is incurred during the training process. A larger value for the C budget and the kernel coefficient encourages overfitting, while a larger epsilon tube value decreases the penalty associated with prediction error.

References

Bastos, J., 2010. Forecasting bank loans loss-given-default. J. Bank. Financ. 34 (10), 2510–2517.
Bastos, J., 2014. Ensemble predictions of recovery rates. J. Financ. Serv. Res. 46, 177–193.
Bergmeir, C., Benitez, J.M., 2012. On the use of cross-validation for time series predictor evaluation. Inform. Sci. 191, 192–213.
Bergmeir, C., Hyndman, R.J., Koo, B., 2018. A note on the validity of cross-validation for evaluating autoregressive time series prediction. Comput. Statist. Data Anal. 120, 70–83.
Bishop, C., 1995. Neural Networks for Pattern Recognition. Oxford University Press.
Breiman, L., 1996. Bagging predictors. Mach. Learn. 24, 123–140.
Breiman, L., 2000. Some infinity theory for predictors ensembles. Technical Report. UC Berkeley, US 577.
Breiman, L., 2001. Random forests. Mach. Learn. 45, 5–32.
Breiman, L., 2004. Consistency for a sample model of random forests. Technical Report 670, UC Berkeley, US 670.
Chen, T., Guestrin, C., 2016. XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794.
Ernst, D., Wehenkel, L., 2006. Extremely randomized trees. Mach. Learn. 63, 3–42.
Friedman, J.H., 2001. Greedy function approximation: A gradient boosting machine. Annu. Stat. 29 (5), 1189–1232.
Friedman, J.H., 2002. Stochastic gradient boosting machine. Comput. Statist. Data Anal. 38 (4), 367–378.

Genuer, R., Poggi, J.-M., Tuleau, C., 2008. Random Forests: Some Methodological Insights. Research Report RR-6729, INRIA.
Hand, D.J., 2009. Mining the past to determine the future: Problems and possibilities. Int. J. Forecast. 25, 441–451.
Hartmann-Wendels, T., Miller, P., Tows, E., 2014. Loss given default for leasing: Parametric and nonparametric estimations. J. Bank. Financ. 40, 364–375.
Hurlin, C., Leymarie, J., Patin, A., 2018. Loss functions for loss given default model comparison. European J. Oper. Res. 268, 348–360.
Loterman, G., Brown, I., Martens, D., Mues, C., Baesens, B., 2012. Benchmarking regression algorithms for loss given default modeling. Int. J. Forecast. 28, 161–170.
Nazemi, A., Pour, F.F., Heidenreich, K., Fabozzi, F.J., 2017. Fuzzy decision fusion approach for loss-given default modeling. European J. Oper. Res. 262, 780–791.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É., 2011. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 12, 2825–2830.
Qi, M., Zhao, X., 2011. Comparison of modeling methods for loss given default. J. Bank. Financ. 35, 2842–2855.
Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J., 2003. Least Squares Support Vector Machines. World Scientific Publishing Company.
Tobback, E., Martens, D., Van Gestel, T., Baesens, B., 2014. Forecasting loss given default models: impact of account characteristics and the macroeconomic state. J. Oper. Res. Soc. 65, 376–392.
Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer.
Wang, H., Hu, D., 2005. Comparison of SVM and LS-SVM for regression. In: International Conference on Neural Networks and Brain, Vol. 1. pp. 279–283.
Witten, D., Hastie, T., Tibshirani, R., 2017. An Introduction to Statistical Learning: With Applications in R. Springer Texts in Statistics.
Yao, X., Crook, J., Andreeva, G., 2015. Support vector regression for loss given default modeling. European J. Oper. Res. 240, 52–538.