Supervised similarity learning for corporate bonds using Random Forest proximities

Jerinsh Jeyapaulraj
jerinsh.jeyapaulraj@blackrock.com
BlackRock, Inc., New York, NY, USA.

Dhruv Desai
dhruv.desai1@blackrock.com
BlackRock, Inc., New York, NY, USA.

Peter Chu
peter.chu@blackrock.com
BlackRock, Inc.
New York, NY, USA.

Dhagash Mehta
dhagash.mehta@blackrock.com
BlackRock, Inc., New York, NY, USA.

Stefano Pasquali
stefano.pasquali@blackrock.com
BlackRock, Inc., New York, NY, USA.

Philip Sommer
philip.sommer@blackrock.com
BlackRock, Inc., New York, NY, USA.

ABSTRACT

Financial literature consists of ample research on similarity and comparison of financial assets and securities such as stocks, bonds, mutual funds, etc. However, going beyond correlations or aggregate statistics has been arduous since financial datasets are noisy, lack useful features, have missing data and often lack ground truth or annotated labels. However, though similarity extrapolated from these traditional models heuristically may work well on an aggregate level, such as risk management when looking at large portfolios, they often fail when used for portfolio construction and trading which require a local and dynamic measure of similarity on top of global measure. In this paper we propose a supervised similarity framework for corporate bonds which allows for inference based on both local and global measures. From a machine learning perspective, this paper emphasis that random forest (RF), which is usually viewed as a supervised learning algorithm, can also be used as a similarity learning (more specifically, a distance metric learning) algorithm. In addition, this framework proposes a novel metric to evaluate similarities, and analyses other metrics which further demonstrate that RF outperforms all other methods experimented with, in this work.

CCS CONCEPTS

• **Applied computing → Economics**.

KEYWORDS

Corporate Bonds, Similarity Learning, Random Forest, Proximity, Distance Metric Learning

1 INTRODUCTION

Similarity learning has found many applications across different industries ranging from facial recognition, recommendation systems for movies, songs, house price estimates, friendship recommendations on social networks, etc. [1] For financial companies, similarity learning can be used to identify similarity across/within assets, similar investors based on their demographics and transaction histories, similar funds and hedge funds, similarity amongst companies based on their transcript calls or annual reports, similarity amongst portfolio managers, health-state similarity for retirement planning, etc. [2–12].

In the present work, we focus on corporate bonds similarity. Corporate bond market is highly diverse, hundred thousand or more securities are traded in various frequencies and volumes. This gives rise to varying liquidity profile for each and often a portion of them is low on liquidity or illiquid (SEC Rule 22e-4). This limited dealer inventory, hampers the ability of a trader/portfolio manager (PMs) to engage in the market. In return, such trades could accumulate, and this could lead to higher tracking error (specifically for index products) and performance impact (for both index and active funds).

A rigorous similarity framework for corporate bonds can be leveraged by traders in the following way. When a bond which is to be traded but isn't liquid enough, the trader or PM can scan through bonds which are similar to the original security and choose one which offers a better price or a specific trading characteristic. A tree-based approach will allow us to incorporate large number of features and capture changes in market dynamic. The similarity derived from this approach will also have captured measures which can provide relative ranking based on these characteristics. Characteristics such as higher liquidity score, better bid or offer price can help the trader make informed decisions. This framework can also be leveraged to replace existing bonds with products which satisfy better attributes such as higher Environmental, Social and Governance (ESG) score.

Machine learning research to identify similarity or clusters for financial assets has primarily focused on stocks, mutual funds and hedge funds (see, e.g., [5–9, 13–17]). While the research on these topics for corporate bonds has been sparse. Ref. [18] used various clustering algorithms such as K-means, Hierarchical clustering, Self-Organizing Maps, Fuzzy-C Mean and Gaussian mixture model to identify investment grade bonds and high yield bonds based on coupon rate, yield-to-maturity, current yield, credit rating, price and the bond being callable or not. They, then evaluate the clusters using ground truth labels that indicate successful and unsuccessful term deposit contracts when telemarketing phone calls were made to clients of a Portuguese retail bank.

In Ref. [19], trade prices of corporate bonds were clustered, and it was shown that regardless of size of the trade, both buy and sell dealer trades with customers (relative to interdealer trading) lead to an increase in price clustering while dealers may use clustered prices when purchasing from and selling to institutions (i.e., may use a clustered price to insulate themselves from the risk of asymmetric information).

In Ref. [20], corporate bonds from the used dataset were first manually sorted based on their sectors. This data was then clustered using bond attributes such as industry classification, country, bid spread, current yield, yield-to-maturity, option-adjusted spread, option-adjusted duration, outstanding amount, coupon rate, excess return, maturity date, issue date, closing price, credit rating and various key rate durations. For each sector, first the dimensionality of the data was reduced using either multiple factor analysis, principal component analysis (PCA) or singular value decomposition. Then, K-means and hierarchical clustering were used to cluster the bonds in the reduced dimensional space. Using the elbow curve method optimal clusters for each sector were then identified. Finally, to compute the weight of each variable used in clustering, the author posed the problem as a supervised classification problem where the aforementioned variables were used as input features and the cluster numbers of each data-point was used as the target variable. A random forest model was then trained on this data and the weights for each feature were assigned based on feature importance from random forest model.

1.1 Shortcomings of Unsupervised Learning Methods for Similarity Computations

Most of the research on similarity learning in the financial domain has been focused on unsupervised methods such as K-means or any other clustering methods on the original datasets or in the lower-dimensional space obtained using PCA or auto-encoders or else. Here, for the given dataset without any target variable, one performs clustering of data-points such that groups of data-points that are "similar", as defined explicitly (by feeding a specific distance metric such as Euclidean, Chebyshev, Minkowski, etc., into the model, e.g., K-means) or implicitly supplied in the form of the chosen objective function. The optimal number of clusters is chosen with the help of another metric such as the elbow curve, Silhoutte score, Gap statistics, etc. Then, for a given data-point one can identify the closest ones according to this distance.

First, in a completely unsupervised formulation of a similarity problem, there is no unique metric to evaluate if the result of similarity is "better" than any other method or not. The aforementioned metrics only evaluate the algorithms with a corresponding definition of "quality of clusters". Moreover, a priori one may not know which features are important or even sufficient to define similarity nor the weights or feature importance of the individual features. In addition, here, the manually supplied distance metric may or may not be appropriate for the underlying data manifold. e.g., if the underlying data manifold is a sphere, then the Euclidean distance may not be an appropriate metric.

1.2 Supervised Similarity Learning

Ideally, the set of important features as well as importance for each feature should be learned in an algorithmic way using the available data rather than manually supplied. Such a machine learning method that learns the distance metric from the given data is called similarity learning. The main goal of a similarity learning method is to learn the similarity metric that quantifies similarity between pairs of data-points. More specifically, the similarity metric is usually defined as the inverse of a distance metric, and hence, broadly speaking, similarity learning is also known the distance metric learning (DML).

DML is usually performed in a supervised or semi-supervised fashion [1]: for the given data, one retrieves (either explicitly given, or else) labels for pairs of data-points yielding whether the two data-points are close (or, similar) or not. Then, an optimization problem is formulated to reverse engineer the distance metric that would have made the pair-wise labels possible. In other words, instead of relying on off-the-shelf distance metrics, DML aims to algorithmically construct task-specific distance metric from the given data. Such a distance metric learned from the data can then be used as a definition of similarity for the given data; can improve the performance of a distance based model such as the K-nearest neighbor (KNN); improve the performance of a distance-based clustering method such as K-means clustering; to perform semi-supervised tasks; for dimensionality reduction tasks; etc. [22–24].

Starting from some of the earliest works (e.g., [25]), one of the most common basic strategies is to start with a parametric distance metric, most commonly the Mahalanobis metric [26], and learning the numerical values of the parameters from the available data by solving an optimization problem. Some examples of Mahalanobis distance metric based DML methods are Mahalanobis Metric for Clustering [25], information theoretic metric learning [27], DML with Eigenvalue Optimization [28], etc. which learn global distance metric and may ignore the local geometry of the data manifold. Other DML methods that learn distance metric based on local neighborhoods of data-points include Large margin nearest neighbor method [29], neighborhood component analysis [30], probabilistic approaches such as [31] etc. There also exist deep learning based DML methods such as Siamese networks [32, 33]. The reader is referred to Refs. [24] for a recent review on DML methods and Ref. [34] for a recent review on deep learning based DML methods.

In the aforementioned research as well as review articles, a powerful DML method usually goes unmentioned or only mentioned in passing[2], although it was already known to Breiman himself [38], that is, the very well-known Random Forest (RF) algorithm[39]. As briefly reviewed in the next Section, this method is not only efficient for metric learning from large and potentially imbalanced datasets and missing values, it also learns a local distance metric, i.e., distance metric that depends on the position of the data-points in the space.

In the present work, we argue that the RF method is a powerful method to learn similarity of financial assets in a supervised fashion. To the best of our knowledge, a DML (though a global one) was explicitly used for the first time in Ref. [9] to learn similarity among mutual funds. Here, we argue that RF is a powerful method to learn similarity from as complex dataset as of corporate bonds. In the next Section, we briefly review the literature on similarity learning using RF, and describe its salient features compared to other DML methods. Then, we apply the RF to learn local distance metric for corporate bond data. Furthermore, we propose a novel objective

[1]Though there exist several unsupervised learning methods to learn a underlying low-dimensional manifold where distance (or any other chosen geometric properties) between the given data-points are preserved, e.g., principal component analysis, deep autoencoders, ISOMAPs, etc., strictly speaking, they are manifold learning methods [21] and not necessarily distance metric learning methods.

[2]See, e.g., Refs [35–37], etc. for other examples of applications of RF based similarity learning.

metric to evaluate DML based similarity methods and apply it in our case. We then discuss the implications of our proposed framework and conclude.

2 RANDOM FOREST AS SIMILARITY LEARNING METHOD

In this Section, we describe how RF can be viewed as a similarity learning method, the precise definition of similarity extracted from RF and the salient features of RF as a similarity learning method.

2.1 Random Forest

We first describe Decision Tree (DT) which is also a popular and powerful machine learning algorithm for both classification and regression tasks [40]. Here, based on the chosen cost function (usually mean squared error for the regression task and Gini or information entropy for the classification task). The algorithm tries different split points for each of the features to identify the most cost efficient (minimum cost) split. Feature which gives the most cost-efficient split is then the topmost level of the corresponding "decision tree". Then, for each of the branches of the split at this level, once again goes through each feature and each split to identify the next cost-efficient split. One iterates the same splitting process for the remaining features, known as greedy and recursive binary splitting. The algorithm stops when a stopping criterion is met, e.g., a predetermined number of levels (called the depth of the tree), or when minimum number of data-points falling in each branch is reached.

Finally, we obtain terminal branches corresponding to the rectangle partitions of the feature space. Continuing with the tree analogy, the final branches are called the *leaf* nodes, and the path from the top level to a leaf node is called the *decision path* for the leaf node. Then, for every data-point that falls into the same leaf node, we make the same prediction which is equal to the mean (for the regression task) or majority voting (for the classification task) of the target values for the training dataset in the leaf node.

Now, we move to describing RF [40–42] which is an ensemble learning method based on DTs. In RF, one constructs multiple DTs in the following way: first, from given training data, one creates T number of datasets using bootstrapping of the training data with replacement. Then, for each of the T datasets, one randomly selects a subset of input features and constructs a DT using the aforementioned DT algorithm. Eventually, an aggregation of all these smaller trees is used to obtain a final model, justifying the name "Random Forest". Here, again, the depth of each DT, the number of DTs, the number of features in each randomly selected subset, etc., are hyperparameters and can be tuned to improve learning.

RF evades the problem of overfitting, that an individual DT may likely suffer from, due to ensembling of DTs. Hence, RF is usually more accurate, and the generalization error for RF reduces as the number of DTs increases. The generalization error of RF also depends on the strength of the individual DTs in RF and the correlation between them. At the same time RF's are less interpretable due to the presence of multiple small DTs.

2.2 Random Forest as an Adaptive Nearest-Neighbors Method

Though a strong relationship between nearest neighbor algorithms and RF may not be immediately obvious, it was indeed shown in Ref. [43] that RF is an adaptive nearest-neighbor algorithm. In this reference, first, a generalization of KNN, called potential nearest neighbors (k-PNNs), was introduced. Then, it was shown that RF can be viewed as an adaptively weighted k-PNN method.

Intuitively, in the regression setting, for example, a leaf node in each DT corresponds to a rectangle partition in the feature space and the predicted value of the target variable for a data-point falling in this partition is the average of the target values of all the training data-points within the partition (see Figure 1a). In other words, all the training data-points present in the partition can be viewed as the nearest-neighbors of any other data-point within the partition. For a typical DT, different partitions corresponding to different leaf nodes may be of different sizes, meaning that the number of nearest-neighbors is different depending on which partition the data-point lands on. The different size (volume) of the partitions also suggests an underlying non-trivial distance metric learned by the algorithm. Eventually, for RF, one aggregates over all the partitions in all the DTs the given data-point lands on, including additional weights into the aggregation. The same intuition can be extended to the classification setting only by replacing averaging of the target values of the training-data in the partition with majority voting of the target values of the training-data.

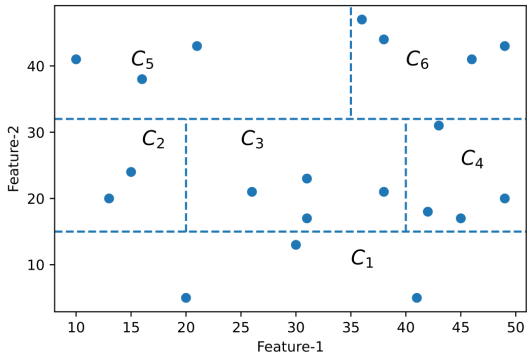
In practice, there are multiple ways to compute pair-wise similarity (and, in turn, distance), called *proximity*, using RF proposed in the literature (see, e.g., [45–47]). We make use of two traditional ways to compute proximity between each pair of data-points using RF. We will investigate other proximity definitions in the future. Below, we describe the two variants of proximity and provide explicit expressions of the same.

2.2.1 Original definition of Proximity. The earliest definition of proximity was provided in [38]: after all the T trees are grown, i.e., a random forest is trained, drop each of the data-points (from both training and OOB sets) down each of the trees. Then, if a pair of data-points fall in the same leaf node, increase their similarity score (i.e., proximity) by one for each tree. Hence, the maximum proximity between a pair of data-points can be T (i.e., the pair of data-points fall in the same leaf node in each of the T trees) and the minimum can be 0 (i.e., the pair never fall in the same leaf node in any of the T trees). Finally, we normalize the proximities for each pair of data-points by dividing them by T. Mathematically, the proximity between data-point i and j is given by

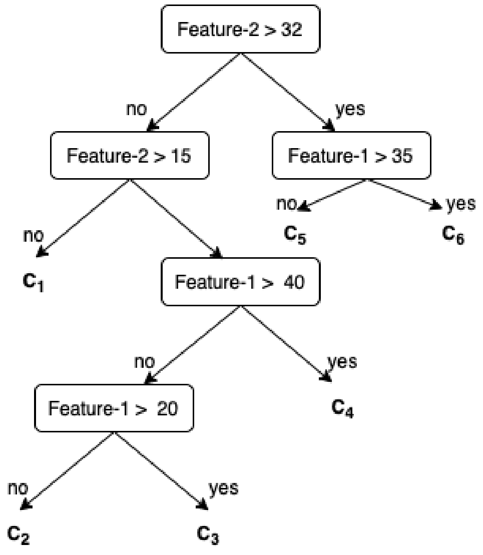
$$Prox(i,j) = \frac{1}{T}\sum_{t=1}^{T} I(j \in v_i(t)), \tag{1}$$

where T is the number of trees in the forest, $v_i(t)$ contains indices of the data-points that end up in the same leaf node as x_i in tree t, and $I(.)$ is the indicator function.

2.2.2 Out-of-Bag Proximity. The aforementioned definition of proximity uses both in-bag and out-of-bag data-points. By design, the in-bag data-points of different classes will terminate in different leaf nodes if all trees are grown such that each leaf node is pure.



(a) Decision Tree partition space



(b) Decision Tree splits

Figure 1: Example of decision tree which captures 6 disjoint regions $C_1, C_2, ...C_3$ to capture overall feature space from feature-1 and feature-2 as seen in (a). (b) represents corresponding leaf nodes in decision tree with respect to this disjoint space. For a test point which falls in C_1, its neighbours are all other data-points which belong to C_1. [44]

This will lead to over-exaggerated class separation in proximity values. One way to resolve this issue is to redefine proximity measure between data-points i and j using only those trees for which both i and j are out-of-bag samples [40, 41]. Mathematically, this proximity, called out-of-bag (OOB) proximity, is defined as:

$$Prox_{OOB}(i,j) = \frac{\sum_{t \in S_i} I(j \in O(t) \cap v_i(t))}{\sum_{t \in S_i} I(j \in O(t))}, \quad (2)$$

where $O(t)$ is the set of indices of OOB training data-points and S_i is the set of trees where observation i is OOB and v_i(t) contains indices of all data-points that end up in the same terminal node as data-point i in tree t.

2.3 Salient Features of Random Forest as a Similarity Method

Using RF as a DML method brings in all the advantages of RF over other ML methods and beyond:

- RF is a non-parametric method requiring to tune minimal number of hyperparameters;
- Unlike many of the traditional DML methods, RF requires minimal amount of data cleaning or data preprocessing. It can even handle missing values at least in theory, handles imbalanced datasets, as well as both numerical and categorical types of input data.
- RF can handle fairly large datasets making it a scalable DML technique unlike many other traditional methods;
- Since RF can handle high-dimensional datasets, there is no specific need to first reduce the dimensionality of the data using principle component analysis or some other methodology unlike many other DML methods;
- One of the most important theoretical advantage of using RF as a DML method over traditional methods is that the feature importance of individual features in computing similarity is algorithmically learned by RF for each data-point. A little thought would reveal that the previous statement yields that RF learns local distance metric (i.e., the distance metric that depends on the location in the space) as opposed to only global distance metric.

3 DATA DESCRIPTION

The universe that we tested on was a subset of the global corporate bond market, consisting mostly of US securities. For this analysis we used the cross-sectional information for the bonds as on 2021-01-01. The cross-sectional features include [48]:
Bond Duration: Measures the sensitivity of the bond's price to changes in the interest rate;
Coupon: Annual interest paid on the bond expressed as a percentage of the bond's face value;
Coupon Frequency: Indicates the number of times the coupon is paid in a given year
Country: The country where the bond is issued;
Age: The number of days since the bond's issuance;
Days to Maturity: The number of days to the bond's maturity;
Amount Issued: The total issue size of the bond offering;
Amount Outstanding: The total size of the bond that haven't yet matured or been redeemed;
Issuer: The company that issued the bond to borrow money;
Industry: The industry of the bond's issuer;
Bond Rating: A composite rating for the bonds based on the ratings given by various credit rating agencies;

Column	Type
Yield to Maturity (Target Variable)	Numerical
Coupon	Numerical
Coupon Frequency	Numerical
Duration	Numerical
Country	Category
Days to Maturity	Numerical
Age	Numerical
Industry	Category
Amount Issued	Numerical
Amount Outstanding	Numerical
Bond Rating	Category

Table 1: List of target variable and input features with their variable types.

Here, we are interested in identifying bonds which are similar in terms of the above input features with the target variable being related to liquidity. Liquidity of an asset is termed as a *multi-dimensional beast* in Ref. [48]. Leaving technicalities aside, there are four major variables that indicate corporate bond liquidity: bid-ask spreads, yield or return spreads, trading volume, and the price impact by trading (with respect to an appropriate index, for example). In the present work, we use the yield of the bond as given by the latest market price available to us as the target variable to demonstrate a first application of the proposed methodology. The range of the target variable is $[0.32, 7.82]$. In Table 1, we summarize the variable list and the data type.

3.1 Data Pre-processing

We used one-hot-encoding for categorical features. There were no missing values in training and testing data. The transformed dataset consisted of 378 categorical encoded features and 9 numerical features resulting in total of 387 features.

4 METHODOLOGY

In this Section, we describe our overall methodology to train and evaluate the RF model on the given data, compute similarities from the final RF model, and eventually evaluate the similarities.

4.1 Train-test Split and Cross-validation

We randomly shuffled the dataset and then selected 90% of the data for training and the remaining 10% data for testing. Given the size of the data, this was an optimal split size.

For the regression task at hand, we used 5-fold cross validation on the training data to perform hyperparameter optimization and extract the best performing model based on optimal parameters.

4.2 Metrics to Evaluate Random Forest Results

For the supervised regression problem at hand, we use mean squared error (MSE) and mean absolute percentage error (MAPE).

4.2.1 Mean Squared Error. The MSE of an estimator measures the average of the squared difference between the actual and predicted values, i.e., if y_i and $\hat{y}_i$ are the observed and predicted values of the target variable for the i-th data-point, then

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{3}$$

Here, smaller value of MSE means a better model.

4.2.2 Mean Absolute Percentage Error. The MAPE is the mean or average of the absolute percentage difference between the actual and predicted values [49], i.e.,

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|. \tag{4}$$

Again, smaller MAPE corresponds to a better model.

4.3 Hyperparameter Optimization

We performed hyperparameter optimization[50] for the number of trees (estimators) and the max depth until which each tree is grown in the RF. While tuning these hyperparameters we used bootstrap aggregation and kept the following parameters as constant: maximum features used to decide for splits (all features), squared error to calculate quality of split, minimum samples needed for a split being 2. We varied maximum depth from 2 to 25 with a step size of 2, and also tested the case of no limit for the depth. For number of estimators we varied them from 100 to 2000 with a step size of 250. In practice, in order to evaluate the performance of the model at these parameter values we used grid search cross-validation as implemented in Scikit-Learn.

4.4 Proximity Computations and Similarity Matrix

Using the trained model, we first extract the OOB and in-bag indices for the training data which were used to train individual tree in RF so that we can actively track OOB samples for each tree. The general idea for proximity and OOB-proximity for training data is to calculate the proximities as defined by 1 and 2, respectively, between data-points i and j.

In case of test data, we calculate the OOB proximity by taking pairs of data-points between train and test data, where the data-point from the training data is OOB. Finally, we generate an $n \times n$ matrix which captures proximity for each data-point against all others. The proximity for a specific data-point with itself will always be 1, i.e the diagonal of the $n \times n$ matrix.

4.5 Metrics to Evaluate Similarity

In general, it is always difficult to evaluate similarity, and especially to determine whether results of one similarity method is *better* than those achieved by another method, as there is no ground truth. In Ref. [45], the authors had introduced a novel way to compare their proposed definition of RF proximity to the traditional definition: they used the proximity matrices arising from different definitions of RF proximities as kernel matrices in a support vector machine by assuming that a data proximity matrix of higher quality leads to higher classification accuracy. In Ref. [47], the authors computed the training and testing errors with respect to the observed and proximity-weighted values of the target variables (say, the proximity-weighted training and testing errors). Then, they compared these two errors with the usual training and testing errors with respect to the observed and the RF predicted values of the target variables (say, RF training and testing errors), respectively, for different proximities. Finally, for whichever definition of proximity the corresponding proximity-weighted errors were closer to the RF errors was considered more accurate (i.e., geometry and accuracy preserving) proximity.

In the present paper, we propose another objective evaluation methodology to compare results from different similarity methods. We follow up from the intuition that a better similarity method should bring similar data-points closer to each other. Hence, the performance of the KNN method using the respective distance metric should yield higher performance for the optimal value of K than the distance metric learned from another method.

Here, we compare the performances of KNN with following distance metrics:

(1) Euclidean distance (d_E);
(2) Gower distance (d_G) (which is a traditional distance metric for mixed-variable type such as our dataset);
(3) Proximity distance, d_{Prox}, where
$$d_{Prox}(i,j) = 1 - Prox(i,j); \tag{5}$$
(4) OOB distance, $d_{Prox_{OOB}}$, where
$$d_{Prox_{OOB}}(i,j) = 1 - Prox_{OOB}(i,j). \tag{6}$$

To correctly take the proximity-weighted distances of the data-points into account, when using python's Sklearn [51] package for KNN, the implementation allows users to supply a pre-computed distance metric. Here, a pre-computed distance metric can be an explicitly coded distance metric function or an $n \times n$ square matrix which captures the distance from each training point to another. For any test data-point, the user needs to input an $m \times n$ matrix which captures the distance between m test data-points with all the original n training data-points which were used to train the KNN algorithm for that specific distance metric. With this implementation we used distance weighted KNN as compared to uniformly weighted.

In summary, for each of the above distance metrics, we run KNN regression for the same dataset with exactly the same train-test split, for various values of K to identify the optimal value of K. Then, we compare the RMSE and MAPE for the optimal values of K to identify the best distance metric.

5 COMPUTATIONAL DETAILS AND RESULTS

In this Section, we provide computational details, and present the results to summarize our experiments as well as their interpretations. We present results of the training of RF to demonstrate how well the RF was able to reproduce the training and test ground-truth labels. Then, we provide results from the similarity computations using the trained RF.

5.1 Computational Details

For this research we used 32-core 32 GB RAM virtual instance on a cloud platform. RF and KNN implementations from sklearn library in python scaled well when used with high number of processors and high memory. Calculating similarity matrix took 10 mins. Storing and calculating proximity matrix requires both efficient computation and large memory because we must take a pass at all leaf nodes of each tree present in the RF for each of the $\binom{n}{2}$ pairs. To the best of our knowledge there are no standard packages in python which provide proximity and OOB proximity calculations from a tree-based method, and hence we created our own which we plan to release in the future.

5.2 Supervised Learning using Random Forest

RF was trained on the training split with cross-validation as described in Section 4. To evaluate the model performance we use RMSE and MAPE as our metrics.

5.2.1 Hyperparameter Optimization and Metrics. Figures 2 shows RMSE as a function of maximum depth as a hyperparameter, for training and testing splits. Based on these plots, we chose max depth 10 as the optimal parameter to trade-off between bias and variance.

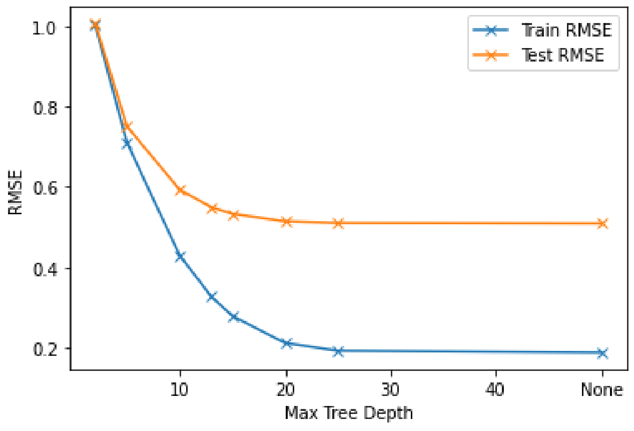


Figure 2: Random Forest Hyperparameter (estimators=1000)

In Table 2, we record the training and testing values of both the metrics. In summary, the trained RF model learned the ground truth function from the data with fairly high accuracy as measured by both the metrics. The high test accuracy also means that the available input features are able to predict the target variable reasonably well, in turn assuring that these input features are indeed sufficient enough for further similarity computations for the specific problem as formulated. Moreover, RF has also captured the (local) importance of each variable for individual predictions which will be implicitly fed into the proximity based similarity computations.

Dataset	RMSE	MAPE
Train	0.44	0.20
Test	0.61	0.24

Table 2: RMSE and MAPE for the trained RF model on the train and test datasets.

5.3 Proximity Similarity and Evaluation of Similarity

Figures 3 and 4 shows train and test RMSE and MAPE, respectively, for KNN regression with respect to all four distance metrics (d_E, d_G, d_{Prox} and $d_{Prox_{OOB}}$) for a range of values for K. For comparison, we also plot the train and test RMSEs and MAPEs for the base RF model which appear as constant lines in the figure as they are independent of K. Here, the test RMSE and MAPE for d_{Prox} and $d_{Prox_{OOB}}$ are always lower than d_E and d_G yielding that, on an average, the pairs of data points in the test dataset are correctly identified as 'nearest-neighbors' as per the latter two distance metrics.

We stress that ideally the difference between the RF train (test) RMSE (MAPE) and KNN train (test) RMSE (MAPE) with either d_{Prox} or $d_{Prox_{OOB}}$ should be the same up to statistical fluctuations. The difference seen in these figures suggest that the definitions of proximities do not capture every detail the RF has learned. Recently, Ref. [47] has proposed a so-called Geometry and Accuracy Preserving proximity definition for RF which, in theory, should close the above-mentioned gap. We plan to explore this and other definitions existing in the literature in the future.

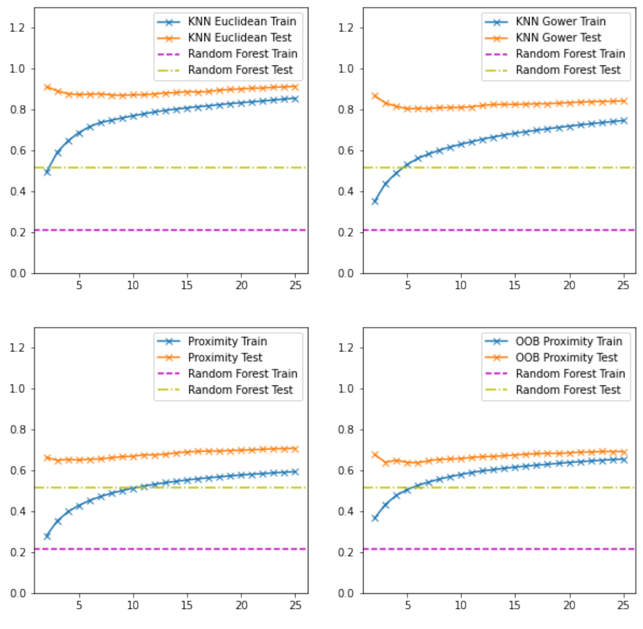


Figure 3: KNN RMSE error compared to Random Forest

6 DISCUSSION AND CONCLUSION

Asset similarity has numerous applications in finance in general and in investment processes. For corporate bonds, which are traded less frequency than many other assets such as stocks, similarity computation can be of great importance in reducing tracking error, transaction costs, as well as to replace bonds with specific themes such as the ones having higher ESG scores.

Most of the research literature on this topic though has been focused on unsupervised clustering which has several drawbacks. The unsupervised methods may not provide information on whether we

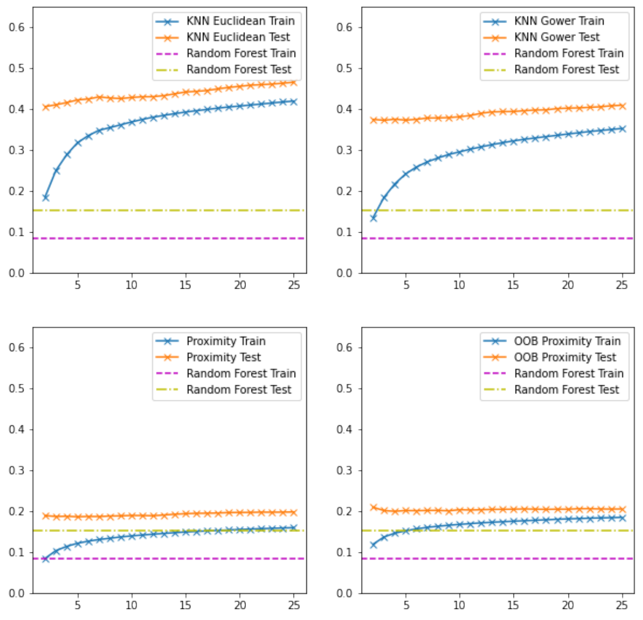


Figure 4: KNN MAPE error compared to Random Forest

have sufficient input features to compute the similarity accurately, nor do they directly yield feature importance or provide objective metrics to evaluate the results. In the present work, we started by arguing that supervised similarity, which mostly is focused on distance metric learning (DML) from the given data, is a more appropriate way to compute similarity among assets. Then, we used Random Forest (RF) based DML which is in turn based on proximity computation. In a nutshell, after a RF is trained on the training dataset, one can compute similarity between two data-points by counting the number of times the pair falls in the same leaf node out of all the trees in the forest, called RF proximity. We used variables listed in 1 as input features and the target variables to train RF for corporate bonds and used two versions of proximities to compute similarities among corporate bonds.

To evaluate similarities computed using RF proximities and using traditional distance metrics (such as Euclidean and Gower), we proposed a novel KNN-based metric which takes in model specific distance metric and predicts the target variable based on KNN with respect to the corresponding distance metric. Compared with the traditional distances, the distance metrics based on RF proximities perform better.

In the future, we will investigate effects of trading corporate bonds using the similarity learning method investigated in the present work on performance of actively managed funds as well as on tracking error of index funds.

There also exist other definitions of RF proximities (see Ref. [47] for a recent review on different proximity definitions) which we plan to investigate to identify the most appropriate proximity definition for corporate bonds. In addition, we also plan to use other tree-based methods such as extreme trees and Gradient-boosted machines to learn and extract similarity in a supervised fashion.

ACKNOWLEDGEMENT

The views expressed here are those of the authors alone and not of BlackRock, Inc.

REFERENCES

[1] Charu C Aggarwal. *Recommender systems*. Springer, 2016.

[2] Martin Lamby and Daniel Isemann. Classifying companies by industry using word embeddings. In *International Conference on Applications of Natural Language to Information Systems*, pages 377–388. Springer, 2018.

[3] Masanori Hirano, Hiroki Sakaji, Shoko Kimura, Kiyoshi Izumi, Hiroyasu Matsushima, Shintaro Nagao, and Atsuo Kato. Related stocks selection with data collaboration using text mining. *Information*, 10(3):102, 2019.

[4] Tomoki Ito, Jose Camacho-Collados, Hiroki Sakaji, and Steven Schockaert. Learning company embeddings from annual reports for fine-grained industry characterization. In *Proceedings of the Second Workshop on Financial Technology and Natural Language Processing*, pages 27–33, 2020.

[5] Achla Marathe and Hany A Shawky. Categorizing mutual funds using clusters. *Advances in Quantitative analysis of Finance and Accounting*, 7(1):199–204, 1999.

[6] John A Haslem and Carl A Scheraga. Morningstar's classification of large-cap mutual funds. *The Journal of Investing*, 10(1):79–89, 2001.

[7] Dhagash Mehta, Dhruv Desai, and Jithin Pradeep. Machine learning fund categorizations. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8, 2020.

[8] Vipul Satone, Dhruv Desai, and Dhagash Mehta. Fund2vec: mutual funds similarity using graph learning. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–8, 2021.

[9] Dhruv Desai and Dhagash Mehta. On robustness of mutual funds categorization and distance metric learning. *The Journal of Financial Data Science*, 3(4):130–150, 2021.

[10] John RJ Thompson, Longlong Feng, R Mark Reesor, and Chuck Grace. Know your clients' behaviours: a cluster analysis of financial transactions. *Journal of Risk and Financial Management*, 14(2):50, 2021.

[11] Cynthia Pagliaro, Dhagash Mehta, Han-Tai Shiao, Shaofei Wang, and Luwei Xiong. Investor behavior modeling by analyzing financial advisor notes: a machine learning perspective. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–8, 2021.

[12] Fu Tan and Dhagash Mehta. Health state risk categorization: A machine learning clustering approach using health and retirement study data. *The Journal of Financial Data Science*, 4(2):139–167, 2022.

[13] Nandita Das et al. hedge fund classification using k-means clustering method. In *9th International Conference on Computing in Economics and Finance*, pages 11–13, 2003.

[14] Maria-Augusta Miceli and Gabriele Susinno. Ultrametricity in fund of funds diversification. *Physica A: Statistical Mechanics and its Applications*, 344(1-2):95–99, 2004.

[15] Ramin Baghai-Wadji, Rami El-Berry, Stefan Klocker, and Markus Schwaiger. The consistency of self-declared hedge fund styles—a return-based analysis with self-organizing maps. *Financial Stability Report*, 9:64–76, 2005.

[16] Rajna Gibson and Sébastien Gyger. The style consistency of hedge funds. *European Financial Management*, 13(2):287–308, 2007.

[17] Hany A Shawky and Achla Marathe. Stylistic differences across hedge funds as revealed by historical monthly returns. *Technology and Investment*, 1(01):26, 2010.

[18] Utkarsha Bagde and Priyanka Tripathi. A comprehensive analysis of traditional clustering algorithms on corporate bond data. In *2018 Second International Conference on Computing Methodologies and Communication (ICCMC)*, pages 281–286. IEEE, 2018.

[19] Brittany Cole, Michael A Goldstein, Shane M Moser, and Robert A Van Ness. Trade price clustering in the corporate bond market. *China Finance Review International*, (ahead-of-print), 2022.

[20] Boyu Wu. *Similar corporate bonds selection by clustering algorithms to increase investment efficiency.* 2020.

[21] Alan Julian Izenman. Introduction to manifold learning. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(5):439–446, 2012.

[22] Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2(2):4, 2006.

[23] Brian Kulis et al. Metric learning: A survey. *Foundations and trends in machine learning*, 5(4):287–364, 2012.

[24] Juan-Luis Suárez, Salvador García, and Francisco Herrera. A tutorial on distance metric learning: Mathematical foundations, algorithms and software. *CoRR*, abs/1812.05944, 2018.

[25] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side-information. NIPS'02, page 521–528, Cambridge, MA, USA, 2002. MIT Press.

[26] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. National Institute of Science of India, 1936.

[27] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216, 2007.

[28] Yiming Ying and Peng Li. Distance metric learning with eigenvalue optimization. *The Journal of Machine Learning Research*, 13(1):1–26, 2012.

[29] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

[30] Jacob Goldberger, Geoffrey E Hinton, Sam Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. *Advances in neural information processing systems*, 17, 2004.

[31] Liu Yang, Rong Jin, Rahul Sukthankar, and Yi Liu. An efficient algorithm for local distance metric learning. In *AAAI*, volume 2, pages 543–548, 2006.

[32] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6, 1993.

[33] Davide Chicco. Siamese neural networks: An overview. *Artificial Neural Networks*, pages 73–94, 2021.

[34] Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

[35] Herbert Pang, Aiping Lin, Matthew Holford, Bradley E Enerson, Bin Lu, Michael P Lawton, Eugenia Floyd, and Hongyu Zhao. Pathway analysis using random forests classification and regression. *Bioinformatics*, 22(16):2028–2036, 2006.

[36] Paulo Angelo Alves Resende and André Costa Drummond. A survey of random forest based methods for intrusion detection systems. *ACM Computing Surveys (CSUR)*, 51(3):1–36, 2018.

[37] Peng Zhao, Xiaogang Su, Tingting Ge, and Juanjuan Fan. Propensity score and proximity matching using random forest. *Contemporary clinical trials*, 47:85–92, 2016.

[38] L Brieman and A Cutler. Random forests. https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.

[39] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[40] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *Random forests*. Springer Series in Statistics. Springer New York Inc., 2009.

[41] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[42] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[43] Yi Lin and Yongho Jeon. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590, 2006.

[44] George H. Chen and Devavrat Shah. *Explaining the Success of Nearest Neighbor Methods in Prediction.* 2018.

[45] Cristofer Englund and Antanas Verikas. A novel approach to estimate proximity in a random forest: An exploratory study. *Expert systems with applications*, 39(17):13046–13050, 2012.

[46] Hongliu Cao, Simon Bernard, Robert Sabourin, and Laurent Heutte. A novel random forest dissimilarity measure for multi-view learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 1344–1351. IEEE, 2021.

[47] Jake S Rhodes, Adele Cutler, and Kevin R Moon. Geometry-and accuracy-preserving random forest proximities. *arXiv preprint arXiv:2201.12682*, 2022.

[48] Philip Sommer and Stefano Pasquali. Liquidity—how to capture a multidimensional beast. *The Journal of Trading*, 11(2):21–39, 2016.

[49] P. M. Swamidass, editor. *MAPE (mean absolute percentage error)MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)*, pages 462–462. Springer US, Boston, MA, 2000.

[50] Philipp Probst, Marvin N Wright, and Anne-Laure Boulesteix. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, 9(3):e1301, 2019.

[51] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.