university of groningen

faculty of economics and business

A comparison of Machine Learning methodologies for proxying the CDS spread

Arjan Faber - S2954370

January 28, 2021

Bachelor's Thesis Econometrics and Operations Research.

Supervisor: Dr. A. Tsvetkov.

Second assessor: Dr. L. Kong.

university of groningen

faculty of economics and business

A comparison of Machine Learning methodologies for proxying the CDS spread

Author: Arjan Faber

Abstract

The bankruptcy of Lehman Brothers and near collapse of AIG in 2008 undermined the confidence in large financial institutions operating on the market. This implied that credit value adjustment (CVA) became an important part of accounting and regulatory standards. The credit default swaps (CDS) spreads of liquid entities are available from the market and used in the calculations of CVA. However, in order to determine CDS spreads of illiquid entities, a proxy method is required. In this research, several machine learning methodologies are considered and compared to current methodologies for proxying the CDS spread. The CDS data comes from several market makers, containing over 2000 CDS entities for the trading days 12 September 2018, 12 September 2019 and 14 September 2020. The predictive accuracy of a proxy methodology is not solely influenced by its design, since it also depends on selecting the right explanatory variables. Therefore, this paper discusses the CDS liquidity proxy, i.e., number of contributors to analyse the effect of CDS liquidity on the CDS spreads and performance of the proxy methodologies. The results show that a single hidden layer artificial neural network (ANN) outperforms the cross-section method, the benchmark method in our research, on all the three trading days, whereas a single hidden layer bayesian neural network (BNN) performs poorly. Furthermore, the results show that both, a simple and efficient regression tree and random forest model, tend to overfit the CDS data. According to the local interpretable model-agnostic explanation (LIME) technique, the most important explanatory variables for the ANN and random forest method belong to the feature rating. In particular, the rating classes that represent a relatively small number of quotes of the data set are considered as the most important explanatory variables for the methods. Also, this paper finds evidence that the CDS liquidity proxy composite depth has a weak relationship with the CDS spreads and does not improve the performance of accuracy of the machine learning models in general.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

From the global financial crisis in 2008 onwards, investors have been using the price of credit default swaps (CDS) as a tool to predict default. A CDS insures the bond buyer against the probability of default of the bond seller. More specifically, a CDS is a derivative contract that involves three parties: a protection seller A, a protection buyer B and a third party C that has a certain default risk and gives party B the incentive to buy insurance from party A. When the protection buyer buys a CDS contract, the counterparty credit risk associated to the underlying bond shifts from the credit risk of the third party to the credit risk of the protection seller. Then, for any other contract that involves company C as the counterparty, the CDS spread that companies A and B traded to each other can be used to assess the credit risk of this company C. More specifically, Flannery et al. (2010) showed that the CDS spread anticipated the agencies rating in advance and therefore found evidence that CDS spreads are very useful for regulators and investors in assessing credit risk.

According to the Basel Committee on Banking Supervision (BCBS), about two-thirds of losses attributed to counterparty credit risk were due to credit value adjustment (CVA) losses and only about one-third were due to actual defaults during the financial crisis. As a consequence, the BCBS introduced a charge to account for counterparty credit risk. This charge is also known as CVA and is the difference between the risk-free portfolio value and the portfolio value after taking into account the counterparty credit risk. In order to obtain the CVA to a portfolio of a company for a particular day, CDS data for that day is required. However, for a large number of entities, CDS quotes are not available from the market. The only possibility to retrieve (artificial) CDS data of these illiquid entities is via proxying the CDS spreads.

Over the past years, researchers have been applying machine learning methods (MLs) on problems in the financial industry. The main reason for this is that MLs can solve complex nonlinear problems that can, e.g, improve (automated) trading and forecasting of stock market trends. Breeden (2020) discusses the application of MLs on credit risk modelling problems. The paper addresses that MLs could be very well suited for small data sets, like credit risk data sets, and not only for big data research problems. However, the paper also addresses that MLs do not always guarantee success and that the choice of MLs and careful assessment of the methods is critical. In particular, the paper shows that for research problems with scarce data, the K-nearest neighbor method performs well, whereas decision trees and neural networks seem to perform poorly. It is interesting to learn more about why a particular machine learning method might work better than a simple linear method. Essentially, MLs are black-boxes and understanding the behaviour of MLs is important in interpreting the performance of these methodologies. Also, when MLs are introduced in research, an important trade-off between simplicity, efficiency and discovering something new about the research problem should be made.

The biggest limitations of the existing CDS proxy methodologies is that these methods as-

sume linearity, use limited information about the illiquid entities, that leads to less accurate prediction performance for both models, and that the methods can only use present daily data and implies that potential useful historical CDS information is not used. The main goal of this paper is to investigate the predictive power of K-nearest neigbors, decision trees and neural networks, and to make a comparison to existing methodologies by only considering information about region, rating and sector of the CDS entities for proxying the CDS spreads for a particular day. Furthermore, to study the behaviour of the candidate methodologies, this paper discusses the effect of a CDS liquidity proxy, i.e., the number of contributors, on the accuracy of the candidate machine learning proxy methods.

In this paper, the current proxy methodologies are reviewed in paragraph 1.2, after which in chapter 2 the relevant literature is discussed. In chapter 3, the used data is described. In chapter 4, the machine learning methodologies are discussed after which in chapter 5 the model comparison metric and evaluation procedure is described. In chapter 6, the calibration of the models is described after which in chapter 7 the results are presented and in chapter 8 the conclusion of this paper is given.

1.2 Current Proxy Methodologies

1.2.1 Intersection Method

As proposed by the European Banking Authority (EBA) in 2013, the proxy CDS spread of an illiquid obligator i on a particular day can be written as follows:

$$y_i^{im} = \frac{1}{m} \sum_{j=1}^{m} y(j), \tag{1}$$

where $m \geq 1$ is the number of liquid spreads with similar characteristics (rating, region and sector) as obligator i and $y(j)$ denotes the spreads in bps of these.[1] This methodology is called the intersection methodology. Sourabh et al. (2018) discussed that the intersection methodology can also be written as a linear model. That is, by using a log transformation of the proxy spreads, the proxy spreads transform from skewed to approximately comform to normal.[2] Hence, the intersection methodology can be written as follows:

$$log(\hat{y}_i^{im}) = \sum_{j=1}^{N-1} a_j I_j(i) + a_0, \tag{2}$$

where N is the total number of buckets, $\hat{y}_i^{im}$ is the proxy spread in bps of an illiquid obligator i, $I_j(i)$ is the indicator function whose value is 1 if the bucket of obligator i equals j, and 0 otherwise. The intercept a_0 represents the log proxy CDS spread of the (A, Financials, Europe). Then, a_j represents the change in log proxy CDS spread from the (A, Financials, Europe) bucket to bucket j. Note that, the choice of intercept bucket does not influence the performance in accuracy of the model according to Sourabh et al. (2018).

[1] 1 bps = 0.01%.

[2] The use of a natural log transformation of the CDS spreads is further discussed in chapter 3.

1.2.2 Cross-Section Method

An alternative method for the intersection method is the cross-section method introduced by Chourdakis et al. (2013b). Instead of using the mean of buckets with liquid quotes similar to entity i, the proxy CDS spread of an illiquid entity i can be written as a product of the characteristic factors rate, sector and region of entity i. This results in the following mathematical expression for the proxy CDS spread for an illiquid entity i on a particular day:

$$y_i^{cs} = M_0 \cdot M_{rating} \cdot M_{sector} \cdot M_{region}, \tag{3}$$

where M_0 is denoted as a scaling factor and M_{rating}, M_{sector}, M_{region} denote the rating, sector and region of entity i, respectively.

A log transformation of the proxy CDS spreads is used in order to write the cross-section methodology as a linear model. Following Sourabh et al. (2018), this gives the following representation for the proxy CDS spread of obligator i on a particular day:

$$log(\hat{y}_i^{cs}) = \sum_{j=1}^{N_{rating}-1} a_{1,j} I_j(i) + \sum_{k=1}^{N_{sector}-1} a_{2,k} I_k(i) + \sum_{l=1}^{N_{region}-1} a_{3,l} I_l(i) + a_0, \tag{4}$$

where N_{rating} is the number of different credit ratings, N_{sector} is the number of different sectors, N_{region} is the number of different regions, $\hat{y}_i^{cs}$ the proxy CDS spread of obligator i in bps, the indicator function $I_j(i)$, $I_k(i)$ and $I_l(i)$ are the indicator functions which are equal to 1, if the rating, sector and region of the ith entity is j, k and l, respectively. The intercept a_0 is the log of the proxy CDS spread of the (A, Financials, Europe) bucket. Then, the coefficients $a_{1,j}$, $a_{2,k}$ and $a_{3,l}$ are the changes of the log proxy CDS spread when going from rating 'A' to rating 'j', the 'financials' sector to sector 'k' and from the 'Europe' region to region 'l', respectively.

2 Literature Review

Chourdakis et al. (2013a) addressed the problem of empty buckets in the intersection methodology. That is, there are combinations of sector, region and rating that contain only illiquid obligators and result in proxy spreads that are undefined. Furthermore, the paper states that the method is unstable over time, e.g., when an entity changes buckets this causes jumps of the proxy spreads for these buckets. Also, the paper shows that the intersection method is likely to underperform due to the non-monotonic CDS spreads for different ratings. Note that, non-monotonicity of CDS spreads imply that the spreads do not increase when moving from rating 'AAA' to 'B' and is an undesirable result.

Chourdakis et al. (2013a) showed that the cross-section method provides monotonic results. Also, the research addresses that the cross-section method solves the problem of undefined proxy spreads obtained in the intersection method and thus is more stable. That is, the cross-section method needs only one liquid quote for a particular region, sector or rating to obtain the proxy spreads for a particular day. The cross-section method can potentially be more

accurate than the intersection method because there is a flexibility to choose combinations of rating, region and sector more precisely. More specifically, each of the top features rating, region and sector are considered separately in the cross-section method. Note that, for the intersection method, the number of empty clusters can be reduced by grouping particular categories in order to reduce the number of clusters. However, this does not always solve the empty bucketing problem (Sourabh et al. (2018)). Overall, the cross-section method can be seen as a robust and stable method compared to the intersection method and therefore will be used as the benchmark method in this paper.

Substantial literature can be found regarding the application of MLs in relation to CDS spread proxying and a selection of these papers are discussed below.

By using CDS data, Luo et al. (2017) showed that deep learning algorithms can provide new perspectives on the credit scoring problem. More specifically, the research showed that the classification performance of a deep belief network outperforms other algorithms such as multinomial logistic regression and support vector machine. The paper addressed the great potential of applying deep learning models to credit scoring problems. In particular, the paper states the flexibility, i.e., nonlinearity of deep learning models to deal with complex data and the consistency of performance of a deep belief network across different rating classes.

Yuankang et al. (2019) showed that historical credit spread data is useful in forecasting credit spreads. The paper considered a modified random forest to forecast the next day movement of credit spreads and showed that the random forest performs poorly on predicting the direction-of-change of the credit spread for the next day. As an alternative, the paper introduced a variant of the neural network called the long short term memory (LSTM) model. The paper shows that the LSTM model outperforms the random forest models with both relatively low RMSE and high accuracy for out-of-sample predictions of the direction-of-change of the credit spread.

Hu et al. (2019) showed that nonlinear machine learning methods, especially the ensemble methods such as bootstrap aggregation (bagging), gradient boosting and random forest, add considerable value to predicting CDS spreads. Also, the paper addresses the interpretability problem, i.e., black-box problem of MLs and considers the Local Interpretable Model-agnostic Explanations (LIME) technique in order to explain the predictions of MLs. As a result, this technique provides better understanding of the prediction results of untrustworthy models according to the paper.

3 Data Description

The CDS data used in this paper is consensus data from several market makers corresponding to a particular trading day. This data set consists of CDS spreads for maturities starting from 6 months up to 10 years. The data set includes the credit rating, sector and region of the quotes. Also, the composite depth, i.e., the number of dealers providing quotes, of the 5 year maturity CDS spread is included in the data set. Since the 5 year maturity CDS quotes are the mostly traded maturities, these CDS spreads are used as our target variable.

In this chapter we discuss the CDS data set corresponding to the trading day Monday 14 September 2020.[3] The total data set consists of 2069 quotes before cleaning. After deleting any quotes with missing data, it can be seen in Figure 1 that the data set consists of many outliers. More specifically, the data set consists of quotes with CDS spreads of 5 years of maturity that are far above a reasonable maximum that would occur during a shock.

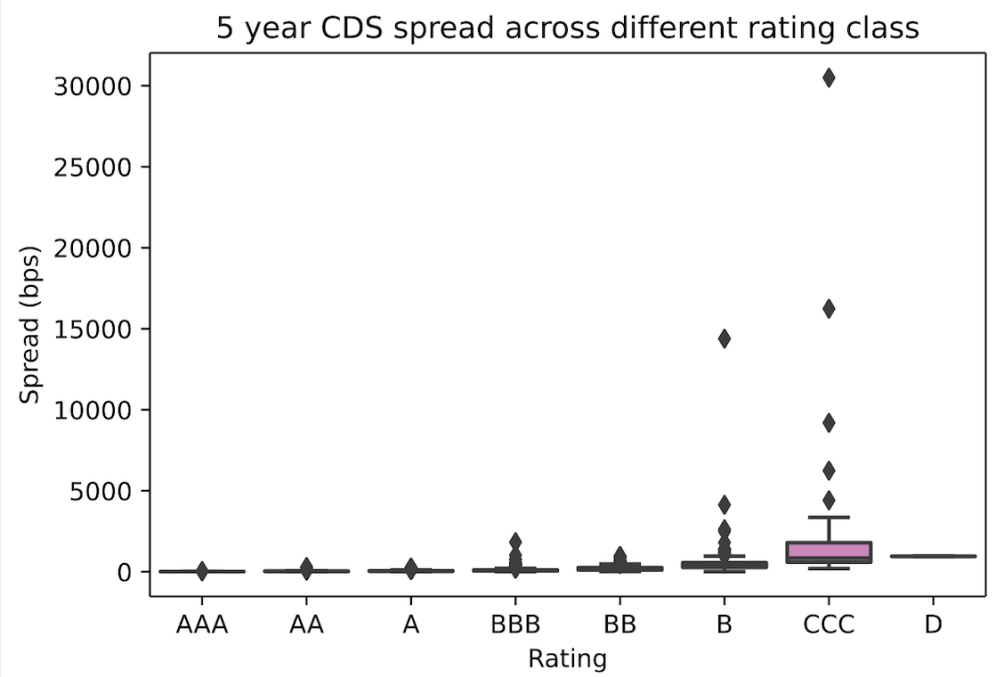

Figure 1: Boxplot (raw data): distribution across ratings for the 5 year maturity CDS spreads based on 14 September 2020 data.

In order to delete outliers, the CDS spreads that fall within the 25th-percentile to 75th-percentile range are kept for each rating class specifically. From Table 1 it can be seen that this procedure of dropping outliers results in 1037 available quotes in the data set. Furthermore, the mean of the CDS spread for the 5 year maturity is 126.9 with a standard deviation of 163.4. This high standard deviation is as expected since the CDS spreads are spread out when considering different combinations of ratings, sectors and regions for the quotes.

[3]For the trading days Wednesday 12 September 2018 and Thursday 12 September 2019 the data exploration and cleaning procedure is very similar.

	Spread5y (bps)
Observations	1037
Mean	126.9
std	163.4
min	11.8
25%	46.6
50%	71.9
75%	123.6
max	1708

Table 1: Statistics cleaned data set based on 14 September 2020 data.

Furthermore, a maximum width of the CDS spread of 1708 bps is obtained. Figure 2 shows the CDS spread distribution per rating class after cleaning the data. From this figure it can be seen that the maximum 5 year maturity CDS spread comes from the 'CCC' rating class. Intuitively, this is a realistic maximum spread, especially when one considers a 'CCC' rated company in a bear market scenario.

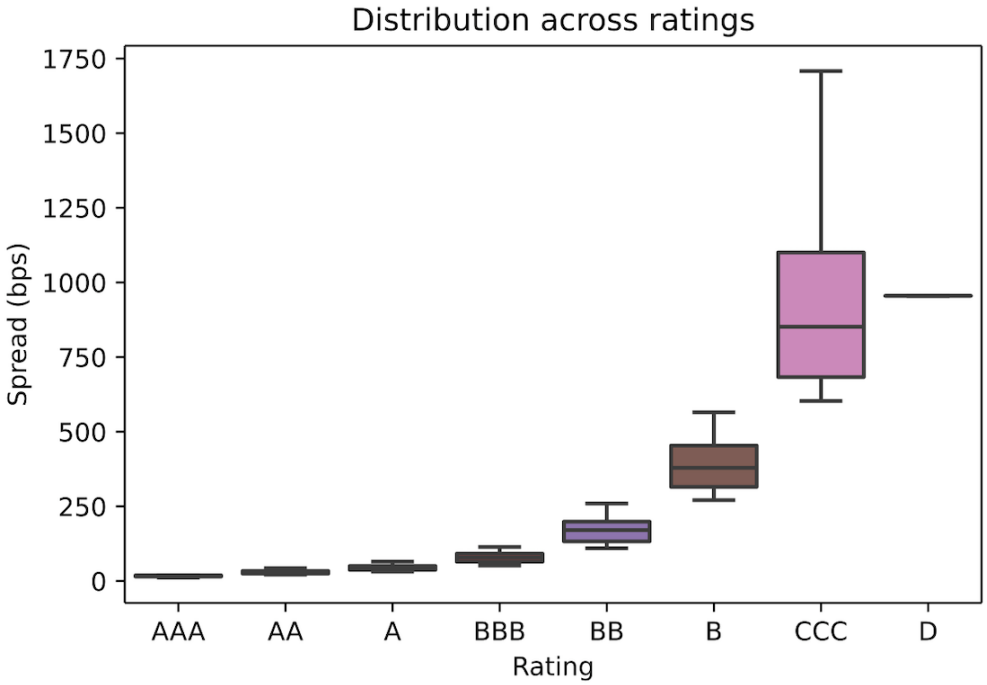

Figure 2: Boxplot (cleaned data): distribution across ratings for the 5 year maturity CDS spreads based on 14 September 2020 data.

From the cleaned data set all the possible buckets for each combination of available rating, sector and region can be found. The number of quotes available per ratings, sectors and region can be found in Table 2. From this table it is interesting to see that approximately 35% of the quotes are considered as investment-grade. Also, the majority of the quotes

correspond to Asian, European and North American quotes, whereas approximately 40% of the data set corresponds to North American quotes alone. Furthermore, most of the quotes correspond to CDS from the financial and industrial sector or correspond to sovereign CDS, whereas there are relatively few quotes from the technology and healthcare sector.

Rating	Observations
AAA	14
AA	79
A	271
BBB	385
BB	172
B	94
CCC	21
D	1

Region	Observations
Africa	9
Asia	213
Caribbean	1
Eastern Europe	22
Europe	285
India	19
Latin America	12
Middle East	13
North America	406
Oceania	45
Offshore	11
Supra	1

Sector	Observations
Basic Materials	69
Consumer Goods	93
Consumer Services	103
Energy	61
Financials	245
Government	106
Healthcare	35
Industrials	139
Technology	40
Telecom	58
Utilities	88

Table 2: Number of quotes per rating, region and sector based on 14 September 2020 data.

For the larger clusters in the data set such as North American and European quotes, we obtain the statistics that are given in Table 3. We can use these statistics to interpret the performances of the candidate methodologies and this will be further discussed in chapter 7.

	North American cluster Spread5y (bps)	European cluster Spread5y (bps)
Observations	406	285
mean	132.9	118.4
std	161.4	137.8
min	15.7	11.8
25%	52.5	50.3
50%	79.6	74.0
75%	130.1	128.0
max	1377.4	1100.5

Table 3: Statistics largest clusters based on 14 September 2020 data.

Based on the combinations of rating, region and sector, all the possible buckets of the CDS quotes can be constructed. That is, there are 8 rating classes, 12 regions and 11 sectors, and results in $8 * 12 * 11 = 1056$ possible buckets. Then, any buckets that are undefined, i.e. the buckets consists of illiquid quotes, can be detected and dropped. This results in a total of 824 empty buckets that are dropped. Consequently, a total of 232 existing buckets based on rating, region and sector are obtained for which the CDS spread of a bucket is found by taking the average of the entities that are similar to a particular bucket. Figure 3 shows the distribution of spreads among different credit ratings within the (North America, Technology) bucket. From this figure it is interesting to see that the 'AA' and 'A' rated quotes in this bucket are close to an approximately normal distribution. That is, the areas of the boxes on both sides next to the medians are approximately equal for both ratings. Furthermore, it can be seen that for the speculative quotes, e.g, the 'BB' rated quotes the data points are more spread out due to the larger difference between the maximum and minimum of the spread for this rating class. Furthermore, the distribution of this rating class, 'BB' rating, is skewed since the areas of the boxes on both sides next to the median are not approximately equal.

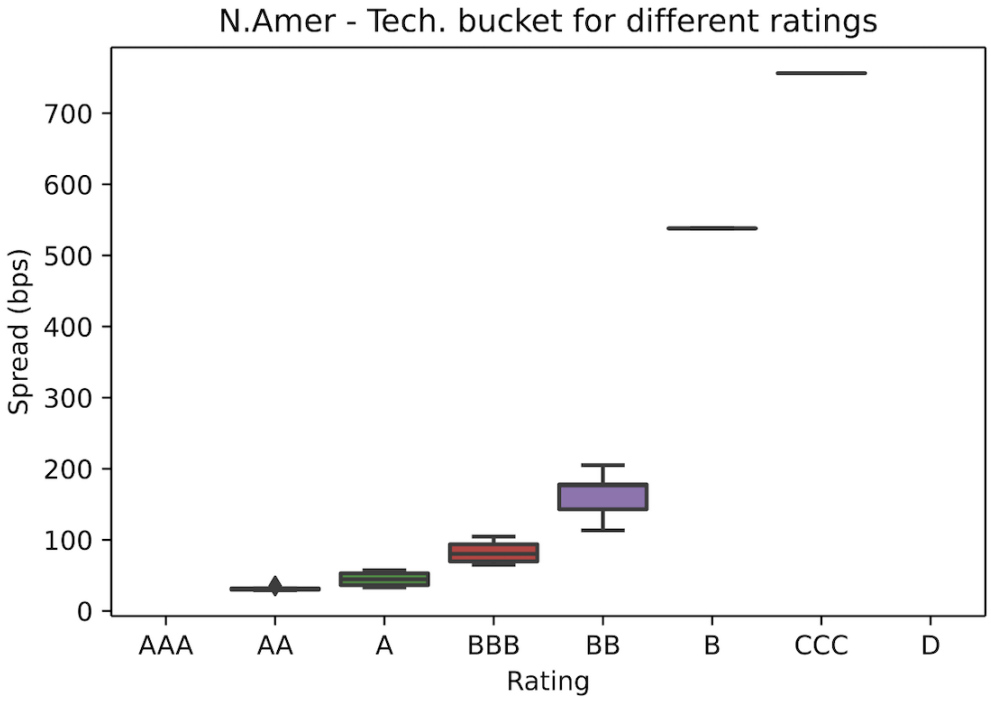

Figure 3: Composition 5 year maturity CDS spreads per rating class: Bucket (North America, Technology) based on 14 September 2020 data.

Table 4 shows that the mean spread over all the buckets is 200.3 (bps), whereas the standard deviation is 252.1 (bps). Since it can be seen from this table that the median is lower than the mean, it can be concluded that the proxy spreads for the buckets are skewed to the right.

	Bucket Spread5y (bps)	Bucket Composite Depth 5y
Observations	232	232
mean	200.3	3.8
std	252.1	1.7
min	11.8	2
25%	46.0	2.8
50%	88.5	4
75%	242.7	5
max	1707	11

Table 4: Statistics buckets based on 14 September 2020 data.

In order to get approximately normally distributed spreads, a transformation on the data is necessary. That is, by taking the natural logarithm of the 5 year maturity CDS spreads of the buckets, it can be seen in Figure 4 that the spreads are approximately normally distributed. Hence, the natural logarithm of the 5 year maturity CDS spreads of the buckets can be used as the target variable for our candidate proxy methodologies.

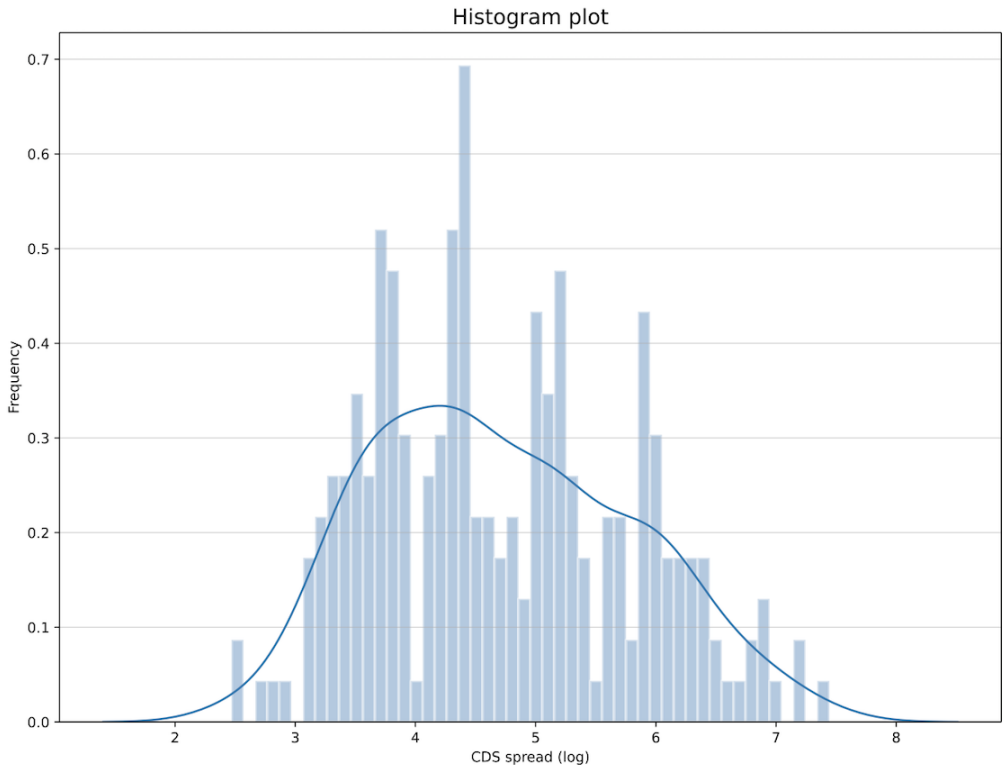

Figure 4: Distribution 5 year maturity CDS spreads (log transformation) of the buckets based on 14 September 2020 data.

Table 3 also shows the mean of the composite depth of the 5 years maturity CDS spread of the buckets. That is, the rounded averages of the composite depths of the 5 year maturity CDS spreads that belong to the same bucket are obtained. It is interesting to see that the average composite depth over the buckets is 3.8 with a standard deviation of 1.7. This implies that approximately 75% of the average composite depths of the buckets fall within one standard deviation from the mean. Based on Markit's liquidity score it can be assumed that spreads with a composite depth greater than 3 are highly liquid. It is likely that CDS entities that are more liquid have more accurate spreads. Therefore, a dummy variable can be introduced that measures the liquidity of a CDS spread, i.e., the dummy variable takes a value 0 if the quote is above 3 (high liquidity) and 1 otherwise (low liquidity). Then, the performance of the proxy methodologies can be analysed for the cases when this liquidity measure is excluded and included.

Note that, in order to use the region, sector and rating variables as input variables for the machine learning methods, the variables are transformed into one-hot-encoded variables in `Python`. More specifically, since the input variables region, sector and rating are categorical variables each feature can be written as a binary variable, i.e., the variable corresponding to a particular feature equals 1 if the CDS entity is characterized by the particular feature and 0 otherwise.

4 Machine Learning Methodologies

4.1 Linear Regression: Extended Cross-Section Model

Beyond the existing literature, the standard cross-section linear model, introduced in paragraph 1.2.2, may be extended by studying the impact of the composite depth of the quotes on the proxy CDS spreads. Therefore, an extra dummy variable is introduced that measures whether the average composite depth of the proxy spreads in each bucket is above a threshold value. Note that, 3 is chosen as our threshold value here and was motivated earlier in this paper. Then, the extended cross-section linear model can be written as follows:

$$log(\hat{y}_i^{cs}) = \sum_{j=1}^{N_{rating}-1} b_{1,j} I_j(i) + \sum_{k=1}^{N_{sector}-1} b_{2,k} I_k(i) + \sum_{l=1}^{N_{region}-1} b_{3,l} I_l(i) + b_4 K(i) + b_0, \qquad (5)$$

where, similar to the standard cross-section model, N_{rating} is the number of different credit ratings, N_{sector} is the number of different sectors, N_{region} is the number of different regions, $\hat{y}_i^{cs}$ the proxy CDS spread of obligator i, the indicator function $I_j(i)$, $I_k(i)$ and $I_l(i)$ are the indicator functions which are equal to 1, if the rating, sector and region of the ith entity is j, k and l, respectively. In addition, the indicator function $K(i)$ whose value is 0 if the bucket of obligator i has a high liquidity score, and 1 otherwise. The intercept b_0 represents the log proxy CDS spread of the (A, Financials, Europe) bucket that has a composite depth above 3 (high liquidity score). The coefficients $b_{1,j}$, $b_{2,k}$ and $b_{3,l}$ are the changes of the log proxy CDS spread when going from rating A to rating j, Financials sector to sector k and from the Europe region to region l, respectively, ceteris paribus. The coefficient b_4 represents the changes in the log proxy CDS spread that follows from a shift to a bucket with a low liquidity score, ceteris paribus.

4.2 K-Nearest Neighbor Regression

4.2.1 Standard Model

Among the supervised machine learning algorithms, the K-NN is considered as one of the simplest and most commonly used algorithms in the field of pattern recognition. The K-NN algorithm stores labeled training data $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{train}$, where $\mathcal{D}_{train}$ is the training data set that consists of N available quotes for a particular day, $\mathbf{y} = y_1, .., y_N, \mathbf{x}_i = (x_{i,1}, .., x_{i,M})$ with $x_{i,j} \in (0, 1)$ for $i = 1, .., N$ and $j = 1, .., M$. That is, M is denoted as the total number of features in the explanatory variable $\mathbf{x}_i$ and y_i the CDS log spread of entity i.[4] Since the target value y_i is continuous the concept of K-NN for regression is considered. That is, the proxy CDS log spread of an illiquid entity q over the K nearest neighbors can be computed as follows:

$$\hat{y}_q^{NN} = \frac{1}{K} \sum_{k=1}^{K} \tilde{y}_k, \qquad (6)$$

[4]Observe that, when A, B, C are denoted as the number of available ratings, sectors and regions respectively, $M = A + B + C$.

where $K < N$ and $\tilde{\mathbf{x}}_k \in \mathbf{x}$ the exploratory variable of a nearest neighbor k with respect to an CDS entity q and $\tilde{\mathbf{y}}_k \in \mathbf{y}$ the corresponding CDS log spread. To find these nearest neighbors to q, the standard Manhattan distance for K-NN defined can be used as follows:

$$d(\mathbf{x}_i, \mathbf{x}_q) = \sum_{j=1}^{M} |x_{i,j} - x_{q,j}|. \tag{7}$$

Then, $d(\mathbf{x}_i, \mathbf{x}_q)$ is simply the number of disagreeing components in features $\mathbf{x}_i$ and $\mathbf{x}_q$. Since our training data consists of many points that share equivalent characteristics of region, rating and sector, a neighbor k is defined as an entity that has at least one disagreeing component compared to CDS entity q, i.e., $d(\tilde{\mathbf{x}}_k, \mathbf{x}_q) \neq 0$. Then, to find the k-th nearest neighbor, the nearest neighbor algorithm sets the current distance $d(\tilde{\mathbf{x}}_k, \mathbf{x}_q)$ as the closest distance to CDS entity q and $\tilde{\mathbf{x}}_k$ as the new most similar vector of features, if the current distance is smaller than the previous closest distance point to features $\mathbf{x}_q$. Note that, due to the binary input variable the problem of similar distances arises. That is, the distance function only measures the number of dissimilarities in features of the CDS entities. In order to deal with multiple nearest neighbors that share a similar distance to entity q, a bootstrapping procedure is applied by summing bootstrapped sample means and divide it by the number of bootstrap samples drawn from the neighbors that share a similar distance to entity q.

The nearest neighbor algorithm is known as a lazy learning algorithm, since first all the distance points are being stored and then the nearest neighbors are identified. It results in minimal need of training because it memorizes the training data to make predictions. As a consequence, the algorithm is computational expensive and takes a relatively long time to find predictions of the illiquid CDS spreads. One of the other drawbacks is that many entities are obtained that share a similar distance value and may result in undesired noise.

4.2.2 Correlation-Based Distance Model

In Fu et al. (2016), the problem of measuring distance between binary variables, resulting in many equal distances, is addressed. As an improvement, an importance based ranking method is introduced. For the categorical input variables, heavier weights are assigned to important bits and small weights are assigned to less important bits. Translating this to our one-hot encoded variables for CDS quotes, the Pearson correlation coefficient to weight the dissimilarities in each top features, i.e., rating, sector and region, of each combination of CDS entities may be considered. For example, when proxying the CDS spread by using the standard nearest neighbors algorithm of an entity characterized as (BBB, Financials, North America), a total of two candidate nearest neighbors is found. Let these be specified as, e.g.,(BBB, Financials, Europe) and (BBB, Technology, North America). Intuitively, these two candidates are not the equivalent. Therefore, using the Pearson correlation coefficient between North American and European CDS derivatives and the Pearson correlation coefficient between Financials and Technology CDS derivatives, the correlations can be compared and used as a new distance measure. That is, for our top-level features: rating, sector and rating, let us denote $x_{i,s,l}$ as the lth sub-feature within top-level feature s of CDS entity i. Then, for n_s the number of sub-features in top-level feature $s \in [1, 2, 3]$, let us define the

correlation distance between $\mathbf{x}_i$ and $\mathbf{x}_q$ as follows:

$$\begin{aligned} R(x_{s,l}, x_{s,n_s}) &= \frac{\sum_{i=1}^{N}(x_{i,s,l} - \bar{x}_{s,l})(x_{i,s,n_s} - \bar{x}_{s,n_s})}{\sqrt{\sum_{i=1}^{N}(x_{i,s,l} - \bar{x}_{s,l})^2}\sqrt{\sum_{i=1}^{N}(x_{i,s,n_s} - \bar{x}_{s,n_s})^2}}, \\ d_{corr}(\mathbf{x}_i, \mathbf{x}_q) &= \sum_{s=1}^{3}\sum_{l=1}^{n_s}\big(1 - R(x_{s,l}, x_{s,n_s+1-l})\big) \cdot |x_{i,s,l} - x_{q,s,n_s+1-l}|, \end{aligned} \tag{8}$$

where $R : \mathbb{R}^2 \to \mathbb{R}$ is the Pearson correlation function with $\bar{x}_{s,l}$ the average value of the lth sub-feature in top-level feature s and $\sum_{s=1}^{3} n_s = M$. Note that, the distances between CDS quotes of interest are now also based on the strength of the relationship between the dissimilar sub-features and not only on the number of the dissimilar features. Also, a higher positive correlation between two dissimilar sub-features of particular CDS quotes of interest results in a smaller distance between these quotes. Hence, the dissimilarities between CDS quotes can be captured and ranked more carefully and may lead to a reduction in noise.

Then, according to the K-NN with a correlation distance measure, $\tilde{\mathbf{x}}^w \in \mathbf{x}$ are the features of the K nearest neighbors with log spreads $\tilde{\mathbf{y}}^w \in \mathbf{y}$ such that the proxy spread of an illiquid entity q can be found as follows:

$$\hat{y}_q^{CNN} = \frac{1}{K}\sum_{k=1}^{K}\tilde{y}_k^w. \tag{9}$$

The major drawback of this method is, however, that the model contains many computations, due to calculating the correlation coefficients of each particular feature dissimilarity. Therefore, this model needs more time to find predictions than the already computational expensive standard K-NN model. Note that, a limitation of our correlation distance function is that we only considered the three top-level features region, rating and sector. Therefore, including a proxy for CDS liquidity will not influence the ordering of nearest neighbors for the proxing the CDS spread.

4.3 Decision Tree Learning

4.3.1 Tree-Based Regression

Tree-based regression analysis falls under the umbrella of the Classification and Regression Trees (CART) introduced by Breiman et al. (1984). The CART learning algorithm is well known for its conceptual simple and efficient way to analyse large data sets. Let us consider $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{train}$ with input variables $\mathbf{x}_i = (x_{i,1}, ..., x_{i,M})$, where M the number of features in the input variable and y_i the associated continuous target value, i.e., the CDS log spread for entity i, for each observation $i = 1, .., N$. The CART algorithm builds a regression tree $\mathtt{T}_1$ by minimizing the Mean Squared Error (MSE) of each node. This can be achieved by selecting the optimal splitting variables, splitting points and shape. More specifically, as a first step to grow a regression tree, the training data $\mathcal{D}_{train}$, containing N entities is split according to a splitting variable j and splitting point $p \in (0, 1)$ into two nodes. These can be defined as

the following pair of half planes:

$$\mathcal{D}_1(j,p) = \{(\mathbf{x},\mathbf{y}) : \mathbf{x}_j \leq p\} \text{ and } \mathcal{D}_2(j,p) = \{(\mathbf{x},\mathbf{y}) : \mathbf{x}_j > p\}, \tag{10}$$

where $\mathbf{x}_j = (x_{1,j}, ..., x_{N,j})$ is the j-th feature variable for $j = 1, .., M$. According to the CART algorithm for regression trees, the response of node b is modelled as a constant c_b for $b = 1, 2$. By minimizing the MSE, this constant can be estimated by taking the average of the log CDS spreads that are in node b and can be mathematically written as follows:

$$\hat{c}_b = \frac{1}{|\mathcal{D}_b(j,p)|} \sum_{k=1}^{|\mathcal{D}_b(j,p)|} \big(y_k | y_k \in \mathcal{D}_b(j,p)\big), \tag{11}$$

with $|\mathcal{D}_b(j,p)|$ the number of CDS quotes in $\mathcal{D}_b(j,p)$. Then, the optimal splitting variable j^* and splitting point p^* can be found by solving:

$$\min_{j,p} \left[\sum_{\mathbf{x}_i \in \mathcal{D}_1(j,p)} (y_i - \hat{c_1})^2 + \sum_{\mathbf{x}_i \in \mathcal{D}_2(j,p)} (y_i - \hat{c_2})^2 \right]. \tag{12}$$

Hence, the optimal split (j^*, p^*) detects the first two nodes of the regression tree. One can continue by applying the same splitting criterion on each of the two nodes just found. When a node is pure, i.e., it can not be split any further, it is an end-node of the tree. Then, the regression tree can be written as follows:

$$\hat{\mathtt{T}}_1(\mathbf{x}_i) = \sum_{b=1}^{B} \hat{c}_b I\{\mathbf{x}_i \in \mathcal{D}_b(j,p)\}, \tag{13}$$

where B is the number of end nodes, i.e. leaf nodes, of the regression tree $\mathtt{T}_1$, $\hat{\mathtt{T}}_1(\mathbf{x}_i)$ the prediction of the log CDS spread of entity i and $\mathcal{D}_b \subset \mathcal{D}_{train}$ for $b = 1, .., B$.

Although the tree-based methods are conceptually simple and efficient learning algorithm, Ho (1995) addresses the issue of overfitting. The problem with decision trees is that they can grow very fast due to unnecessary sub-trees. This results in that the model captures a lot of noise and that reduces the confidence of performing well on other data than the training data set.

4.3.2 Bagging Regression Trees

For unstable procedures such as decision trees algorithms, bagging is considered as a great technique to improve regression trees models, according to Breiman (1994). Bagging, also known as bootstrap aggregation, is an ensemble method. In machine learning, ensemble methods are techniques that generally improve the overall performance of a model by learning from multiple models instead of a single model. The overall goal of bagging is to reduce the variance of the model and to do so, repeated random samples are collected with replacement from the training data set. That is, when a sample is collected from the training data it does not have effect on the next sample collected. On each of the sample training data sets the

model can be trained and the predictions can be obtained. With bagging the final prediction is then found by taking the average of the predictions from the bootstrapped data. The bagging regression trees method uses regression trees as model and produces a predefined number of regression trees from the CART algorithm on bootstrapped training data. By taking the average of the predictions obtained from each regression tree, the variance can be reduced significantly due to the random element as a result of bootstrapping. Note that, our data set consists of a relatively large number op predictors characterized by 'BBB' and 'BB' rating. Also, when looking at the sector and region, there is a relatively large number of financials, European and North American quotes. Therefore, it is very likely that the collection of bagged regression trees will have these strong predictors from the training data in the top split. As a result, fairly similar trees would be obtained, and thus implies a very high correlation between the trees.

4.3.3 Random Forests Regression

After introducing the concept of CART and bagging regression trees we now introduce random forest regression. The random forest regression method was introduced in Breiman (2001) and is a slight modification of bagging regression trees. The random forest regression decorrelates the bagging regression trees by randomly choosing $m_{rf} \leq M$ splitting candidate(s) from feature variables $\mathbf{x}_1, .., \mathbf{x}_M$ with $\mathbf{x}_j \in \mathcal{D}_{train}$ for $j = 1, .., M$. Then, the CART algorithm is used to grow a tree until the predefined minimum node size or the maximum depth of the tree is reached. Let us denote $\Theta = \{\Theta_k\}_{k=1}^{K}$ as the random output with K i.i.d. elements that characterizes the random forest trees in terms of splitting variables, cut points at each nodes, and end-node values. Then, the k-th random tree can be written as follows:

$$\hat{\mathtt{T}}_k(\mathbf{x}_i, \Theta_k, \mathcal{D}_{train}) = \sum_{b=1}^{B} \hat{c}_b I\{\mathbf{x}_i \in \mathcal{D}_b^*(j, p)\}, \tag{14}$$

where $\Theta_k = \{\mathcal{D}_b^*, \hat{c}_b\}_{b=1}^{B}$, B is a random variable that follows a probability distribution P and it represents the number of end nodes of the k-th tree. The number of end nodes of the k-th tree depends on the random selected feature variables $\mathbf{x}_s \in \mathbf{x}$. Furthermore, $\mathcal{D}^* \subset \mathcal{D}_{train}$ denotes the bootstrapped data and $\mathcal{D}_b^*$ the b-th node obtained from splitting procedures such that $\mathcal{D}_b^* \subset \mathcal{D}^*$ and $\hat{c}_b$ defined as (11). Then, the prediction of the log CDS spread of an entity i, using random forest regression, is defined as follows:

$$\hat{y}_i^{rf} = \frac{1}{K} \sum_{k=1}^{K} \hat{\mathtt{T}}_k(\mathbf{x}_i, \Theta_k, \mathcal{D}_{train}), \tag{15}$$

where K is the size of the random forest in terms of number of regression trees. An example of two random trees for obtaining the proxy spread of bucket (BB, Technology, North America) can be found in Figure 5.

Since the random forest model computes the total average across all the independent CART predictions, it results in a strong accuracy. That is, Breiman (2001) showed, by using the Strong Law of Large Numbers, that random forests do not overfit when B increases.

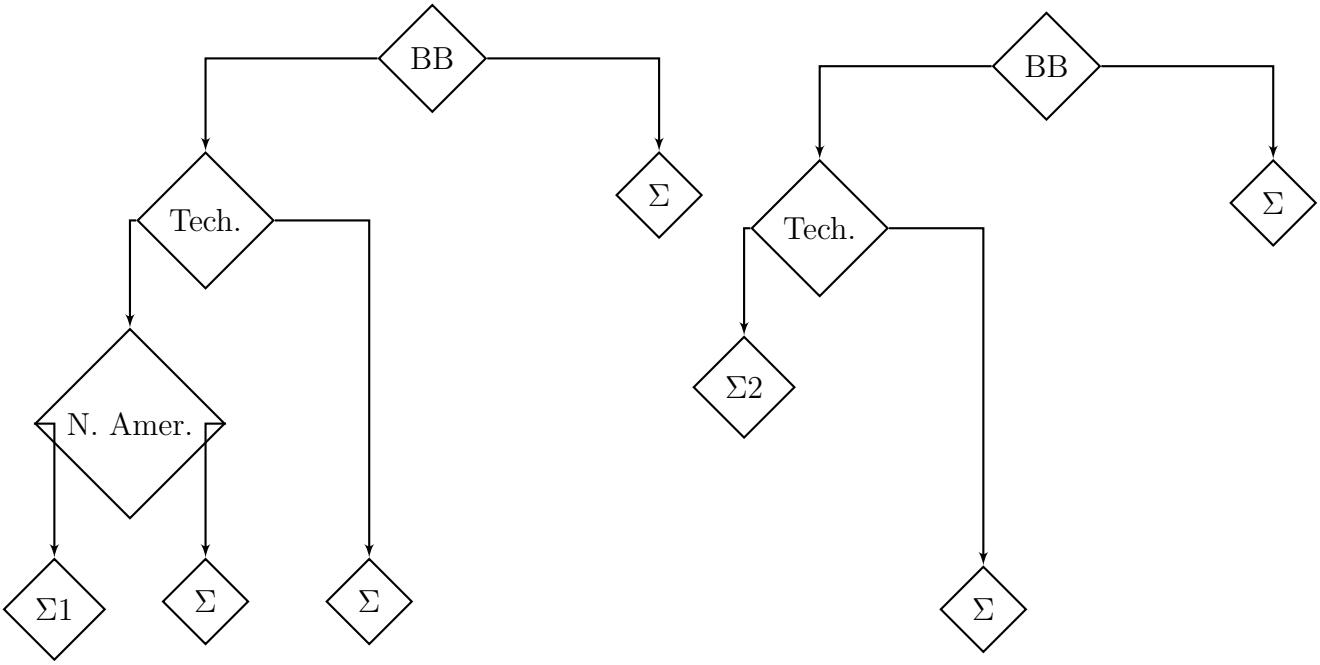

Figure 5: Two random regression trees from a random forest corresponding to the bucket (BB, Technology, North America).

Note that, Random Forests are designed such that it minimizes the overall prediction error and thus are sensitive to imbalanced data (Chen et al. (1999)). That is, the Random Forest model will tend to focus more on the prediction accuracy of the securities with features that occur often in the data set, which likely results in poor accuracy for the securities with features that do not occur very often. Furthermore, a limitation of random forest is that by choosing a very large number of trees, the model takes a relatively long time to compile.

4.4 Neural Networks

4.4.1 Artificial Neural Network (ANN)

Artificial neural networks were first developed as models to mimic the learning of the human brain (cf. Hinton (1992)). In these models, the nodes, also called artificial neurons, fired a signal by using activation functions when the total information sent to the nodes exceeded a certain threshold. The aggregation of nodes in each learning step of a neural network is called a layer. All the layers defined between the input layer and the output layer are called the hidden layers and each hidden layer in the model may perform different nonlinear transformations on the information it receives. Glorot et al. (2011) motivated the choice of rectified linear unit (ReLu) activation functions for the hidden layers. The research paper addressed the cheap computation cost of ReLu, since it only selects a subset of the neurons being active and therefore making the activation of nodes sparse and efficient. Furthermore, the mathematical computations for optimization are much easier with the ReLu activation function since the transformations applied on the activated nodes are linear.

Let us start investigating the potential of neural networks as a tool for proxying credit spreads by introducing a simple single hidden layer ANN. That is, let us again define $\mathcal{D}_{train}$ as our training set with N observations. Consider $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{train}$ with input variables $\mathbf{x}_i = (x_{i,1}, ..., x_{i,M})$, where M is the number of features in the input variable and y_i the associated continuous target value, i.e., the CDS log spread for entity i, for observation $i = 1, .., N$. Then, for Z the number of nodes in the hidden layer, the network can be defined as follows:

$$\begin{aligned} \mathbf{H}_i &= g(\mathbf{w}_1^T \mathbf{x}_i + \boldsymbol{\beta}_1), \\ S_i &= \mathbf{w}_2^T \mathbf{H}_i + \beta_2, \\ f(\mathbf{x}_i) &= S_i, \end{aligned} \tag{16}$$

where $\mathbf{w}_1 \in \mathbb{R}^{Z \text{x} M}$ the weights of the nodes in the hidden layer, $\boldsymbol{\beta}_1 \in \mathbb{R}^Z$ the bias of the nodes in the hidden layer, $g : \mathbb{R}^Z \to \mathbb{R}^Z$ with $g(a) = \max(0, a)$ the ReLu activation function for the hidden layer, $\mathbf{w}_2 \in \mathbb{R}^Z$ the weights associated to the nodes of the output layer, $\beta_2 \in \mathbb{R}$ the bias of the output layer and $f : \mathbb{R}^M \to \mathbb{R}$ the linear activation function of the output layer such that with $\hat{y}_i = f(\mathbf{x}_i)$ we obtain the log proxy spread of the i-th entity.

The cost function of ANN models for regression can be specified as the MSE i.e., $C(\hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$. Then, by calculating the gradients of the cost function with respect to all weights and biases in the network, the weights and biases can be updated. This optimization technique is called gradient descent and we apply back-propagation to obtain the gradients and be able to update the weights and biases. More specifically, to calculate the gradients after the first epoch we can apply the chain rule for each entity i. The complete calculations to obtain the gradients for the weights and biases for the hidden layer and output layer can be found in the appendix (subsection 10.1). Then, for learning cycle t, with $t \in [0, T]$ and T the predefined number of epochs, we can update the weights and biases as follows:

$$\begin{aligned} \mathbf{w}_1^{(t+1)} &= \mathbf{w}_1^{(t)} - \lambda \frac{\partial C}{\partial \mathbf{w}_1^{(t)}}, \\ \mathbf{w}_2^{(t+1)} &= \mathbf{w}_2^{(t)} - \lambda \frac{\partial C}{\partial \mathbf{w}_2^{(t)}}, \\ \boldsymbol{\beta}_1^{(t+1)} &= \boldsymbol{\beta}_1^{(t)} - \lambda \frac{\partial C}{\partial \boldsymbol{\beta}_1^{(t)}}, \\ \beta_2^{(t+1)} &= \beta_2^{(t)} - \lambda \frac{\partial C}{\partial \beta_2^{(t)}}, \end{aligned} \tag{17}$$

where $\lambda \in \mathbb{R}$ is the learning rate parameter. Then, $(\mathbf{w}_1^*, \boldsymbol{\beta}_1^*, \mathbf{w}_2^*, \beta_2^*)$ is the optimal collection of weights and biases. Note that, the weights and biases are initialized randomly by taking values from a uniform distribution with bounds [-1, 1] for the output layer and from a normal distribution with mean 0 and standard deviation 0.01 for the hidden layer. An example of the single-hidden layer ANN model can be found in Figure 6. Note that, the figure shows the input and output layer consisting of N nodes, i.e. N CDS entities, this could also be specified as single nodes at the input and output layer, i.e. $\mathbf{x}, \hat{\mathbf{y}} \in \mathbb{R}^N$, respectively.

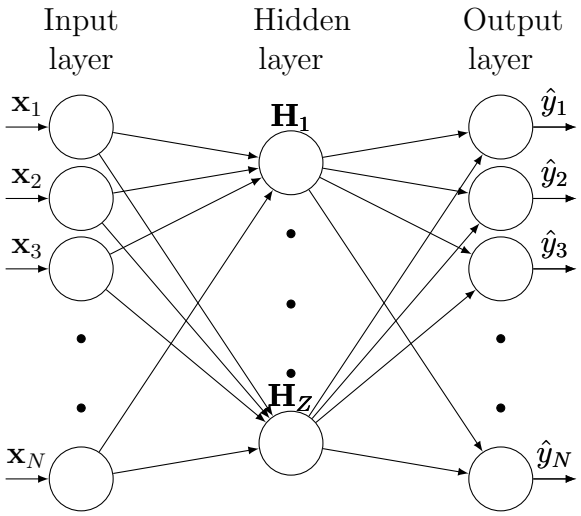

Figure 6: A single hidden layer ANN model for predicting N CDS spreads .

In order to optimize the performance of the model, calibration is necessary. That is, the number of nodes in the hidden layer, Z ,number of learning cycles, T and the the learning rate ,λ, need to be tuned. This will be further discussed in detail in chapter 6.

4.4.2 Bayesian Neural Network (BNN)

In general, ANN models require a lot of training data in order to perform well. Since the CDS data is scarce this may lead to poor prediction results. By constructing and updating measures of beliefs about the CDS log spread of a particular entity, the network may require less training data. To investigate if this is indeed the case, a single hidden layer Bayesian Neural Network is introduced next by following Blundell et al. (2015), Hastie et al. (2009) and Valentin Jospin et al. (2020). Furthermore, the model is built in `Python` by following Hasz (2019), the developer of the package `ProbFlow`, to build Bayesian models with `TensorFlow`.

Note that, for the ANN model the weights and biases are initialized randomly and updated each learning cycle. In order to transform our previous introduced single hidden layer ANN into a single hidden layer BNN, we now consider the idea of weights and biases uncertainty introduced by Blundell et al. (2015). Let us write $\boldsymbol{\theta} \in \{\bar{\mathbf{w}}_1, \bar{\mathbf{w}}_2, \bar{\boldsymbol{\beta}}_1, \bar{\beta}_2\}$, representing the parameter of the single hidden layer BNN model. Then, the parameters change after adjusting the belief about the parameters from the prior probability, $p(\boldsymbol{\theta})$, to the posterior probability, $p(\boldsymbol{\theta}|\mathcal{D}_{train})$, after learning from the data $\mathcal{D}_{train}$. As a result, the posterior probability of the parameters, given training data $\mathcal{D}_{train}$, can be written by using to the Bayes rule, as follows:

$$p(\boldsymbol{\theta}|\mathcal{D}_{train}) = \frac{p(\mathcal{D}_{train}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D}_{train})} = \frac{p(\mathcal{D}_{train}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\boldsymbol{\theta})p(\mathcal{D}_{train}|\boldsymbol{\theta})d\boldsymbol{\theta}}, \tag{18}$$

where $p(\boldsymbol{\theta})$ represents the prior probability of the parameter before training, $P(\mathcal{D}_{train}|\boldsymbol{\theta})$ the probability of training data $\mathcal{D}_{train}$ given parameters $\boldsymbol{\theta}$ and $p(\mathcal{D}_{train})$ the prior probability of training data $\mathcal{D}_{train}$. Consequently, the proxy CDS log spread $\hat{y}_i$ can be defined as a posterior distribution as follows:

$$p(\hat{y}_i|\mathbf{x}_i, \mathcal{D}_{train}) = \int p(\hat{y}_i|\mathbf{x}_i, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}_{train}) d\boldsymbol{\theta}. \tag{19}$$

In practice, it turns out that it is very difficult to obtain the distribution for the proxy spread by using integration. Therefore, finding a variational approximation to the Bayesian posterior distribution on the weights and biases is a more reasonable approach. Also, this method is more efficient than a sophisticated Markov Chain Monte Carlo simulation according to Hinton and van Camp (1993).

An approximation, $q(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ with $\boldsymbol{\alpha} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}^2\}$, of the posterior distribution, $p(\boldsymbol{\theta}|\mathcal{D}_{train})$, is a simpler variational approximation of the posterior distribution. Then, a lower bound for the prior distribution $p(\mathcal{D}_{train})$ can be specified as follows:

$$\log\, p(\mathcal{D}_{train}) \geq \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\alpha})}[\log\, p(\mathcal{D}_{train}|\boldsymbol{\theta})] - d_{KL}(q(\boldsymbol{\theta}|\boldsymbol{\alpha}), p(\boldsymbol{\theta})), \tag{20}$$

where $\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\alpha})}[\log\, p(\mathcal{D}_{train}|\boldsymbol{\theta})]$ is the expected log likelihood of the observed data with a particular parameter of the BNN following a distribution with parameters $\boldsymbol{\alpha}$. Furthermore, $d_{KL}(q(\boldsymbol{\theta}|\boldsymbol{\alpha}), p(\boldsymbol{\theta}))$ is a measure of difference between the variational distribution $q(\boldsymbol{\theta}|\boldsymbol{\alpha})$ and prior distribution $p(\boldsymbol{\theta})$. Note that, the right hand side of (Equation 20) is also known as the Evidence Lower Bound (ELBO) expression. That is, the measure of difference between our variational distribution and prior distribution for the weights and biases is minimized by maximizing the ELBO with respect to the variational distribution, $q(\boldsymbol{\theta}|\boldsymbol{\alpha})$. This can mathematically be written as follows:

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}}\; d_{KL}(q(\boldsymbol{\theta}|\boldsymbol{\alpha}), p(\boldsymbol{\theta})) - \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\alpha})}[\log\, p(\mathcal{D}_{train}|\boldsymbol{\theta})], \tag{21}$$

where $q(\boldsymbol{\theta}|\boldsymbol{\alpha}^*) = \mathcal{N}(\boldsymbol{\theta}, \mu^*, \boldsymbol{\sigma}^{2^*})$. Note that, finding the likelihood over the variational posterior is again computationally not possible. Therefore, the cost function of the BNN, $F(\mathcal{D}_{train}, \boldsymbol{\alpha}) = d_{KL}(q(\boldsymbol{\theta}|\boldsymbol{\alpha}), p(\boldsymbol{\theta})) - \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\alpha})}[\log\, p(\mathcal{D}_{train}|\boldsymbol{\theta})]$, can be approximated by taking Monte Carlo sample drawns from the variational posterior distribution such that we obtain the following approximation for the cost function:

$$F(\mathcal{D}_{train}, \boldsymbol{\alpha}) \approx \sum_{r=1}^{S} \log\, q(\boldsymbol{\theta}^{(r)}|\boldsymbol{\alpha}) - \log\, p(\boldsymbol{\theta}^{(r)}) - \log\, p(\mathcal{D}_{train}|\boldsymbol{\theta}^{(r)}), \tag{22}$$

where $\boldsymbol{\theta}^{(r)}$ is the rth random sample drawn from the variational posterior distribution $q(\boldsymbol{\theta}|\boldsymbol{\alpha})$. Then, with the approximation for the cost function, the BNN can be optimized using the tractable objective function:

$$\boldsymbol{\alpha}^* := \underset{\boldsymbol{\alpha}}{\operatorname{argmin}}\; \sum_{r=1}^{S} \log\, q(\boldsymbol{\theta}^{(r)}|\boldsymbol{\alpha}) - \log\, p(\boldsymbol{\theta}^{(r)}) - \log\, p(\mathcal{D}_{train}|\boldsymbol{\theta}^{(r)}), \tag{23}$$

That is, by using Bayes by Backprop, introduced by Blundell et al. (2015), the derivatives of the distributions are calculated and then used to update the parameters each learning cycle. More specifically, Kingma et al. (2015) showed that the collection of parameters, $\boldsymbol{\alpha}$, can be incorporated in $\boldsymbol{\theta}$ as follows:

$$\begin{aligned} \boldsymbol{\alpha} &= \{\boldsymbol{\mu}, \boldsymbol{\sigma}^2\}, \\ \boldsymbol{\epsilon} &= \mathcal{N}(\mathbf{0}, \mathbf{I}), \\ h(\boldsymbol{\epsilon}) &= \boldsymbol{\theta} = \boldsymbol{\mu} \ + \ \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \end{aligned} \tag{24}$$

where $\odot$ represents piece-wise multiplication. Then, for learning cycle $t \in [0, \bar{T}]$, the following updating scheme can be applied:

$$\begin{aligned} \Delta\boldsymbol{\mu} &= \frac{\partial h}{\partial \boldsymbol{\theta}} + \frac{\partial h}{\partial \boldsymbol{\mu}}, \\ \Delta\boldsymbol{\sigma} &= \frac{\partial h}{\partial \boldsymbol{\theta}} \cdot \frac{\boldsymbol{\epsilon}}{\boldsymbol{\sigma}} + \frac{\partial h}{\partial \boldsymbol{\sigma}}, \\ \boldsymbol{\mu}^{(t+1)} &= \boldsymbol{\mu}^{(t)} - \lambda_B \Delta\boldsymbol{\mu}^{(t)}, \\ \boldsymbol{\sigma}^{(t+1)} &= \boldsymbol{\sigma}^{(t)} - \lambda_B \Delta\boldsymbol{\sigma}^{(t)}, \end{aligned} \tag{25}$$

where, $\boldsymbol{\mu}^{(0)}$ and $\boldsymbol{\sigma}^{(0)}$ are randomly initialized from $\mathcal{N}(0, \frac{1}{\sqrt{N}})$ for each parameter $\boldsymbol{\theta}$. Then, $\boldsymbol{\alpha}^* = \{\boldsymbol{\mu}^*, \boldsymbol{\sigma}^{*^2}\}$. Note that, $p(\boldsymbol{\theta}|\mathcal{D}) := q(\boldsymbol{\theta}|\boldsymbol{\alpha}^*)$ such that $\mathbb{E}[\boldsymbol{\theta}] = \boldsymbol{\mu}^*$. As a consequence, we obtain the following single-hidden layer BNN:

$$\begin{aligned} \bar{\mathbf{H}}_i &= g(\bar{\mathbf{w}}_1^T \mathbf{x}_i + \bar{\boldsymbol{\beta}}_1), \\ \bar{S}_i &= \bar{\mathbf{w}}_2^T \bar{\mathbf{H}}_i + \bar{\beta}_2, \\ \hat{y}_i &= \mathbb{E}[\bar{S}_i], \end{aligned} \tag{26}$$

where $\hat{y}_i \sim \mathcal{N}(\bar{S}_i, \Sigma)$. Furthermore, $\lambda_B \in \mathbb{R}$ is the learning rate of the BNN, $\bar{\mathbf{w}}_1 \in \mathbb{R}^{\bar{Z} \text{x} M}$ the weights of the nodes in the hidden layer, $\bar{\boldsymbol{\beta}}_1 \in \mathbb{R}^{\bar{Z}}$ the bias of the nodes in the hidden layer, $g : \mathbb{R}^{\bar{Z}} \to \mathbb{R}^{\bar{Z}}$ with $g(a) = \max(0, a)$ the ReLu activation function used as activation function for the hidden layer, $\hat{\mathbf{w}}_2 \in \mathbb{R}^{\bar{Z}}$ the weights associated to the nodes of the output layer and $\hat{\beta}_2 \in \mathbb{R}$ the bias of the output layer of the BNN model. An illustration of the single hidden layer BNN model can be found in Figure 7.

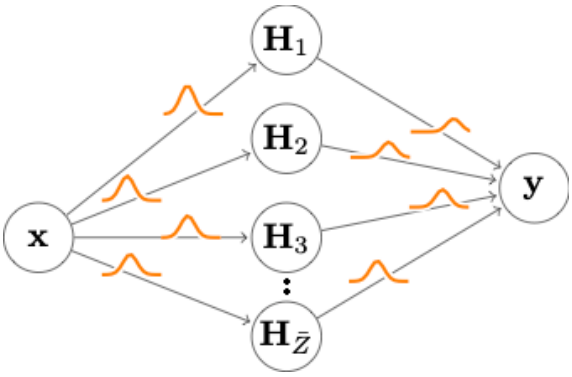

Figure 7: A single hidden layer BNN model.

In order to calibrate the BNN model, we need to tune the learning rate, λ_B, the number of nodes in the hidden layer, $\bar{Z}$, and the number of learning cycles, $\bar{T}$.

5 Model Performance Evaluation

In the previous chapter several machine learning methods for proxying the CDS spread were introduced. In order to compare the performance of the different models, the following comparison metric and evaluation procedure are used.

5.1 Root Mean Squared Error (RMSE)

The RMSE is a commonly used measure of accuracy to compare machine learning models. This metric provide us with a measure of how much our proxy spreads deviate, on average, from the actual spreads in the data set. That is, it represents the standard deviation of the residual of the sample data. A lower RMSE indicates a better accuracy. The RMSE, mathematically, is defined as follows:

$$RMSE := \sqrt{\frac{1}{N_1}\sum_{i=1}^{N_1}(y_i - \hat{y}_i)^2}, \tag{27}$$

where N_1 is the number of values of our target variable $\mathbf{y} \in \mathbb{R}^{N_1}$. Note that, here we consider the log-transformation of the CDS spreads as target variable.

5.2 Leave-One-Out Cross-Validation (LOOCV)

In order to learn more about how models perform on unseen data, the data set is split into a training set and a test set. A popular choice is following the 80%/20% splitting rule. However, since we would like to predict missing credit spreads, we would like to use all the data that is available. Therefore, a leave-one-out cross-validation procedure is used instead of splitting the data into 80% training data and 20% test data. More specifically, by using the leave-one-out cross-validation algorithm, the models are trained for each instance, using all other instances as a training set and using the selected instance as a single-instance test set. Hence, in this way we can obtain the complete data set as an out-of-sample prediction. We can use the RMSE to obtain an accuracy measure $S_{LOOCV} \in \mathbb{R}$ from leave-one-out cross-validation for the sample and out-of-sample performance as follows:

$$\begin{aligned} S_{LOOCV}^{(train)} &:= \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{N-1}\sum_{k=1}^{N-1}\left(y_{ik}^{(train)} - \hat{y}_{ik}^{(train)}\right)^2}, \\ S_{LOOCV}^{(test)} &:= \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i^{(test)} - \hat{y}_i^{(test)}\right)^2}, \end{aligned} \tag{28}$$

where N is the number of observations, $\hat{y}_{ik}^{(train)}$ is the prediction of $y_{ik}^{(train)}$, the target value of the k-th CDS instance from the i-th training data set, and $\hat{y}_i^{(test)}$ the prediction of the i-th out-of-sample CDS instance with target value $y_i^{(test)}$.

6 Hyper-Parameter Optimization

In order to optimize the performance of the machine learning methods, the hyper-parameters need to be tuned for each model. More specifically, for each model we seek for the optimal set of hyper-parameters that minimizes overfitting and maximizes the accuracy of the model from the leave-one-out cross-validation procedure. Note that, for OLS regression models, minimizing the sum of squares errors provide us with the optimal performance of the OLS regression model. Therefore, further model calibration is not required for the cross-section model. Furthermore, for the $K-$NN model we can evaluate different values for $K \in \mathbb{R}$ and obtain the K for which overfitting is minimized. In this chapter we will discuss the calibration of the models based on the 14 September 2020 data. In a similar way the hyper-parameters of the models can be obtained for other trading days.

6.1 K-Nearest Neighbors

The K-NN regression model has one hyper-parameter that need to tuned, namely the number of nearest neighbors K. In order to optimize the K-NN model, the K is chosen that minimizes overfitting or underfitting and that maximizes the overall accuracy. For the standard K-NN both cases, excluding and including the dummy variable composite depth, are considered. Figure 8 and Figure 9 show the plots of the RMSEs from leave-one-out cross-validation for $K \in [1, 10]$ for the standard K-NN model, excluding the composite depth dummy variable and including this dummy variable, respectively.

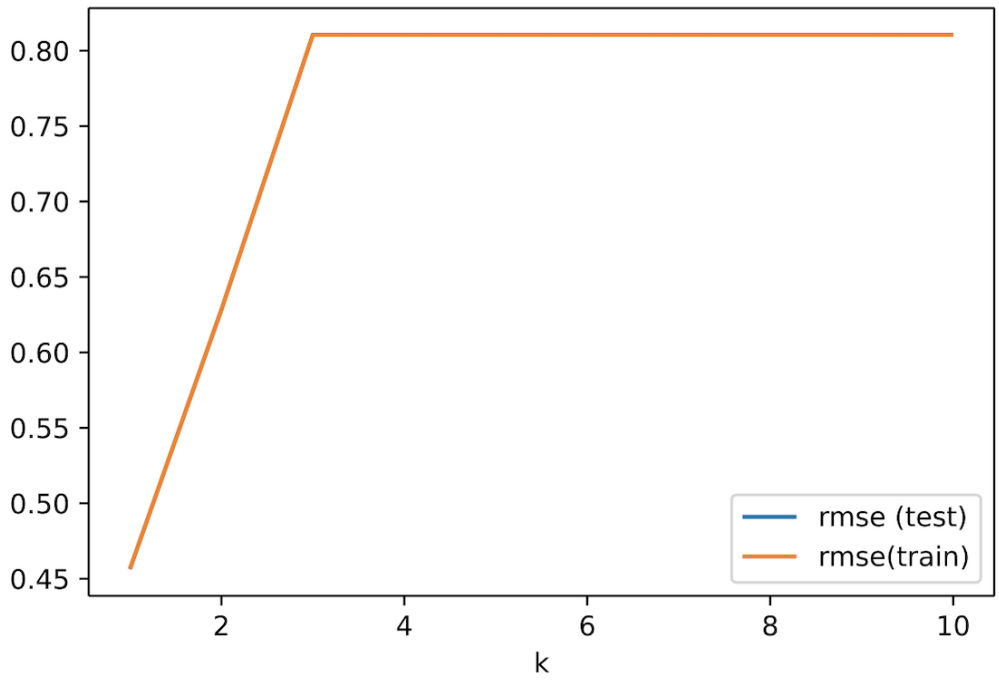

Figure 8: Standard K-NN Regression model: optimal choice of K based on 14 September 2020 data.

It is interesting to see that for both cases the model does not suffer from overfitting or underfitting for all $K \in [1, 10]$, approximately. Without the composite depth dummy variable, the optimal number of neighbors is $K = 1$ with an RMSE from training the model of 0.457 and an out of sample RMSE of 0.456. One can see that for $K > 1$ the RMSEs are consistently higher. This shows that our assumption about generating undesired noice when we consider the standard Manhattan distance for the algorithm is correct. Intuitively, one can argue that the strength of relationship between features that the quotes differ may be important in finding the nearest neighbors as discussed earlier in paragraph 4.2.2. Also, there is a very small difference between the accuracy of training and testing, but since the overall accuracy is significantly better for $K = 1$ than for other choices of K, we decide to pick $K = 1$ as the optimal hyper-parameter. When one considers the inclusion of the dummy variable composite depth, a value of the training RMSE of 0.451 and test RMSE of 0.450 is found at $K = 2$ (Figure 9).

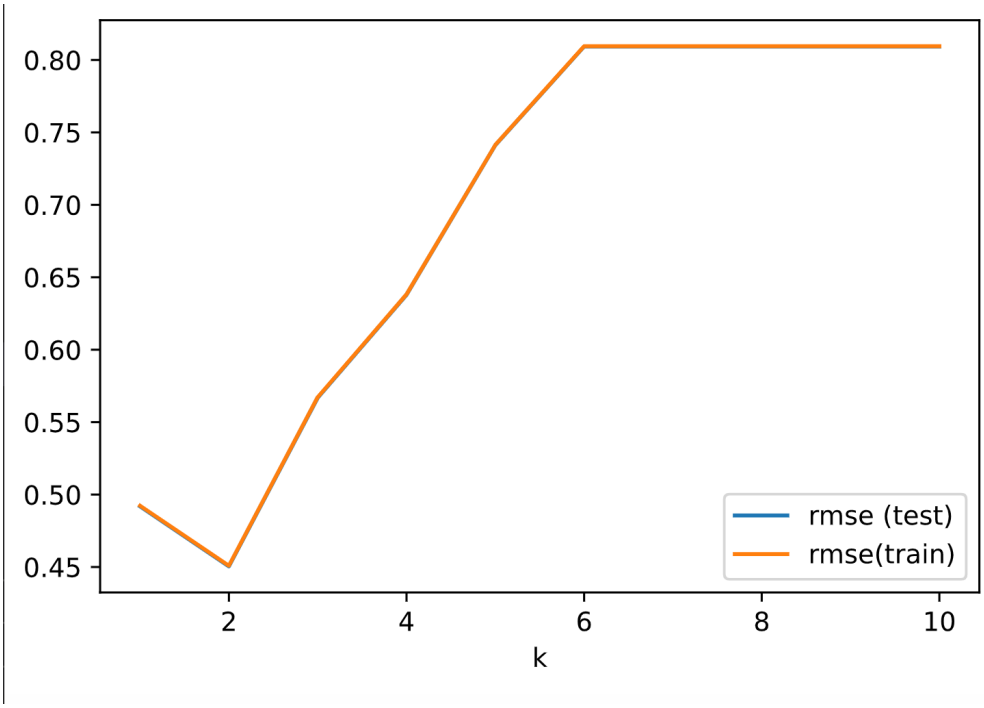

Figure 9: Standard K-NN Regression model including the Composite Depth dummy variable: optimal choice of K based on 14 September 2020 data.

This implies that, for $K = 1$ neighbors, the model is too complex, and improves when two nearest neighbors are considered. Note that, the RMSEs now gradually increases when the dummy variable composite depth is included. This shows that the dummy composite depth is able to locally take out some noise, especially for $K \leq 5$, and captures noise when many nearest neighbors are considered, i.e., $K > 5$. Note that, an improvement for the minimum of the RMSE is found by including the composite depth dummy variable. Hence, the CDS liquidity proxy improves the model in terms of explaining the spreads when considering near-

est neighbors to determine the CDS spread of a illiquid entity.

Figure 10 shows the plot of the RMSEs obtained from the leave-one-out cross-validation of the correlation distance K-NN model for $K \in [1, 10]$. Since this model is computationally expensive, only a small subset of the training data is used such that we speed up the process of obtaining the results, i.e., it now takes less than an hour to obtain the results from leave-one-out cross-validation for a particular K. However, due to the decision of shortening the training, we now see that this K-NN model suffers from overfitting. It is assumed that overfitting reduces when all the training data would be used. One can see that the out-of-sample performance of the accuracy does improve when the correlation distance is used in order to obtain the nearest neighbors. That is, a minimum RMSE on the test set of 0.385 is obtained for $K = 3$ nearest neighbors when the dummy variable composite depth is excluded. Due to limiting training an RMSE of 0.174 is obtained on the training data set. Hence, the model suffers overfitting due to our choice of opting for saving time instead of minimizing overfitting. For the purpose of validation, Figure 11 shows the level of overfitting of the correlation distance K-NN model when the dummy variable composite depth is included. It can be seen that including the dummy variable does not influence the behaviour of the model due to the design of our correlation-distance based algorithm.

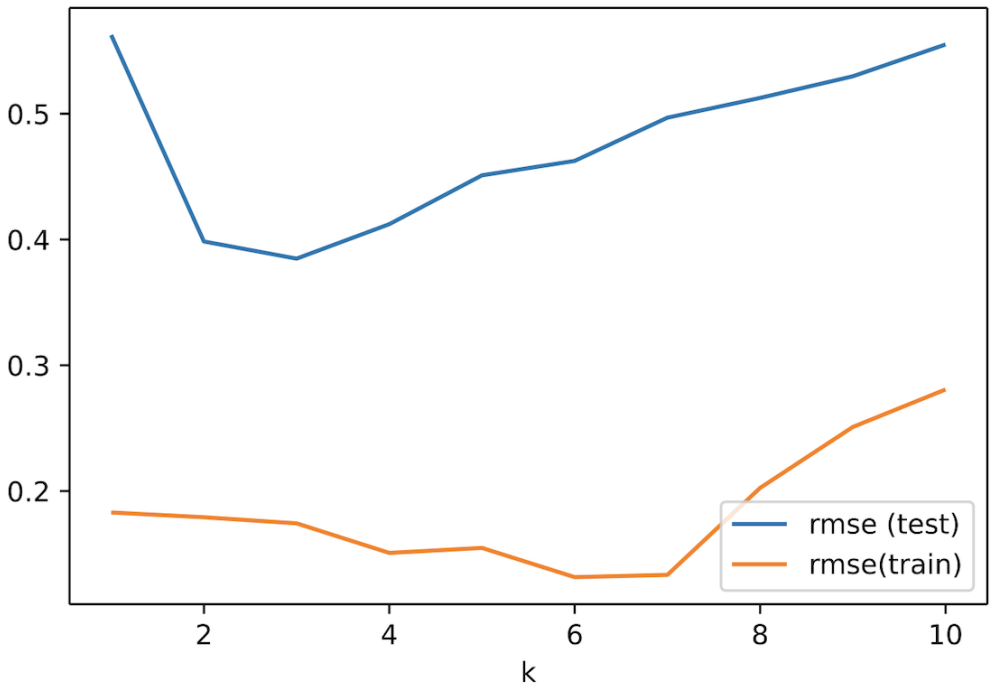

Figure 10: Correlation Distance K-NN Regression model: optimal choice of K based on 14 September 2020 data.

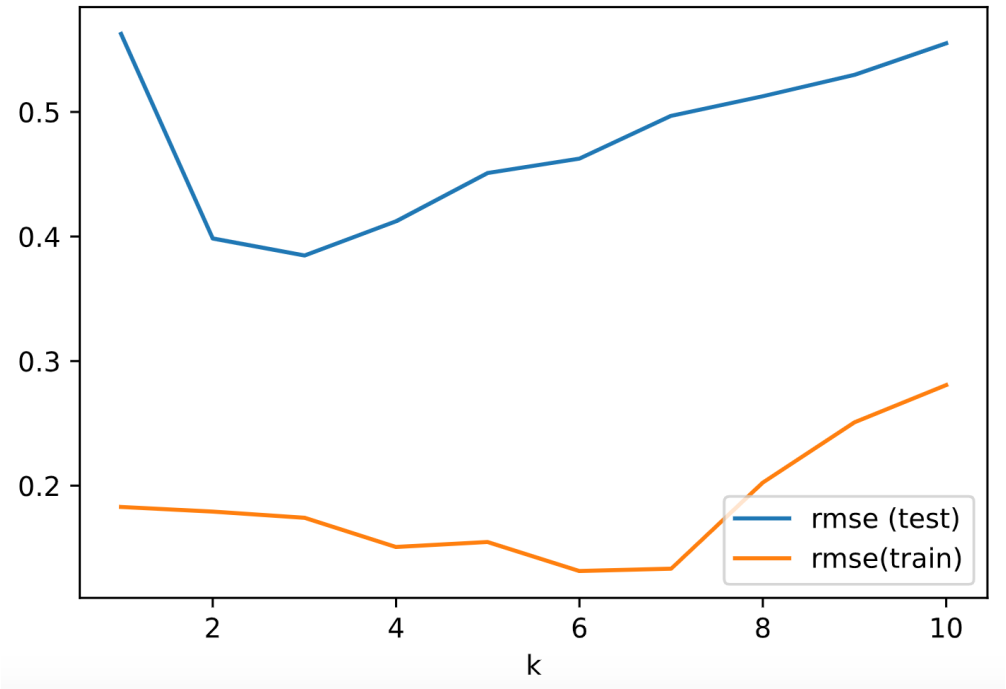

Figure 11: Correlation Distance K-NN Regression model: optimal choice of K based on 14 September 2020 data (including dummy composite depth).

6.2 Random Search

For the decision tree models and neural networks, multiple hyper-parameters need to be tuned. The random search algorithm can be used in order to obtain the set of hyper-parameters that results in an optimal performance for each model. James and Yoshua (2012) showed that random search hyper-parameter optimization is more efficient than the commonly used grid search and manual search algorithms. By using random search as our hyper-parameter optimization algorithm, we define a search space as a bounded domain of hyper-parameter values and randomly sample points in that domain, whereas with grid search we would define a search space as a grid of hyper-parameter values and evaluate every position in the grid. Therefore, random search can find combinations of hyper-parameters that would not have been guessed intuitively. In `Python`, the function `RandomSearchCV()` can be used in order to obtain the optimal hyper-parameters. In order to minimize overfitting, the function uses cross-validation. Note that, now a 10-fold cross-validation is chosen and 100 different combinations of candidate hyper-parameters are chosen each time. The reason for this is that random search is a relatively slow algorithm and by using a 10-fold cross-validation instead of a leave-one-out cross-validation it saves a substantial amount of time. Note that, by using a 10-fold cross-validation for random search the randomness in obtaining the hyper-parameters increases. This may lead to sub-optimal hyper-parameters but reduces the overall time to obtain the hyper-parameters.

6.2.1 Regression Tree

For the regression tree model, the selection procedure of selecting samples for training each tree, maximum number of levels in a tree, maximum number of features to consider at every split and the minimum number of samples required at each leave node need to be specified. Table 5 shows the hyper-parameters obtained from the random search algorithm without the dummy variable composite depth as input variable.

Selecting samples for training each tree	'best'
Max. number of levels in a tree	'None'
Max. number of features to consider at every split	'sqrt'
Min. number of samples required at each leaf node	1
Min. number of samples at each split	2

Table 5: Hyper-parameters obtained after random search for the regression tree model based on 14 September 2020 data.

Table 6 shows the hyper-parameters obtained from the random search algorithm on the data that includes this dummy variable as input variable.

Selecting samples for training each tree	'random'
Max. number of levels in a tree	60
Max. number of features to consider at every split	'sqrt'
Min. number of samples required at each leaf node	1
Min. number of samples at each split	2

Table 6: Hyper-parameters obtained after random search for the regression tree model based on 14 September 2020 data (including dummy variable composite depth).

Then, we can check the impact of the size of the tree on the problem of overfitting for both cases. That is, by taking the difference between the RMSEs on the training data and the RMSEs on the test data from the leave-one-out cross-validation procedure we obtain the level of overfitting for different tree sizes. Figure 12 shows the plot of the level of overfitting, i.e. difference between RMSE on the train data and RMSE on the out-of-sample data, with respect to the size of the tree without considering the dummy variable composite depth. Figure 13 shows the plot of the level of overfitting, i.e. difference between RMSE on the train data and RMSE on the out-of-sample data, with respect to the size of the tree when we consider the dummy variable composite depth. It is interesting to see that increasing the size of the tree leads to an increase in overfitting for tree size smaller than 20 in both figures. The level of overfitting can be reduced by setting the depth of the trees to a maximum of 10 instead of 'None' and 60 for the case when we exclude the dummy variable and include the dummy variable, respectively.

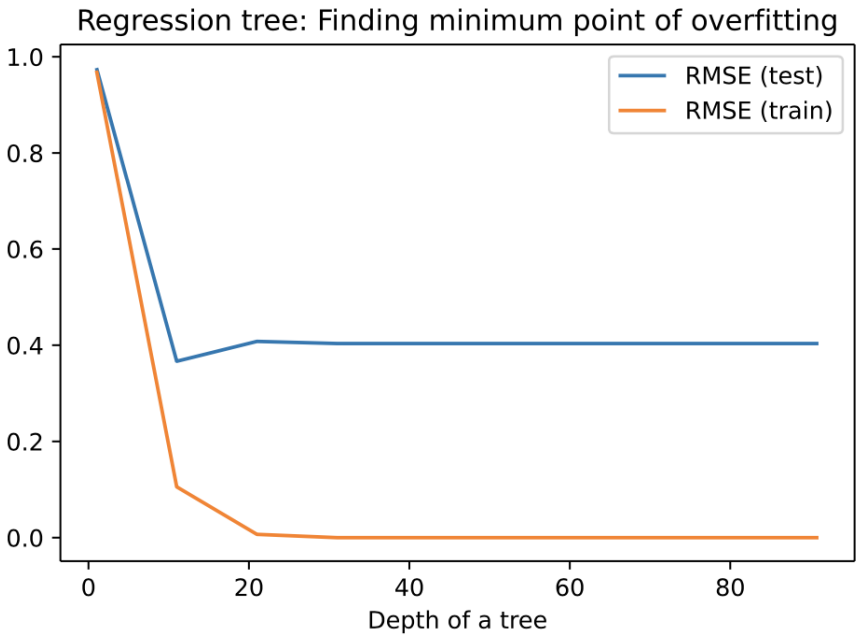

Figure 12: Regression tree model: level of overfitting based on 14 September 2020 data.

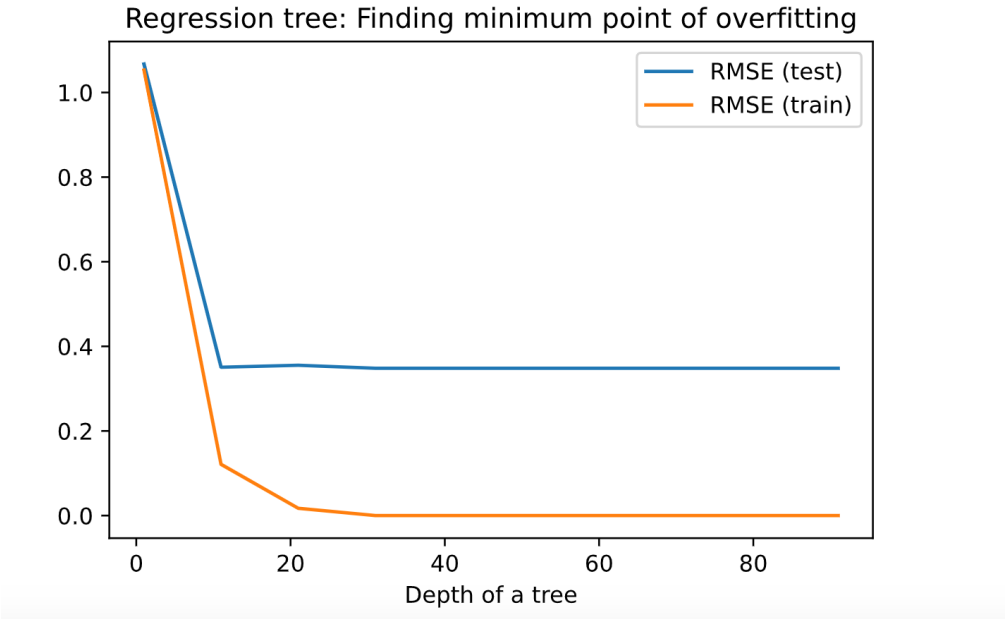

Figure 13: Regression tree model: level of overfitting based on 14 September 2020 data (including the dummy variable composite depth).

6.2.2 Random Forest

In order to optimize the performance of the random forest regression model on the CDS data set, the number of trees in the forest, the maximum number of candidates for splitting, the maximum number of splits of the trees, the minimum number of CDS quotes required for splitting, the minimum number of CDS quotes required to be at a node and whether we want the model to use bootstrap samples when building trees need to be specified. After running the `RandomSearchCV()` function, a set of combinations of hyper-parameters is found for the Random Forest model and are given in Table 7 and Table 8. Where Table 7 shows the hyper-parameters of the model for the input variables without the dummy variable composite depth and Table 8 shows the hyper-parameters of the model for the input variables with the dummy variable composite depth. Observe that we find the same set of hyper-parameters for the two cases.

Selecting samples for training each tree using bootstrap with replacement	'False'
Size of the forest	400
Max. number of levels in a tree	'None'
Max. number of features to consider at every split	'sqrt'
Min. number of samples required to split a node	2
Min. number of samples required at each leaf node	1

Table 7: Hyper-parameters obtained after random search for the random forest model based on 14 September 2020 data.

Selecting samples for training each tree using bootstrap with replacement	'False'
Size of the forest	400
Max. number of levels in a tree	'None'
Max. number of features to consider at every split	'sqrt'
Min. number of samples required to split a node	2
Min. number of samples required at each leaf node	1

Table 8: Hyper-parameters obtained after random search for the random forest model based on 14 September 2020 data (including dummy variable composite depth).

Furthermore, by using this set of combinations of hyper-parameters, the effect of increasing the size of the random forest on overfitting of the model on the training data can be investigated. Figure 14 plots the RMSEs of the training and out-of-sample predictions and Figure 15 plots the level of overfitting by taking the difference between the RMSEs on the training data and RMSEs on the test data for different sizes of the forest.

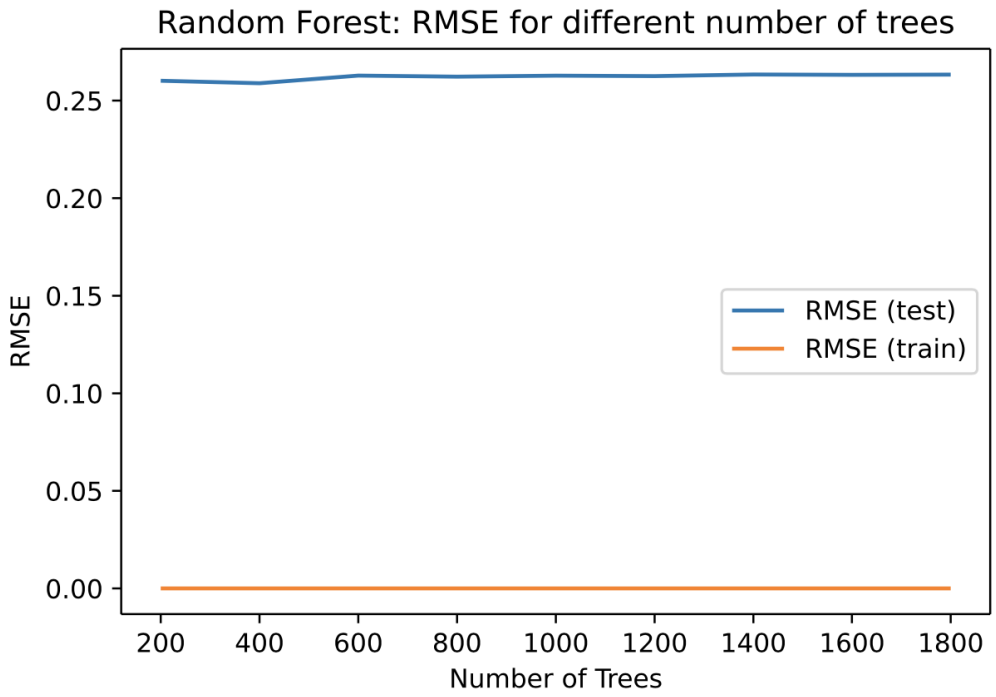

Figure 14: Random forest model: RMSE with respect to the size of the forest based on 14 September 2020 data.

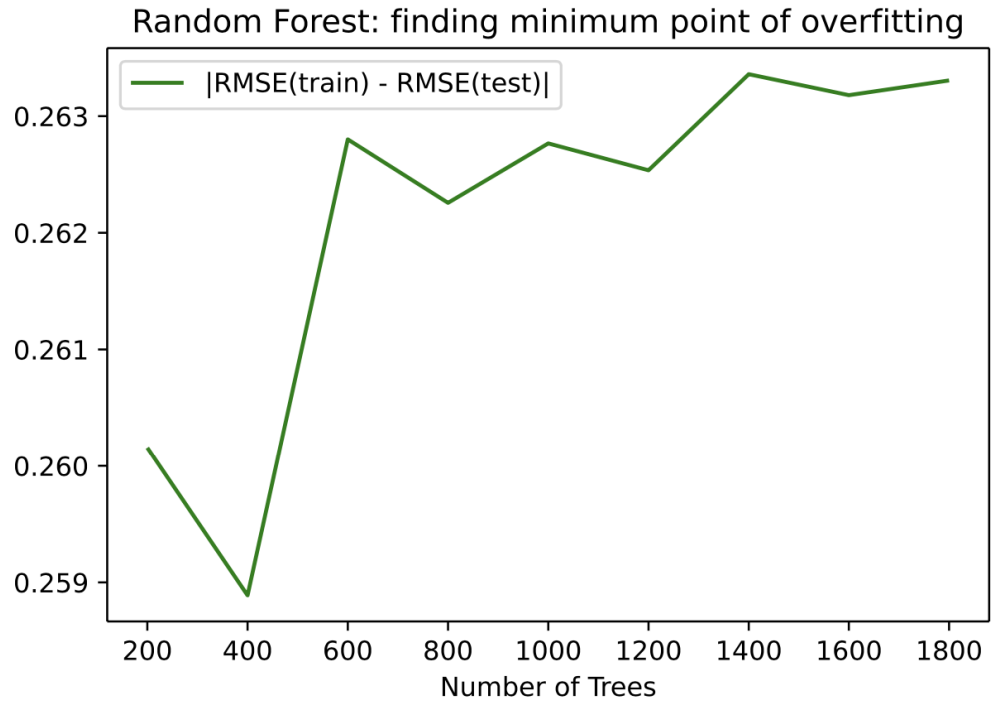

Figure 15: Random forest model: finding the minimal value of overfitting with respect to the size of the forest based on 14 September 2020 data.

Observe that a minimum level of overfitting is reached when the size of the forest is kept at 400 for the model when the dummy variable composite depth is excluded, and a minimum value of overfitting is reached when the forest contains 1400 trees for the case when the dummy variable composite depth is included as an input variable (Figure 17). Therefore, only the size of the forest is adjusted when the dummy variable is included, i.e., to 1400 number of trees. Also, it is interesting to see that the random forest model finds almost perfect prediction results on the training data, i.e. very high in-sample accuracy, whereas the out-of-sample accuracy is significantly lower for both cases (Figure 14 and Figure 16). In particular, these results bring attention to the drawbacks of the random forest method and will be further discussed in the results section of this paper.

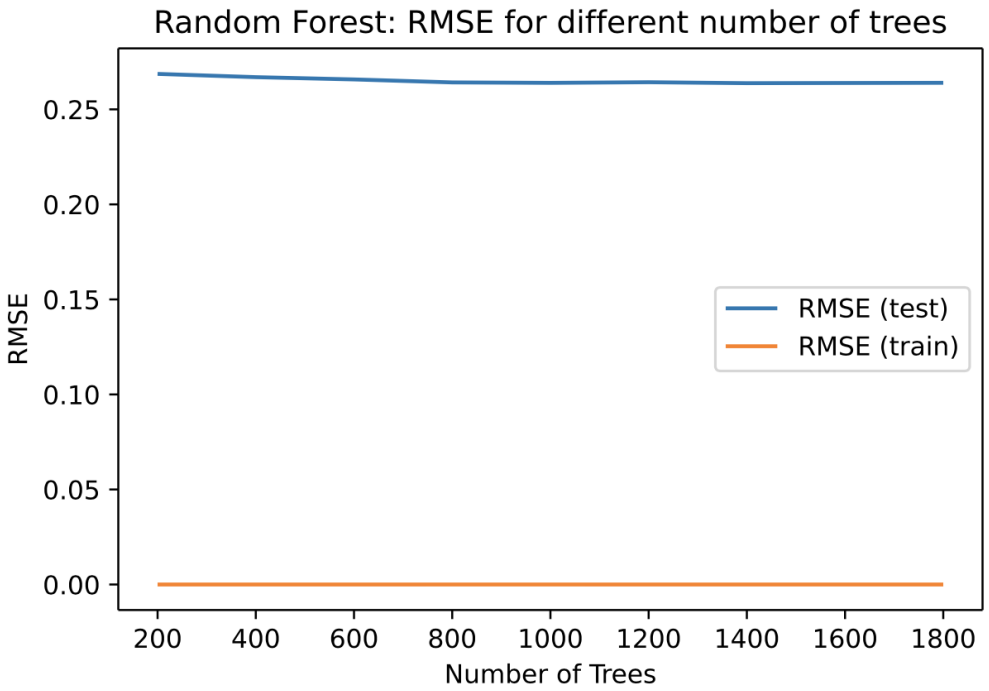

Figure 16: Random forest model: RMSE with respect to the size of the forest based on 14 September 2020 data (including dummy variable composite depth).

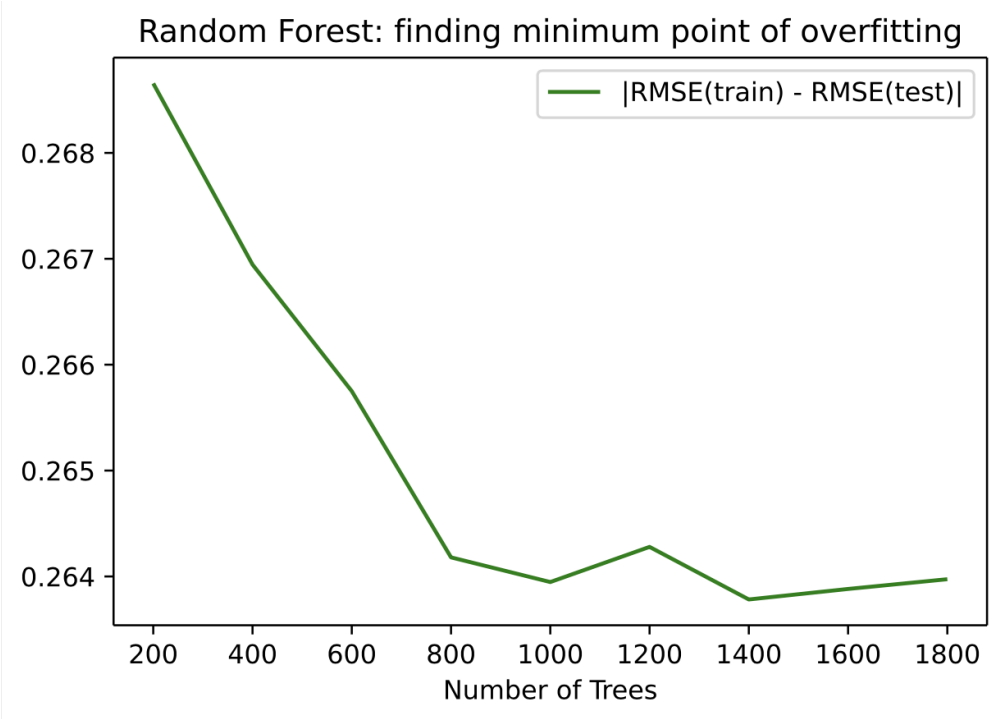

Figure 17: Random forest model: finding the minimal value of overfitting with respect to the size of the forest based on 14 September 2020 data (including dummy variable composite depth).

6.2.3 Artificial Neural Network

For the neural network, the number of nodes in the hidden layer, the learning rate and the number of learning cycles need to be tuned. In Table 9 the optimal hyper-parameters of the ANN obtained from the random search algorithm are given, and in Table 10 the optimal hyper-parameters of the ANN from random search including the dummy variable composite depth as input variable are given. It is interesting to see that, when the dummy variable composite depth is included as an input variable, the ANN model requires less nodes and less learning cycles. The reason for this may be that including the composite depth variable implies that the model is able to learn more each learning cycle because of the included liquidity proxy. Also, excluding the dummy variable composite depth results in that the model asks for more nodes, becomes more nonlinear, and this implies that the model gets more adaptive, hence learns more details.

Number of nodes in the hidden layer	100
Learning rate	0.1
Number of learning cycles	1000

Table 9: Hyper-parameters obtained after random search for the ANN model based on 14 September 2020 data.

Number of nodes in the hidden layer	90
Learning rate	0.1
Number of learning cycles	525

Table 10: Hyper-parameters obtained after random search for the ANN model based on 14 September 2020 data (including the dummy variable composite depth).

6.2.4 Bayesian Neural Network

In Table 11, the optimal hyper-parameters of the BNN obtained from the random search algorithm are given, whereas in Table 12 the optimal hyper-parameters of the BNN from random search including the dummy variable composite depth as input variable, are given. For the BNN, the dummy variable composite depth does not have impact on the choice of optimal hyper-parameters after applying the random search algorithm.

Number of nodes in the hidden layer	20
Learning rate	0.01
Number of learning cycles	100

Table 11: Hyper-parameters obtained after random search for the BNN model based on 14 September 2020 data.

Number of nodes in the hidden layer	20
Learning rate	0.01
Number of learning cycles	100

Table 12: Hyper-parameters obtained after random search for the BNN model based on 14 September 2020 data (including the dummy variable composite depth).

7 Results

In this chapter, the performances of the candidate proxy methodologies on our CDS data set of 14 September 2020 are discussed for each candidate proxy methodology. A summary of the results is provided in Table 13. Also, a performance comparison of the methodologies with respect to the trading days 12 September 2018 and 12 September 2019 is made in order to compare the performances of the candidate proxy methods for different trading days. The RMSE is used to measure the performance of accuracy and the leave-one-out cross-validation procedure is used to study the out-of-sample performance. Two cases are considered, namely including and excluding the dummy variable composite depth as an extra input variable. In order to interpret the RMSEs that we obtain, we use the statistics of the largest clusters in the CDS data set of 14 September 2020 given in Table 3. When we consider the mean and standard deviation of the North American cluster, it can be observed from this table that the coefficient of variation is 1.214. Furthermore, when we consider the mean and standard deviation of the European cluster, it can be observed that the coefficient of variation is

approximately 1.164. Also, we use an explanation technique called LIME that was introduced by Ribeiro et al. (2016). LIME locally approximates the predictions of the black-box models of our interest with a linear interpretable model. LIME trains linear interpretable models on local approximations of original data points. Note that, the LIME algorithm is kept fairly simple due to the local approximations of a linear interpretable model. In this way, we try to get a good understanding of some of the predictions of the black-box models introduced in this paper. However, although this technique may give some insights, we should be aware that the LIME technique is based on a linear interpretable model and therefore may not be powerful enough to explain all the behaviour of our black-box models.

7.1 Cross-section Method

Table 16 shows the OLS regression results of the standard cross-section model. From this table it is interesting to see that, for the feature region, only the coefficients of North America, Latin America and India are significant when a 0.1 confidence-level is chosen. Furthermore, when the feature sector is considered, the consumer goods, consumer services and telecom categories are the only significant coefficients when a 0.1 significance level is chosen. It is interesting to see that all the ratings are significant for any relevant significance level. Furthermore, the intercept is significant at any relevant confidence-level. Table 17 shows the regression results of the extended cross-section model. It can be observed that, although the intercept is significant for any relevant significance level, the coefficient of the liquidity proxy composite depth is not significant for any relevant significance level. Note that, this result depends on the trading day. That is, when we consider the trading day 12 September 2018, we find a significant coefficient of the liquidity proxy composite depth for the 0.05 significance level (Table 18).

When we consider the RMSE as accuracy metric, we find an in-sample RMSE of 0.162 and out-of-sample RMSE of 0.274 of predicting the log CDS spread when we exclude the dummy variable composite depth. This translates to that the proxy spreads are roughly a factor of 1.176 (exp(0.162)) and 1.315 (exp(0.274)) out on average from the true CDS spreads (in bps) during training and testing, respectively. Furthermore, we find an in-sample RMSE of 0.162 and out-of-sample RMSE of 0.275 when we include the dummy variable composite depth. This translates to that the proxy spreads are roughly a factor of 1.176 (exp(0.162)) and 1.317 (exp(0.275)) out on average from the true CDS spreads (in bps) during training and testing, respectively. Hence, the proxy spreads obtained from the cross-section method are approximately a factor of one coefficient of variation of the European cluster away, on average, from the true CDS spreads in bps during training. For testing, the obtained proxy spreads are approximately a factor of 1.1 coefficient of variation of the North American cluster away, on average, from the true spreads in bps.

7.2 Standard K-NN

The simplicity of K-NN makes the method an attractive candidate for proxying CDS spreads. In paragraph 6.1, the optimization of the hyper-parameters of the K-NN model were discussed. We find that, for the case that we do not consider a liquidity proxy, $K = 1$ nearest

neighbor is optimal. This implies that the algorithm seeks for the CDS quotes that have only one dissimilar feature compared to a particular CDS entity of interest. Since it is very likely that many quotes are obtained that have one dissimilar feature compared to this CDS entity of interest, a bootstrap procedure is used in order to obtain one proxy spread. This leads to an in-sample RMSE of 0.457 and an out-of-sample RMSE of 0.456. During the calibration of the model, we find that the accuracy drops very fast when we include neighbors that have less than two features in common. The reason for this may be that more exploratory variables are needed in order to make more specific clusters of CDS spreads that have a particular distance to a CDS spread of interest. Also, the strength of relationship between features that CDS entities differ may be of importance. Furthermore, when we consider the liquidity proxy composite depth, we find from calibrating the model that the model improves in terms of prediction accuracy. That is, an in-sample RMSE of 0.451 and out-of-sample RMSE of 0.450 are obtained by choosing $K = 2$ nearest neighbors. Hence, an improvement is found by using the liquidity proxy in the K-NN model and suggests that adding more exploratory variables improves creating the composition of clusters of nearest neighbors to particular CDS entities. This translates to that the proxy spreads are roughly a factor of 1.578 (exp(0.456)) out on average from the true CDS spreads (in bps) during training and testing when we exclude the dummy variable composite depth. Furthermore, the proxy spreads are roughly a factor of 1.570 (exp(0.451)) out on average from the true CDS spreads (in bps) during training and testing when we include the dummy variable composite depth. Hence, the proxy spreads obtained from the standard K-NN method are approximately a factor of 1.3 coefficient of variation of the North American cluster away, on average, from the true CDS spreads in bps when we exclude and include the dummy variable composite depth.

7.3 Correlation-Distance K-NN

In order to see if the standard K-NN can be improved in terms of prediction accuracy, the correlation-distance K-NN method was introduced. The main motivation of developing this alternative K-NN method was to reduce undesired noise. More specifically, we would like to investigate if the strength of relationship between features can be used in order to obtain more accurate distances between CDS entities. Note that, we do not consider the liquidity proxy composite depth for this model due to the design of the correlation distance function. From calibrating the model, we find that $K = 3$ nearest neighbors results in a minimization of overfitting[5] of the CDS data. This corresponds to an in-sample RMSE of 0.174 and an out-of-sample RMSE of 0.385. This translates to that the proxy spreads are roughly a factor of 1.190 (exp(0.174)) and 1.470 (exp(0.385)) out on average from the true CDS spreads

[5]To check if a model is definitely overfit, an upper bound of the out-of-sample RMSE can be consided, i.e., using the model complexity measure called the Vapnik–Chervonenkis (VC) dimension in order to obtain an upperbound for the out-of-sample RMSE of each model. According to Poczos (2015), the upper bound of the estimation error is defined as follows:

$$\epsilon \leq 4\sqrt{\frac{VC_f log(n+1) + log2}{n}}, \tag{29}$$

where VC_f denotes the VC dimension of a method f and n the number of data points. However, it is yet computational difficult to obtain the VC dimensions of some of our models. Especially, for the calibrated regression tree and random forest model it is hard to obtain the VC dimensions on our CDS dataset.

(in bps) during training and testing, respectively. Hence, the proxy spreads obtained from the correlation-distance K-NN method during training are approximately a factor of one coefficient of variation of the North American cluster away, on average, from the true CDS spreads in bps. For testing, the proxy spreads obtained from the correlation-distance K-NN method during training are approximately a factor of 1.2 coefficient of variation of the North American cluster away, on average, from the true CDS spreads in bps. The cause of difference between in-sample and out-of-sample RMSE is caused by reducing the training time and therefore implies that the model does not learn all the training data that is available. Nevertheless, observe that we find an improvement of out-of-sample prediction accuracy when we use the correlation-distance function. That is, an out-of-sample improvement of (exp(0.385) - exp(0.456))/exp(0.456) $*100\% = 6.9\%$ is found. However, due to the time consuming algorithm of obtaining correlation based distances, the out-of-sample accuracy improvement by the correlation-distance K-NN model is neglectable.

7.4 Regression tree

The next candidate proxy methodology that was introduced is the regression tree method. The reason that we would like to consider this method as a candidate proxy methodology is that this method is known for both, its simplicity and efficiency in finding predictions on complex data. After obtaining the optimal set of hyper-parameters, we find an in-sample RMSE of 0.134 and an out-of-sample RMSE of 0.355 when we exclude the dummy variable composite depth. For the case that we include the dummy variable composite depth, we find an in-sample RMSE of 0.140 and an out-of-sample RMSE of 0.382. This translates to that the proxy spreads are roughly a factor of 1.143 (exp(0.134)) and 1.426 (exp(0.355)) out on average from the true CDS spreads (in bps) during training and testing, respectively, when we exclude the dummy variable composite depth. When we include the dummy variable composite depth, the proxy spreads are roughly a factor of 1.150 (exp(0.140)) and 1.456 (exp(0.382)) out on average from the true CDS spreads (in bps) during training and testing, respectively. Hence, the proxy spreads obtained from the regression tree method during training are approximately a factor of one coefficient of variation of the European cluster away, on average, from the true CDS spreads in bps. For testing, the proxy spreads obtained from the regression tree method are approximately a factor of 1.2 coefficient of variation of the North American cluster away, on average, from the true CDS spreads in bps. Observe that there is a significant difference between the in-sample RMSE and out-of-sample RMSE for both cases, when we exclude and include the dummy variable composite depth. Hence, we find evidence that the model tends to overfit the CDS data. More specifically, including the dummy variable composite depth leads to an increase in overfitting. A possible reason for overfitting on our CDS data is that the data is imbalanced. That is, in chapter 3 we discussed the data set and found that the majority class of quotes are 'A' and 'BBB' rated, the region Asia, Europe and North America, and are financials, sovereign or industrial CDS quotes (Table 2). Therefore, due to the design of the model, finding predictions for the minority class becomes particularly difficult for this method.

7.5 Random forest

As a potential improvement for the single regression tree model, we introduced the random forest model. After model calibration, we find an in-sample RMSE of 2.673e-14 and an out-of-sample RMSE of 0.259 when we exclude the dummy variable composite depth. For the case that we include the dummy variable composite depth, we find an in-sample RMSE of 1.175e-7 and an out-of-sample RMSE of 0.264. This translates to that the proxy spreads are roughly a factor of 1 (exp(2.673e-14)) and 1.296 (exp(0.259)) out on average from the true CDS spreads (in bps) during training and testing, respectively, when we exclude the dummy variable composite depth. When we include the dummy variable composite depth, the proxy spreads are roughly a factor of 1 (exp(1.175e-7)) and 1.302 (exp(0.264)) out on average from the true CDS spreads (in bps) during training and testing, respectively. Hence, the proxy spreads obtained from the random forest method during testing are approximately a factor of 1.1 coefficient of variation of the North American cluster away, on average, from the true CDS spreads in bps. The results for this model are remarkable. Especially, the in-sample accuracy using the RMSE as our accuracy metric is a very interesting result and should be carefully interpreted. In order to get a better understanding of these predictions of the random forest model, we use the LIME technique. Figure 18 shows the LIME results of the random forest model on our CDS data of 14 September 2020 where we included the dummy variable composite depth. The LIME results show the 10 most important exploratory variables in the valuation of the prediction of the random forest model.

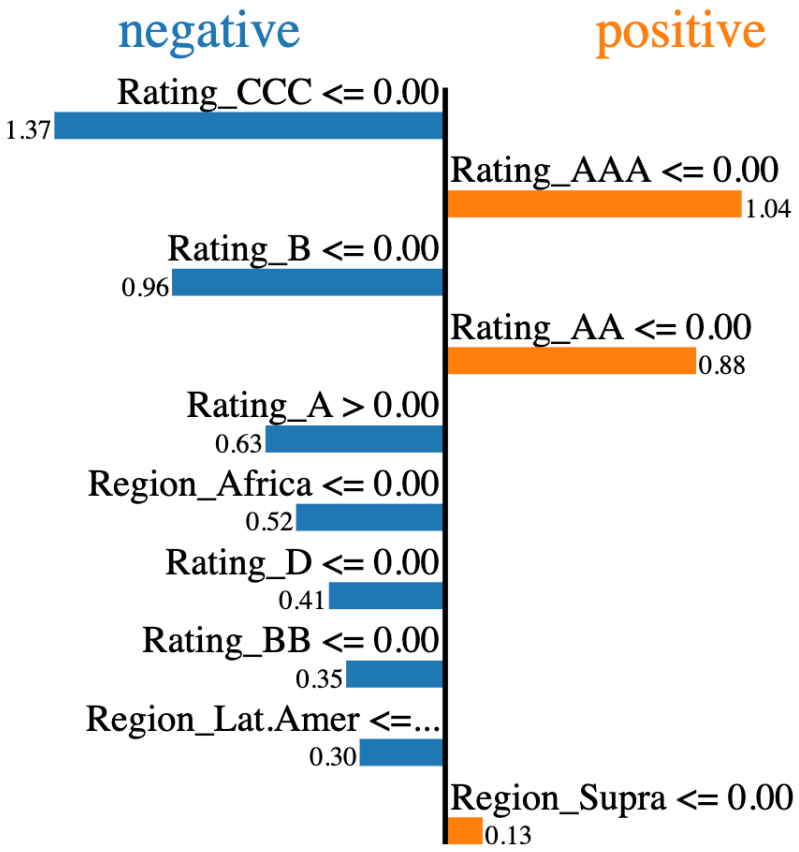

Figure 18: LIME results showing the 10 most important features in providing negative/positive valuation in prediction for the random forest model.

First of all, it is interesting to see that the feature rating is a very important explanatory variable in the valuation of the prediction of the model compared to region and sector. Especially, the rating classes 'CCC' and 'AAA' influence the valuation of the prediction substantially. That is, the table shows that for quotes that do not have a 'CCC' rating (the one-hot encoded variable corresponding to this rating equals 0) provide a negative valuation to the prediction for the random forest model. Similarly, quotes that do not have a 'AAA' rating provide a positive valuation to the prediction. From our data exploration in chapter 3 we found that both ratings, 'AAA' and 'CCC', belong to the minority class of the CDS data set of 14 September 2020. That is, there exists only 21 and 14 quotes characterized with these ratings, respectively. Similarly, Africa and Supra are in the top 10 of most important features and there are only 9 and 1 quotes characterized with these regions in the data set, respectively. Therefore, it becomes clear that the imbalanced data set causes problems for the random forest model. This clarifies why the training and testing performance differ a lot. The model is able to predict the quotes of the majority class almost perfectly, whereas problems occur when we try to predict the minority class with the random forest model.

7.6 ANN

Another nonlinear method that was introduced in this research is the single-hidden layer artificial neural network. After calibration, we find an in-sample and out-of-sample RMSE of 0.151 when we do not consider the dummy variable composite depth. When we include the dummy variable composite depth, the in-sample and out-of-sample RMSE is 0.131. This translates to that the proxy spreads are roughly a factor of 1.163 (exp(0.151)) and 1.140 (exp(0.131)) out on average from the true CDS spreads (in bps). Hence, the proxy spreads obtained from the ANN method during testing are approximately a factor of 1 and 0.98 coefficient of variation of the European cluster away, on average, from the true CDS spreads in bps, when we exclude and include the dummy variable composite depth, respectively. Note that, for both cases, excluding and including the dummy variable composite depth, we obtain that the in-sample and out-of-sample RMSE are equivalent. In order to interpret these accuracy performances, we use the LIME technique for the ANN model on our CDS data of 14 September 2020. Figure 19 shows the LIME results of the ANN model on our CDS data of 14 September 2020, where we included the dummy variable composite depth. The LIME results show the 10 most important exploratory variables in the valuation of the prediction of the ANN model.

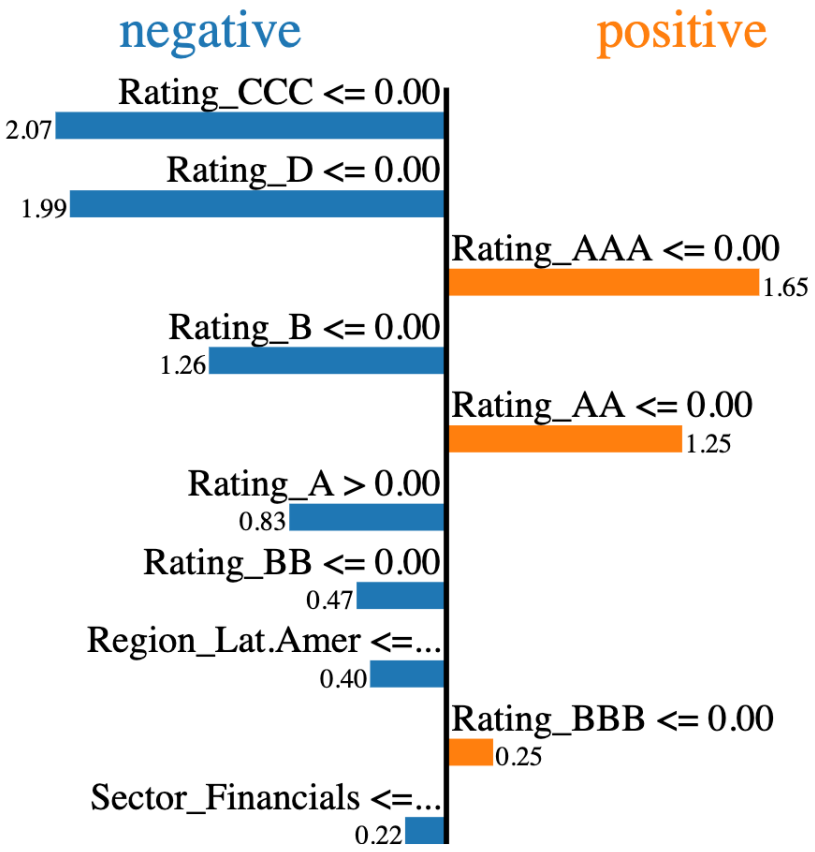

Figure 19: LIME results showing the 10 most important features in providing negative/positive valuation in prediction by the ANN.

It is interesting to see that the feature rating is, similar as for the random forest model, the most important explanatory group of variables in the valuation of the prediction of the CDS spread compared to the other features. Note that, for the ANN, the feature region may be less important in the valuation of the predictions. That is, now only the region Latin America appears in the figure. Similar as for the random forest model, the rating class 'CCC' influences the valuation of the prediction substantially. However, the difference now is that the 'D' rated quotes also provide an almost equivalent valuation in the predictions by the ANN model as the 'CCC' rated quotes. What does this tell us? The differences between the LIME results of the random forest model and ANN model show that the ANN model is learning more from the minority class and, therefore, the ANN model is able to learn more details. This also corresponds to the difference between the in-sample and out-of-sample performance of the models. That is, since the ANN model focuses more on learning smaller details, the ANN model is able to perform well on both, in-sample and out-of-sample.

7.7 BNN

The final candidate proxy methodology we introduced in this paper is the single-hidden layer bayesian neural network. After calibration, we find an in-sample and out-of-sample RMSE of 1.064 and 1.063, respectively, when we do not consider the dummy variable composite depth. When we include the dummy variable composite depth, the in-sample and out-of-sample RMSE are 1.064 and 1.065, respectively. This translates to that the proxy spreads are a factor of 2.895 (exp(1.063)) and 2.901 (exp(1.065)) out on average from the true CDS spreads (in bps). Hence, the proxy spreads obtained from the BNN method during training and testing are approximately a factor of 2.4 coefficient of variation of the North American cluster away, on average, from the true CDS spreads in bps when we exclude and include the dummy variable composite depth, respectively. Compared to the other models, the performances of the BNN model are out of order. In order to get an understanding of the poor performance of the model, we can take a look at the behaviour of the ELBO (Equation 20) over the course of the learning cycles shown in Figure 20.

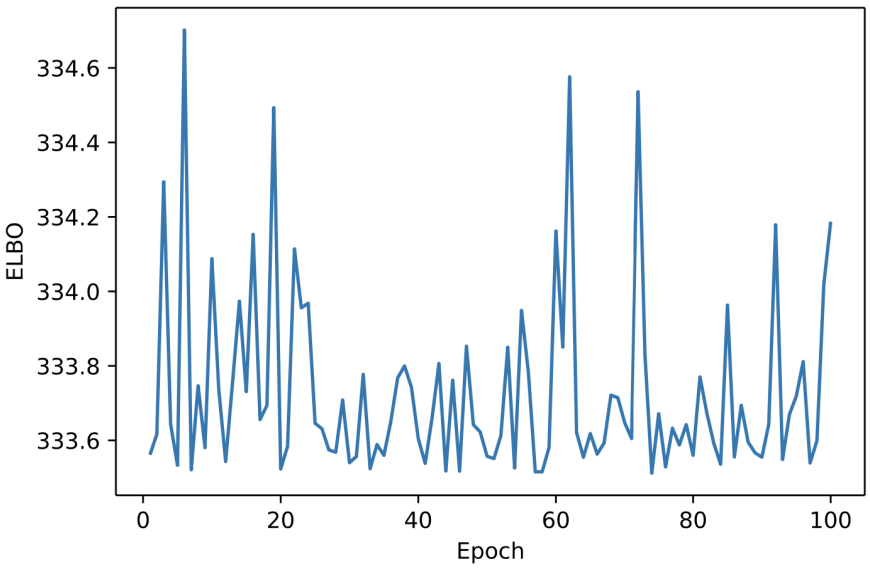

Figure 20: BNN model: Plot of the ELBO over the course of the learning cycles based on 14 September 2020 data.

From this figure it is interesting to see that the ELBO does not converge. Ideally, the ELBO would converge such that a resulting proxy for the true posterior can be obtained. However, since the ELBO does not converge, a good proxy for the true posterior can not be obtained, even when we use the log transformation of the CDS spreads. Therefore, in order to make a neural network bayesian, a good understanding of what makes a neural network learn and apply information well is important. Hence, studying the behaviour of neural networks in more detail is required in order to get an idea about what a better proxy for the true posterior should be such that the BNN can be considered as a competitive proxy candidate for our research problem.

7.8 Day-to-day comparison

In Table 13 the performance of accuracy of the machine learning models, based on the CDS data of 14 September 2020 ,are summarized for the case that the composite depth dummy variable is excluded and included (final two columns). As discussed in the previous paragraph, we found that the ANN model shows the most promising results, whereas the random forest model tends to overfit the data and the BNN performs poorly.
It is now interesting to see if this is a one day advantage or it is a general result for every day. Therefore, we also perform the candidate proxy methodologies on the CDS data of 12 September 2019 and CDS data of 12 September 2018 and summarize the performance of accuracy in Table 14 and Table 15, respectively.

Model	RMSE (Train)	RMSE (Test)	(dum.) RMSE (Train)	(dum.) RMSE (Test)
CS	0.162	0.274	0.162	0.275
K-NN standard	0.457	0.456	0.451	0.450
K-NN cor.	0.174	0.385	0.174	0.385
Regression tree	0.134	0.355	0.140	0.382
Random forest	2.673e-14	0.259	1.175e-97	0.264
ANN	0.151	0.151	0.131	0.131
BNN	1.064	1.063	1.064	1.065

Table 13: Proxy performance trading day 14 September 2020.

Model	RMSE (Train)	RMSE (Test)	(dum.) RMSE (Train)	(dum.) RMSE (Test)
CS	0.207	0.327	0.206	0.328
K-NN standard	0.481	0.481	0.476	0.475
K-NN cor.	0.321	0.432	0.321	0.432
Regression tree	0.280	0.376	0.159	0.367
Random forest	2.680e-14	0.296	2.224e-07	0.311
ANN	0.128	0.128	0.198	0.198
BNN	0.990	0.992	0.989	0.995

Table 14: Proxy performance trading day 12 September 2019.

Model	RMSE (Train)	RMSE (Test)	(dum.) RMSE (Train)	(dum.) RMSE (Test)
CS	0.179	0.292	0.177	0.294
K-NN standard	0.449	0.448	0.443	0.443
K-NN cor.	0.214	0.500	0.214	0.500
Regression tree	0.273	0.427	0.223	0.400
Random forest	2.726e-14	0.256	2.563e-07	0.418
ANN	0.200	0.200	0.135	0.135
BNN	0.941	0.948	0.941	0.949

Table 15: Proxy performance trading day 12 September 2018.

The results indicate that the performances of the models are fairly similar over the three trading days we consider. That is, the ANN outperforms the other models in terms of overfitting/underfitting and overall performance of accuracy, whereas the BNN model performs poorly. Furthermore, the regression tree and random forest overfit the three CDS data sets significantly. Also, it can be seen that, although the random forest model improves in terms of out-of-sample accuracy, the model does not improve in terms of overfitting compared to the regression tree for the three trading days. It is also interesting to see that the K-NN correlation distance model does not outperform the out-of-sample RMSE of the standard K-NN model for the trading day 12 September 2018 but outperform the out-of-sample RMSE of the standard K-NN model when we consider the other two trading days. Also, what can be

seen from the tables is that the liquidity proxy does not provide significant improvements in overall accuracy for the models in general. In particular, when we consider the ANN model, we see that the liquidity proxy does not provide a significant improvement for the trading day 12 September 2019 but does provide an improvement in accuracy for the other trading days.

7.9 Application to CVA

When we consider the formula of the CVA calculation, we use the CDS spreads in order to calculate the risk adjusted discount factor $\exp(-(r+s)T)$, where r represents the risk-free rate, s the CDS spread and T the maturity date. We found that the ANN model outperforms the benchmark method, i.e. cross-section method, and the other candidate proxy methods in terms of accuracy for all the three trading days. Let us determine what the impact of opting for the ANN model instead of the cross-section method is when we exclude and include the dummy variable composite depth to predict the CDS spreads.

From the trading day 14 September 2020, when we exclude the dummy variable composite depth, we find a difference of out-of-sample RMSE of 0.123 between the two models. This implies that using the proxy spreads obtained from the ANN model results in a ((exp(0.151) - exp(0.274)) / exp(0.274)) $*$ 100% $= -12.6\%$ change of the factor out on average to the true CDS spreads (in bps) compared to using the cross-section method based on 14 September 2020 data. Furthermore, from the CDS data of 14 September 2020, we find a difference of out-of-sample RMSE of 0.144 between the two models when we include the liquidity proxy. This implies that using the proxy spreads obtained from the ANN model results in a ((exp(0.131) - exp(0.275)) / exp(0.275)) $*$ 100% $= -13.4\%$ change of the factor out on average to the true CDS spreads (in bps) compared to using the cross-section method based on 14 September 2020 data. Hence, an improvement of 0.6% is realized when including the liquidity proxy.

From the trading day 12 September 2019, we find a difference of out-of-sample RMSE of 0.199 between the two models when we exclude the dummy variable composite depth. This implies that using the proxy spreads obtained from the ANN model results in a ((exp(0.128) - exp(0.327)) / exp(0.327))$*$100% $= -18\%$ change of the factor out on average to the true CDS spreads (in bps) compared to using the cross-section method based on 12 September 2019 data. Furthermore, from the CDS data of 12 September 2019, we find a difference of out-of-sample RMSE of 0.130 between the two models when we include the liquidity proxy. This implies that using the proxy spreads obtained from the ANN model results in a ((exp(0.198) - exp(0.328)) / exp(0.328)) $*$ 100% $= -12.2\%$ change of the factor out on average to the true CDS spreads (in bps) compared to using the cross-section method based on 12 September 2019 data. Hence, including the dummy variable does not lead to a further improvement of the performance of accuracy of the ANN model for this trading day.

Based on the data of the trading day 12 September 2018, when we exclude the dummy variable composite depth, we find a difference of out-of-sample RMSE of 0.092 between the cross-section and ANN method. This implies that using the proxy spreads obtained from the ANN model results in a ((exp(0.200) - exp(0.292))/ exp(0.292)) $*$ 100% $= -8.8\%$

change of the factor out on average to the true CDS spreads (in bps) compared to using the cross-section method. Furthermore, including the composite depth dummy variable implies that the proxy spreads obtained from the ANN model results in a ((exp(0.135) - exp(0.294))/exp(0.294)) $*$ 100% = -14.7% change of the factor out on average to the true CDS spreads (in bps) compared to using the cross-section method based on the 12 September 2018 data. Hence, an improvement of 5.9% is realized when including the liquidity proxy.

Overall, a significant improvement compared to the cross-section method is found. The cross-section method is a simple and fast method. However, the potential of the ANN model makes this method a great proxy methodology. That is, in our research we only considered a single hidden layer for the ANN method and it may be interesting to see if increasing the number of hidden layers leads to a further improvement of accuracy performance. Also, in our literature review we discussed that neural networks (the LSTM model discussed in Yuankang et al. (2019)) can perform well on time-series data, whereas the cross-section method is limited to day-to-day predictions. It would therefore be interesting to see if using historical CDS data further improves the performance accuracy of the ANN method on CDS data.

8 Conclusion

The purpose of this research was to introduce and compare different machine learning methodologies for proxying the CDS spread as an improvement to existing proxying methodologies. The benchmark method used in this paper is the cross-section methodology, introduced by Chourdakis et al. (2013a). This method is a significant improvement to the intersection method that was introduced by the European Banking Authority. However, the performance in terms of prediction accuracy of the cross-section method is not optimal. In order to improve the cross-section method, the liquidity proxy composite depth is considered. However, the results show that including composite depth as an additional explanatory variable in the cross-section model does not lead to a significant improvement in prediction performance for the trading days considered. A great improvement of accuracy is found by considering nonlinear models. That is, this paper introduced several machine learning methodologies such as K-Nearest Neighbors, a regression tree, random forest, artificial neural networks (ANN) and bayesian neural networks (BNN) and calibrated the models for each trading day specifically. Our results show that the ANN model and random forest model outperform the cross-section method on all the three trading days when the liquidity proxy is not considered. In particular, by opting for the ANN method, a -13% change of the factor out on average to the true CDS spreads (in bps) compared to the cross-section method is realized based on 14 September 2020 data. Our results also show that the performance improvement in terms of prediction accuracy, as a result of the inclusion of the liquidity proxy, is really day dependent. That is, when we include the liquidity proxy, the accuracy of the ANN model improves on only two out of the three trading days considered in our research, but still outperforms the cross-section model. More specifically, for the ANN model, the inclusion of a liquidity proxy results in a 0.6% improvement for the trading day 14 September 2020 and a 5.9% improvement for the trading day 12 September 2018, whereas the performance does not improve for the trading day 12 September 2019. Furthermore, the out-of-sample accuracy of the random forest model based on the trading day 12 September 2018 is slightly worse

than the out-of-sample accuracy of the cross-section model when the liquidity proxy is considered. Also, the random forest model tends to overfit the data on all the three trading days regardless of the liquidity proxy, whereas the ANN model is remarkably stable. According to LIME, we find that the ANN is able to learn more details, whereas imbalanced data may be the cause of overfitting for the random forest model. Hu et al. (2019) showed that the random forest method adds considerable value as a proxy methodology for the CDS spreads, whereas our results showed that the ANN outperforms the random forest model. That is, the ANN model is a stable proxy method and does not overfit the CDS data, whereas we showed that the random forest model tends to overfit the CDS data. In order to extend our research, it would be interesting to consider deep artificial neural networks as an improvement for our current ANN model. Also, due to the flexibility of the ANN, time-series data may be considered to find out if the prediction accuracy of the ANN model can be further improved by using historical data.

9 References

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *32nd International Conference on Machine Learning, ICML 2015*, 2:1613–1622.

Breeden, J. L. (2020). Survey of Machine Learning in Credit Risk. *SSRN Electronic Journal*, (May).

Breiman, L. (1994). Bagging predictors. *Department of Statistics University of California*, (2):19.

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Breiman, L., H. Friedman, J., A. Olshen, R., and J. Stone, C. (1984). Classification Algorithms and Regression Trees. In *Classification and Regression Trees*, pages 246–280. Wadsworth International Group, Belmont, CA.

Chen, C., Liaw, A., and Breiman, L. (1999). Using Random Forest to Learn Imbalanced Data. *Discovery*, pages 1–12.

Chourdakis, K., Epperlein, E., Jeannin, M., and Mcewen, J. (2013a). A cross-section across CVA. (February).

Chourdakis, K., Epperlin, E., Jeannin, M., and Mcewen, J. (2013b). A cross-section for CVA. *Risk*, pages 20–21.

Flannery, M. J., Houston, J. F., and Partnoy, F. (2010). Credit Default Swap spreads as viable substitutes for credit ratings. In *The University of Pennsylvania Law Review*, volume 158, pages 2085–2123.

Fu, H., Kong, X., and Wang, Z. (2016). Binary code reranking method with weighted hamming distance. *Multimedia Tools and Applications*, 75(3):1391–1408.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. *AISTATS*, (14).

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*, volume 27.

Hasz, B. (2019). Bayesian Neural Networks and Tensorflow 2.0.

Hinton, G. (1992). How Neural Networks Learn from Experience. *Scientific American*, 267(3):144–151.

Hinton, G. E. and van Camp, D. (1993). Keeping neural networks simple by minimizing the description length of the weights. pages 5–13.

Ho, T. K. (1995). Random Decision Forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, pages 278–282.

Hu, N., Li, J., and Meyer-Cirkel, A. (2019). Completing the Market: Generating Shadow CDS Spreads by Machine Learning. *IMF Working Papers*, 19(292).

James, B. and Yoshua, B. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(1):281–305.

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. *Advances in Neural Information Processing Systems*, 2015-Janua(Mcmc):2575–2583.

Luo, C., Wu, D., and Wu, D. (2017). A deep learning approach for credit scoring using credit default swaps. *Engineering Applications of Artificial Intelligence*, 65(October 2016):465–470.

Poczos, B. (2015). Advanced Introduction to Machine Learning, Vapnik-Chervonenkis Theory. Technical report.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why should I trust you?" Explaining the predictions of any classifier. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17-aug:1135–1144.

Sourabh, S., Hofer, M., and Kandhai, D. (2018). Liquidity risk in derivatives valuation: an improved credit proxy method. *Quantitative Finance*, 18(3):467–481.

Valentin Jospin, L., Buntine, W., Boussaid, F., Laga, H., and Bennamoun, M. (2020). Hands-on Bayesian Neural Networks - a Tutorial for Deep Learning Users. *ACM Comput. Surv.*, 1(1):1–35.

Yuankang, R. ., Xiong, ., Cai, H., Diego-Guerra, I., Lu, Y., Xu, X., and Yin, Y. (2019). Forecasting Credit Spreads: A Machine Learning Approach.

10 Appendix

10.1 A1

$$
\begin{aligned}
\frac{\partial C}{\partial \mathbf{w}_2} &= \frac{\partial C}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial \mathbf{w}_2} \\
&= -\frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)\mathbf{L}_i \\
&= -\frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)\cdot \max(0, \mathbf{w}_1^T\mathbf{x}_i + \boldsymbol{\beta}_1) \\
&= \begin{cases} 0 & \text{if } \mathbf{w}_1^T\mathbf{x}_i + \boldsymbol{\beta}_1 \leq 0 \\ -\frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)\cdot(\mathbf{w}_1^T\mathbf{x}_i + \boldsymbol{\beta}_1) & \text{if } \mathbf{w}_1^T\mathbf{x}_i + \boldsymbol{\beta}_1 > 0, \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial C}{\partial \mathbf{w}_1} &= \frac{\partial C}{\partial \mathbf{L}_i}\frac{\partial \mathbf{L}_i}{\partial \mathbf{w}_1} \\
&= \frac{\partial C}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial \mathbf{L}_i}\frac{\partial \mathbf{L}_i}{\partial \mathbf{w}_1} \\
&= -\frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)\mathbf{w}_2^T\frac{\partial}{\partial \mathbf{w}_1}\max(0, \mathbf{w}_1^T\mathbf{x}_i + \boldsymbol{\beta}_1) \\
&= \begin{cases} \mathbf{0} & \text{if } \mathbf{w}_1^T\mathbf{x}_i + \boldsymbol{\beta}_1 \leq 0 \\ -\frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)\mathbf{w}_2^T \otimes \mathbf{x}_i & \text{if } \mathbf{w}_1^T\mathbf{x}_i + \boldsymbol{\beta}_1 > 0, \end{cases}
\end{aligned} \tag{30}
$$

$$
\begin{aligned}
\frac{\partial C}{\partial \beta_2} &= \frac{\partial C}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial \beta_2} \\
&= -\frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)\cdot 1,
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial C}{\partial \boldsymbol{\beta}_1} &= \frac{\partial C}{\partial \mathbf{L}_i}\frac{\partial \mathbf{L}_i}{\partial \boldsymbol{\beta}_1} \\
&= \frac{\partial C}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial \mathbf{L}_i}\frac{\partial \mathbf{L}_i}{\partial \boldsymbol{\beta}_1} \\
&= -\frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)\frac{\partial}{\partial \boldsymbol{\beta}_1}\max(0, \mathbf{w}_1^T\mathbf{x}_i + \boldsymbol{\beta}_1) \\
&= \begin{cases} 0 & \text{if } \mathbf{w}_1^T\mathbf{x}_i + \boldsymbol{\beta}_1 \leq 0 \\ -\frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)\cdot \mathbf{1}^T & \text{if } \mathbf{w}_1^T\mathbf{x}_i + \boldsymbol{\beta}_1 > 0. \end{cases}
\end{aligned}
$$

Table 16: Regression result: standard cross-section model based on 14 September 2020 data.

Dep. Variable:	Spread	**R-squared:**	0.977
Model:	OLS	**Adj. R-squared:**	0.974
Method:	Least Squares	**F-statistic:**	304.9
Date:	Thu, 24 Dec 2020	**Prob (F-statistic):**	4.79e-150
Time:	15:50:58	**Log-Likelihood:**	93.105
No. Observations:	232	**AIC:**	-128.2
Df Residuals:	203	**BIC:**	-28.26
Df Model:	28		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	3.8150	0.047	81.812	0.000	3.723	3.907
Rating_AA	-0.4439	0.043	-10.317	0.000	-0.529	-0.359
Rating_AAA	-1.0615	0.096	-11.103	0.000	-1.250	-0.873
Rating_B	2.1417	0.041	52.849	0.000	2.062	2.222
Rating_BB	1.3060	0.038	34.680	0.000	1.232	1.380
Rating_BBB	0.5978	0.037	16.248	0.000	0.525	0.670
Rating_CCC	2.9216	0.055	53.459	0.000	2.814	3.029
Rating_D	3.0577	0.188	16.261	0.000	2.687	3.428
Sector_Basic Materials	-0.0639	0.051	-1.242	0.216	-0.165	0.038
Sector_Consumer Goods	-0.0940	0.050	-1.885	0.061	-0.192	0.004
Sector_Consumer Services	-0.0994	0.049	-2.047	0.042	-0.195	-0.004
Sector_Energy	-0.0168	0.051	-0.328	0.743	-0.118	0.084
Sector_Government	-0.0648	0.048	-1.356	0.177	-0.159	0.029
Sector_Healthcare	-0.0826	0.063	-1.312	0.191	-0.207	0.041
Sector_Industrials	-0.0480	0.051	-0.935	0.351	-0.149	0.053
Sector_Technology	-0.0443	0.056	-0.792	0.429	-0.155	0.066
Sector_Telecom	-0.1264	0.049	-2.555	0.011	-0.224	-0.029
Sector_Utilities	-0.0302	0.051	-0.596	0.552	-0.130	0.070
Region_Africa	0.0318	0.087	0.365	0.716	-0.140	0.204
Region_Asia	0.0193	0.037	0.526	0.600	-0.053	0.092
Region_Caribbean	0.0218	0.181	0.120	0.904	-0.335	0.379
Region_E.Eur	0.0810	0.061	1.321	0.188	-0.040	0.202
Region_India	0.1105	0.057	1.934	0.055	-0.002	0.223
Region_Lat.Amer	0.3275	0.070	4.668	0.000	0.189	0.466
Region_MiddleEast	0.0066	0.060	0.109	0.914	-0.113	0.126
Region_N.Amer	0.0759	0.033	2.291	0.023	0.011	0.141
Region_Oceania	-0.0193	0.047	-0.414	0.679	-0.111	0.072
Region_OffShore	0.0832	0.064	1.295	0.197	-0.044	0.210
Region_Supra	-0.1999	0.196	-1.022	0.308	-0.586	0.186

Omnibus:	2.816	**Durbin-Watson:**	2.263
Prob(Omnibus):	0.245	**Jarque-Bera (JB):**	2.444
Skew:	0.209	**Prob(JB):**	0.295
Kurtosis:	3.278	**Cond. No.**	20.6

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Table 17: Regression result: extended cross-section model based on 14 September 2020 data.

Dep. Variable:	Spread	**R-squared:**	0.977
Model:	OLS	**Adj. R-squared:**	0.973
Method:	Least Squares	**F-statistic:**	293.1
Date:	Thu, 24 Dec 2020	**Prob (F-statistic):**	7.96e-149
Time:	15:48:41	**Log-Likelihood:**	93.173
No. Observations:	232	**AIC:**	-126.3
Df Residuals:	202	**BIC:**	-22.94
Df Model:	29		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	3.8169	0.047	81.091	0.000	3.724	3.910
dummy comp depth	-0.0089	0.026	-0.343	0.732	-0.060	0.042
Rating_AA	-0.4434	0.043	-10.279	0.000	-0.529	-0.358
Rating_AAA	-1.0553	0.098	-10.822	0.000	-1.248	-0.863
Rating_B	2.1431	0.041	52.506	0.000	2.063	2.224
Rating_BB	1.3070	0.038	34.524	0.000	1.232	1.382
Rating_BBB	0.5981	0.037	16.216	0.000	0.525	0.671
Rating_CCC	2.9233	0.055	53.156	0.000	2.815	3.032
Rating_D	3.0608	0.189	16.223	0.000	2.689	3.433
Sector_Basic Materials	-0.0654	0.052	-1.264	0.208	-0.167	0.037
Sector_Consumer Goods	-0.0954	0.050	-1.902	0.059	-0.194	0.003
Sector_Consumer Services	-0.1013	0.049	-2.068	0.040	-0.198	-0.005
Sector_Energy	-0.0175	0.051	-0.341	0.733	-0.119	0.084
Sector_Government	-0.0660	0.048	-1.375	0.171	-0.161	0.029
Sector_Healthcare	-0.0807	0.063	-1.274	0.204	-0.205	0.044
Sector_Industrials	-0.0489	0.052	-0.950	0.343	-0.151	0.053
Sector_Technology	-0.0458	0.056	-0.815	0.416	-0.157	0.065
Sector_Telecom	-0.1268	0.050	-2.557	0.011	-0.225	-0.029
Sector_Utilities	-0.0309	0.051	-0.607	0.545	-0.131	0.069
Region_Africa	0.0344	0.088	0.393	0.695	-0.138	0.207
Region_Asia	0.0235	0.039	0.606	0.545	-0.053	0.100
Region_Caribbean	0.0286	0.183	0.157	0.876	-0.331	0.389
Region_E.Eur	0.0830	0.062	1.345	0.180	-0.039	0.205
Region_India	0.1162	0.060	1.950	0.053	-0.001	0.234
Region_Lat.Amer	0.3299	0.071	4.670	0.000	0.191	0.469
Region_MiddleEast	0.0112	0.062	0.180	0.857	-0.111	0.133
Region_N.Amer	0.0775	0.034	2.311	0.022	0.011	0.144
Region_Oceania	-0.0175	0.047	-0.374	0.709	-0.110	0.075
Region_OffShore	0.0886	0.066	1.336	0.183	-0.042	0.219
Region_Supra	-0.1979	0.196	-1.009	0.314	-0.585	0.189

Omnibus:	2.958	**Durbin-Watson:**	2.261
Prob(Omnibus):	0.228	**Jarque-Bera (JB):**	2.588
Skew:	0.214	**Prob(JB):**	0.274
Kurtosis:	3.291	**Cond. No.**	22.5

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Table 18: Regression result: extended cross-section model based on 12 September 2018 data.

Dep. Variable:	Spread	**R-squared:**	0.965
Model:	OLS	**Adj. R-squared:**	0.960
Method:	Least Squares	**F-statistic:**	188.2
Date:	Thu, 21 Jan 2021	**Prob (F-statistic):**	4.73e-129
Time:	13:32:40	**Log-Likelihood:**	72.321
No. Observations:	230	**AIC:**	-84.64
Df Residuals:	200	**BIC:**	18.50
Df Model:	29		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	3.8921	0.051	76.398	0.000	3.792	3.993
dummy comp depth	-0.0662	0.031	-2.119	0.035	-0.128	-0.005
Rating_AA	-0.3363	0.050	-6.670	0.000	-0.436	-0.237
Rating_AAA	-0.9197	0.079	-11.705	0.000	-1.075	-0.765
Rating_B	1.8382	0.042	43.443	0.000	1.755	1.922
Rating_BB	1.1056	0.042	26.131	0.000	1.022	1.189
Rating_BBB	0.4680	0.041	11.350	0.000	0.387	0.549
Rating_CCC	2.4322	0.056	43.820	0.000	2.323	2.542
Rating_D	3.2170	0.224	14.372	0.000	2.776	3.658
Sector_Basic Materials	-0.0477	0.058	-0.818	0.414	-0.163	0.067
Sector_Consumer Goods	-0.1150	0.056	-2.071	0.040	-0.224	-0.005
Sector_Consumer Services	-0.1334	0.055	-2.433	0.016	-0.242	-0.025
Sector_Energy	-0.0877	0.059	-1.489	0.138	-0.204	0.028
Sector_Government	-0.0964	0.050	-1.931	0.055	-0.195	0.002
Sector_Healthcare	-0.1029	0.067	-1.543	0.125	-0.234	0.029
Sector_Industrials	-0.1269	0.055	-2.290	0.023	-0.236	-0.018
Sector_Technology	-0.0776	0.062	-1.250	0.213	-0.200	0.045
Sector_Telecom	-0.1162	0.059	-1.953	0.052	-0.233	0.001
Sector_Utilities	-0.0345	0.060	-0.575	0.566	-0.153	0.084
Region_Africa	0.2513	0.085	2.944	0.004	0.083	0.420
Region_Asia	0.0989	0.041	2.430	0.016	0.019	0.179
Region_Caribbean	0.0968	0.120	0.805	0.422	-0.140	0.334
Region_E.Eur	0.1186	0.065	1.821	0.070	-0.010	0.247
Region_India	0.1801	0.074	2.427	0.016	0.034	0.326
Region_Lat.Amer	0.2843	0.072	3.967	0.000	0.143	0.426
Region_MiddleEast	0.0193	0.088	0.219	0.827	-0.154	0.192
Region_N.Amer	0.0482	0.035	1.361	0.175	-0.022	0.118
Region_Oceania	0.1091	0.055	1.996	0.047	0.001	0.217
Region_OffShore	0.1412	0.105	1.339	0.182	-0.067	0.349
Region_Supra	0.1466	0.143	1.023	0.308	-0.136	0.429

Omnibus:	0.451	**Durbin-Watson:**	1.775
Prob(Omnibus):	0.798	**Jarque-Bera (JB):**	0.563
Skew:	-0.094	**Prob(JB):**	0.755
Kurtosis:	2.848	**Cond. No.**	22.6

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.