Machine Learning for Enhanced Bond Risk Modelling

Erik Konstenius

MSc in Data Science

Copenhagen Business School

Final project in

Quantitative Risk Management

2022

This paper investigates if machine learning models could provide accurate bond ratings and a more complete and comprehensive view of the credit risk of bonds. The results may be of interest for investors, issuers, and investment banks. Models evaluated include XGBoost, CatBoost, random forest, multilayer perceptron, support vector machine and logistic regression. SMOTE is applied to deal with class imbalances. A random forest model optimized using Bayesian Optimization achieved the best performance with almost 70 % accuracy and a good distribution of incorrect predictions. The paper also acknowledges the importance of explainable AI in financial risk management by investigating global and local interpretability of the model using feature importance scores and Shap values.

1. Introduction

a. Bond ratings

A bond is a *fixed income instrument* that can in many ways be seen as a tradable loan and a cornerstone in most economies (Choudhry, 2010). An *issuer* could be a corporation, government, municipality or supranational that wants to raise capital by issuing a bond. By buying a bond, the investor is in principle lending money to the issuer. In return, the issuer pays back the principal amount plus interest at a specified date. The price "p" of a bond at a specific date is given by the following equation where "t" is the number of periods until maturity, "c" is the coupon payment, "r" is the interest rate, i.e., *the yield to maturity*, and "F" is the face value (Investopedia, n.d.).

$$p = \sum_{t=1}^{T} \frac{c}{(1+r)^t} + \frac{F}{(1+r)^t}$$

Equation 1. Bond pricing

The price of a bond can fluctuate based on the issuer's ability to pay back the principal amount plus the coupons the investors expect for taking on the risk. The risk originates from macro events such as political and economic indicators and the risk associated with the issuer. The total risk associated with holding issued bonds is often broken down into the following risks:

- credit risk – risk that the issuer goes bankrupt and defaults on the debt
- liquidity risk – risk that the instrument cannot be traded easily
- inflation risk – risk that the holder's assumption of the inflation rate is off
- interest risk – the instrument's sensitivity for changes in the interest rate.

Fitch, Moody's and Standard & Poor's are credit rating agencies that among other things give bonds ratings based on the issuers ability to pay back the issued amount. High rated bonds, like "AAA", are generally trading with a lower risk premium as the risk is comparably low (Fidelity.com). The ratings are comparable between rating agencies, and "the conversion table" between the agencies is shown below.

Moody's	S&P	Fitch	Moody's	S&P	Fitch
Aaa	AAA	AAA	Ba2	BB	BB
Aa1	AA+	AA+	Ba3	BB-	BB-
Aa2	AA	AA	B1	B+	B+
Aa3	AA-	AA-	B2	B	B
A1	A+	A+	B3	B-	B-
A2	A	A	Caa1	CCC+	CCC+
A3	A-	A-	Caa2	CCC	CCC
Baa1	BBB+	BBB+	Caa3	CCC-	CCC-
Baa2	BBB	BBB	Ca	CC	CC
Baa3	BBB-	BBB-	C	C	C
BB+	BB+			D	D

Figure 1. Bond hierarchy for Moody's, S&P and Fitch

It is important to note that these ratings should not be treated as absolute truths. Economists have in the aftermath of the financial crisis in 2008 blamed credit agencies like the three previously mentioned for severely underestimating the risk associated with instruments like subprime mortgage-backed securities. Parts of the blame was also directed to investors for naively believing that the agencies were correct (Wharton, 2009).

b. Motivation and topic delimitation

This paper will investigate if machine learning models could provide accurate bond ratings for bonds in both the primary and secondary market. Machine learning algorithms can automatically learn complex patterns in the data that may be difficult for humans to identify. They can also handle large amounts of data and can incorporate a wide range of different types of information, such as historical data, economic indicators, and market trends. This can help to provide a more complete and comprehensive view of the credit risk of a bond. It could also help investment banks give more accurate price recommendations to the issuer and thereby give a better estimate of how much the investor is expected to pay for issuing debt.

I personally found this topic interesting as a machine learning-based model could ensure that all bonds, whether they are already issued or not, receive a rating. Today, each rating agency only produce ratings for a small portion of all bonds. This means that getting accurate aggregated statistics of for example high yield corporate bonds issued in NOK is difficult as many of the bonds do not have a bond rating and thus is not included.

The model will be trained to predict which credit rating a given bond should have. The label is a custom-built rating that takes an equally weighted average of a numeric representation of the three ratings by S&P, Moody's and Fitch. The model is mainly trained on features that describe the credit risk associated with the bond. This delimitation is chosen because of an inability to extract more data. Future papers could investigate how more focus on liquidation risk, inflation risk and interest risk could improve the performance of the model.

2. Dataset

Bloomberg's search function for fixed income data was used to extract the data. The dataset consists of five thousand issued bonds where the amount issued exceeds fifty million euro, the issuer originates in Europe and the bond has at least one rating from either Fitch, Moody's or S&Ps. The dataset of bonds included ones issued between the years 1996 and 2022. Some common features that may be used to model the credit risk of bonds include the issuer's credit rating, the bond's maturity date, the bond's interest rate, the bond's coupon rate, the bond's outstanding balance, the issuer's financial strength, the issuer's industry, and the overall state of the economy (Choudhry, 2010). The features used in this paper are listed below.

- Issuer
- ISIN
- Rating S&P
- Rating Moody's
- Rating Fitch
- Amount issued
- Years to maturity
- Issue date
- Total assets
- Debt to common equity
- I-spread
- Total debt to total capital
- Currency
- Adjusted duration
- Country of origin

Furthermore, a second dataset has been used to map the country code to a region. This mapping reduces the dimensionality of the feature space and helps the model converge faster. The dataset to map the country code to a region can be accessed here.

Bias in machine learning models can lead to unfair and inaccurate predictions or decisions. For example, if a model is trained on data that is not representative of the problem you are trying to solve, it may not be able to generalize well to new data, and this can result in mediocre performance on unseen data. One way to overcome bias in a machine learning model is to ensure that the data used to train the model is representative of the general population. The dataset in this paper has been delimited to European issuers as a larger scope would disperse the regional distribution of the datapoints and leave only a few observations in each region and thus ending up with a fair representation of each regions is improbable. The results from this paper are at the same time most relevant to European bond issuances. It is also important to carefully choose the metrics used to evaluate the model, as these can influence the model's performance and its ability to generalize to new data. The motivation for the choice of metrics is discussed in section 5.c. Additionally, using techniques such as regularization and cross-validation can help to reduce overfitting and improve the model's ability to generalize to new data. Worth noting is the presence of the bias-variance tradeoff where complex models that are custom built for one purpose often tend to show exceptional performance on that set but has so high bias that it can only be applied in just that setting. The tradeoff refers to the fact that reducing one type of error often leads to an increase in the other type of error. Simpler models generally have less bias but higher variance, while models that try to capture complex relationships between the features often have more bias but less variance. Multiple models will be evaluated to find one that is suitable for the particular dataset. More on the shortlisting of models will be discussed in section 5.a.

3. Regular ordinal classification, custom ordinal classification, regression

Supervised machine learning algorithms fall into two fields that each try to solve a different type of task. Classification algorithms tries to find decision boundaries that separates datapoints into classes. The goal of regression algorithms is to predict a target numeric value (Géron, 2019). Bond ratings are ordinal values where the ratings have an order to them. We would ideally want to capture this ordering in the model. This paper will try three different methods to train a model to predict the bond ratings. First, the dataset will be used to solve a regular ordinal classification task where the ordering is implicitly defined by the label value. Second, the dataset will be used to solve an ordinal classification task where the labels contain information about the ordered nature. In this task, each rating is represented as a vector of ones and zeroes. The number of ones in one vector represents the order of the rating from k -1 to 0 where k is the number of ratings. The highest rating, "AAA", will be represented by an all-ones column vector of length 17. This approach has been influenced by a paper by Mark Hall and Eibe Frank but has been extended to work on neural networks (Frank and Hall, 2001). Thirdly, the dataset will be used to solve a regression task where the continuous prediction is converted back to a text rating. The next section describes how the ratings were encoded depending on the task.

4. Pipeline: preprocessing, modelling, tuning, evaluation

This section will highlight some of the key data wrangling that has taken place to create a dataset that is suitable for training. A more detailed guide is found in the notebook.

a. Custom bond rating

The first preprocessing step was to fix the data types. This included converting the bond ratings to a numeric representation that machine learning algorithms can process. The custom rating was created by calculating the mean over the ratings given by the three rating agencies and taking the integer value of the average. This calculation returned as expected values that were similar to Bloomberg's own composite rating. Bonds that

had too few observations were merged. A model with more classes is more difficult to train than a model with fewer classes if the data size is the same. Furthermore, a model with highly imbalanced classes will in extreme cases completely ignore the minority class. Upsampling a minority class that contains only a few observations could introduce bias in the model if the observations in the minority class are not representative of that class as a whole. The ratings as well as the two proposed ways to represent it in numerical form are shown below.

Custom rating	Numeric encoding	Custom ordinal encoding						
AAA	0	1	1	1	...	1	1	1
AA+	1	1	1	1	...	1	1	0
AA	2	1	1	1	...	1	0	0
AA-	3	1	1	1	...	0	0	0
A+	4	1	1	1	...	0	0	0
A	5	1	1	1	...	0	0	0
A-	6	1	1	1	...	0	0	0
BBB+	7	1	1	1	...	0	0	0
BBB	8	1	1	1	...	0	0	0
BBB-	9	1	1	1	...	0	0	0
BB+	10	1	1	1	...	0	0	0
BB	11	1	1	1	...	0	0	0
BB-	12	1	1	1	...	0	0	0
B+	13	1	1	1	...	0	0	0
B	14	1	1	1	...	0	0	0
B-	15	1	1	0	...	0	0	0
CCC	16	1	0	0	...	0	0	0

Figure 2. Bond rating encodings. The numeric encoding is used for the regression models and the regular ordinal classification. The custom ordinal encoding is used for classification tasks that accepts an entire sequence as label.

b. Nulls

The simplest way to deal with nulls is to remove all rows with a null value in any of the columns. This method is infeasible in this case as it would omit a sizable portion of the entire dataset. We can also assign the nulls a value like the average of that column. A drawback with that method is that it does not make use of the relationship between the null value and the other columns. Another drawback is that the common method of doing it would create data leakage. Instead, I ran an ensemble of decision trees using the random forest algorithm to impute values in one column at a time. For each column, a random forest regressor is trained to predict the missing values in that column based on the values in the other columns. I chose to only include the columns with floats in this model as they are "the closest to being preprocessed" out of all the columns. These columns are however not ideal as the column values are on vastly different scales. Random forest is however an algorithm that can, to some extent, deal with both values in vastly different scales and outliers as opposed to using algorithms like the unsupervised KNN that would otherwise be a good option. If I would have used the KNN, the Euclidean distance between two points would not be a good estimator to impute values because of the different scales of the feature values.

c. Outliers

Outliers are data points that are significantly different from the majority of the data. These data points may be errors or valid examples that are simply unusual. Outliers can cause the model to overfit to the noise in the data, which can result in poor performance on new data. Outliers can also cause the model to make predictions that are far from the true values, which can reduce the performance of the model (Géron, 2019). The data appears to have quite extreme outliers (see figure 1 in the appendix). The highest five percent in the columns "amount_issued", "total_assets", "debt_to_common_equity" and "i_spread_mid" will be classified as too extreme to be accepted in the training phase. That means that an observation will be deleted if a value in any of the four columns exceeds the limit of what is acceptable. The 1 % lowest values in the column

"i_spread_mid" are also removed. The removal of outliers decreased the number of observations from 4,984 to 4,519. Figure 2 in the appendix shows a feature scatterplot matrix after the outliers have been removed.

d. Feature engineering

A date variable could be important in this case considering that the risk sentiment of a particular bonds is expected to correlate with the current economic environment. Figure 3 in the appendix shows how the average credit rating changes over time. But it is important to represent the date value in a meaningful way. The date column is converted to only capture the year. All numeric features are then standardized. This means that the values are subtracted by the mean and divided by the standard deviation. Some form of feature scaling is necessary for practically all machine learning algorithms when the features have very different scales (Géron, 2019). Support Vector Machines with the RBF kernel or L1 and L2 regularization in linear regression and logistic regression assume that the features are zero-centered and have the same unit variance (scikit-learn, n.d.). The dataset used in this paper both contains features like the amount issued that can be billions of EUR and ratios. Standardization zero-centers the data and makes the features have unit variance. Standardization is effective for datasets with outliers (Géron, 2019). The equation for standardization is shown below.

$$z = \frac{x - \mu}{\sigma}, where\ \mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)\ and\ \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Equation 2. Standardization

The country codes are joined with a second dataset to retrieve the sub-region the issuer originates from. This representation will create a less sparse feature vectors and will not increase the dimensionality of the parameter space too much. This will help the model converge faster and hopefully return higher performance. Furthermore, only the tops four sub-regions are chosen. The sub-regions are Western Europe, Northern Europe, Southern Europe, and Northern America. Figure 4 in the appendix shows the average rating across the four sub-regions. Bonds issued in Western Europe appears to be the most stable while bonds issued in Southern Europe is perceived as riskier.

e. Splitting

The dataset was split into a training, a validation, and a test set. Each corresponding to 70 %, 10 % and 20 % of the total dataset respectively. The train set is used to train the model. The validation set is used to evaluate the performance of out-of-the-box baseline models. The test set is used to evaluate the performance of the final hyperparameter tuned model(s). The smallest set contains about 400 observations – which should be enough to get a fair distribution of data with respect to the true data generating process of the entire bond universe given the delimitations mentioned in section 1.b. Furthermore, each model has also been evaluated using 5-fold cross validation to reduce the chance that the model happens to get a set whose distribution is different from what is expected. The metrics used to evaluate the methods will be discussed later in this paper.

f. Upsampling

An imbalanced dataset is one where the distribution of labels is highly skewed. Models trained on highly imbalanced datasets will spend most of its time training on the majority class and not learn enough from the classes with fewer observations to create good decision boundaries. In some cases, this results in a classifier that never classifies new samples as the minority classes. Some models are better at dealing with imbalanced dataset than others, yet the issue remains to some degree (Google Machine Learning, 2022). The distribution in this

dataset is centered between the ratings in the middle (see figure 3 below). The three ratings with the highest number of observations contains almost twenty-one times more observations than the three ratings with the lowest number of observations.

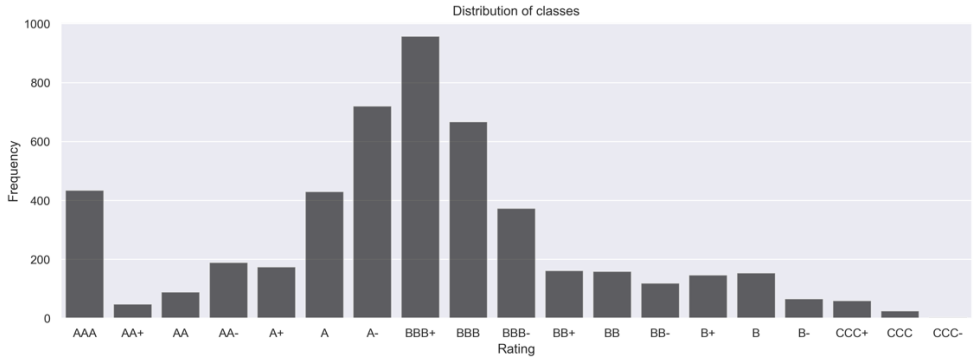

Figure 3. Distribution of bond ratings (0 = least risky, 16 = most risky)

The model could not afford under sampling the majority classes because of the small dataset. Synthetic Minority Over-sampling Technique (SMOTE) has been applied to upsample the minority classes in the training data. In short, SMOTE interpolates new data samples to the minority class by selecting a datapoint from the minority class as well as a number of neighbors in the feature space that also belong to the same class. New observations are then placed randomly between the selected observations. This process is repeated until the classes are equally large (Chawla et.al, 2011). The result is a dataset where each class contain the same number of observations. The upsampling is only done to the training data in order to prevent data leakage. Upsampling the entire dataset would change the distribution of the test set and prediction made on new samples would not reflect the true data generating process.

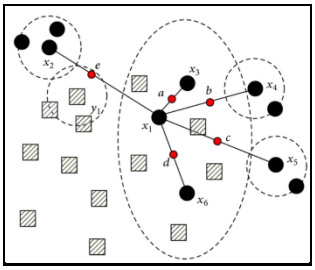

Figure 4. Upsampling using the Synthetic Minority Over-sampling Technique (SMOTE)

5. Modelling

a. Shortlist promising models

It is advised to train multiple models that come from different families of algorithms. The idea is to get a sense of which algorithms work and tune the models that show the most promising results. The models used in this paper include linear regression, logistic regression, support vector machines (SVM), random forest, XGBoost, CatBoost and neural networks (multilayer perceptron). Below is a very brief summary of each algorithm.

- **Logistic regression**: a modification of linear regression to enable discriminative classification. Linear regression calculates a weighted sum of the input features and an intercept. The output of the linear regression algorithm is fed to the sigmoid function that transforms the value to be between 0 and 1. Values higher than 0.5 are classified as the positive class, and values lower are classified as the negative class. For multiclass, the model is applied to each class individually and the final prediction is the class that receives the highest probability (Géron, 2019). The weights, or parameters, can be trained using a closed form solution through the normal equations or iteratively using gradient descent. Regularization can be applied using the L1 or L2 norm or elastic nets that applies both L1 and L2 regularization. The algorithm is lightweight and provides high interpretability. The algorithm is shown below.

$$\hat{p} = \sigma(x^t\theta), where\ \sigma(t) = \frac{1}{1 + \exp(-t)}\ and\ \theta\ is\ the\ feature\ vector$$

$$\hat{y} = \begin{cases} 0\ if\ \hat{p} < 0.5 \\ 1\ if\ \hat{p} \geq 0.5 \end{cases}$$

Equation 3. Logistic regression. The sigmoid function, σ(t), converts the output from a standard linear regression model into a range between zero and one, and outputs the positive class if the value is 0.5 or larger.

- **Support vector machines**: the goal of SVMs is to find the best hyperplane that separates the classes by maximizing the margin between support vectors. The model is often turned into a soft margin classifier to reduce its sensitivity for outliers. The kernel-trick is applied to capture non-linear relationships. This paper applied an RBF-kernel because it is usually difficult to understand which kernel works the best and the chosen kernel is popular and often the first choice (Murphy, 2022). The algorithm is a discriminant classifier by design but can be applied to regression tasks as well. The algorithm behind support vector machine is rather intuitive which makes predictions from the model more explainable compared to predictions made by more complex algorithms like boosting algorithms or neural networks.

- **Random forest**: a bagging algorithm that trains an ensemble of decision trees. The algorithm starts by using bootstrapping to select a random sample of observations. For each observation it selects a random sample of features to create a decision tree model. Each decision tree repeatedly splits the data in a way to maximize the purity of each resulting split. The algorithm repeats this process for a specified number of iterations to create a forest of decision trees. The results from each tree are aggregated. For classification tasks, the final prediction could be a majority vote, while for a regression task, the final prediction could be the average. The hyperparameters of the model specify how trees should be created. The decision boundaries of a random forest are jagged as the predictions are made

by decision trees. The algorithm is robust to outliers, works well with non-linear data, has a low risk of overfitting and can deal with imbalanced datasets (Géron, 2019).

- **Extreme gradient boosting (XGBoost)**: ensemble boosting algorithm that sequential trains weak learners where each learners tries to learn from the errors, i.e., gradients, of the past learners. The model is initiated by a constant value. It then computes the residuals, trains a regression tree with the feature of the previous model, which at the first stage is just a line, and updates the prediction to incorporate information from the regression tree. The learning rate defines how much the model should listen to how the regression tree wants to change the prediction. XGBoost is an optimized version of gradient boosting that is easy to use and offer state-of-the-art performance (Géron, 2019).

- **CatBoost**: similar to XGBoost but requires less hyperparameter tuning and can deal with more data types. CatBoost also splits each leaf from the previous tree using the same condition resulting in symmetric tree growth. The feature-split pair that accounts for the lowest loss is selected and used for all the level's nodes. This increases generalizability and decreases training time (Neptune AI, 2020). It is currently ranked state-of-the-art and the best performing paper among 3856 submissions across eleven benchmarks and eight datasets (Papers with Code, n.d.).

- **Multilayer Perceptron**: building block of deep learning. The network is made up of neurons that have been organized into layers. Each neuron is generally connected to every other neuron in the following lower and upper layer and values from one neuron is passed to another by multiplying the value by a weight and then fed through an activation function like the sigmoid function. Each layer usually also has a bias. Training is conducted by initializing the weights and biases with random values, assuming transfer learning has not been used, and feeding batches of data through the network in a forward pass to generate a prediction. The prediction is then evaluated using a defined loss function and subsequent tweaks of the weights and biases of the network are then conducted by calculating the gradient of the loss function using backpropagation and the chain rule. Optimizers that are built on the concept of gradient descent update the weights and biases. Additionally, this paper has used batch normalization to regularize the network, make the network less sensitive to hyperparameter tuning and speed up training (Géron, 2019). A so-called "bias layer" has been added to restrict the model to only make predictions that are consistent with the encoded vectors representing the ratings used in this paper. This layer eliminates any chances for output vectors such as [0, 1, 0, 1, 1] by letting output neurons share weights but not biases. The bias layer was inspired by the paper "*Rank consistent ordinal regression for neural networks with application to age estimation*" by Kilcher (2019) and the implementation found here. The binary cross entropy loss together with the sigmoid function makes sure that each value in the output vector is either 0 or 1.

b. Hyperparameter optimization

The hyperparameters of the most promising models were tuned using Bayesian Optimization. Traditional methods for hyperparameter tuning like grid search and random search are either computationally expensive or can in the case of random search yield parameters that do not produce satisfactory performance. Traditional methods quickly become obsolete when the parameter space grows large. Bayesian Optimization searches for the best parameters by iteratively calling a surrogate function, in this case training a machine learning model, and creates a probabilistic model of the function mapping from hyperparameter values to the objective function. The objective function in the case of classification could be accuracy. The Bayesian Optimization

algorithm will run until a user-defined maximum trials (Optuna, 2020). The number of trials was set to 40 and the metric to optimize was the weighted cross validated F1 score. Details of the metric and why it is of interest is covered in the next section.

c. Evaluation metrics

In general, it is best to use multiple metrics to evaluate the performance of a multiclass classification model. Using different metrics provide different perspectives on the performance of a model and can help identify potential biases and shortcomings in the generated predictions. This can help give a more complete picture of the model's performance and identify its strengths and weaknesses. Model performance is in this paper evaluated by an overall assessment of the accuracy, precision, F1 score and five-fold cross validated F1 score as well as classification reports, distribution of predictions and confusion matrices. We direct special attention to the cross-validation metric as its metric is less dependent on one single split that could be unrepresentative of the greater population. Accuracy measures the proportion of correct predictions to the total number of predictions. Precision measures the proportion of positive predictions that are actually correct. The F1 score is the harmonic mean of precision and recall, and it provides a single score that represents how the model was able to predict the positive class correctly while keeping the false positives low (Géron, 2019).

The full modelling pipeline can be seen below.

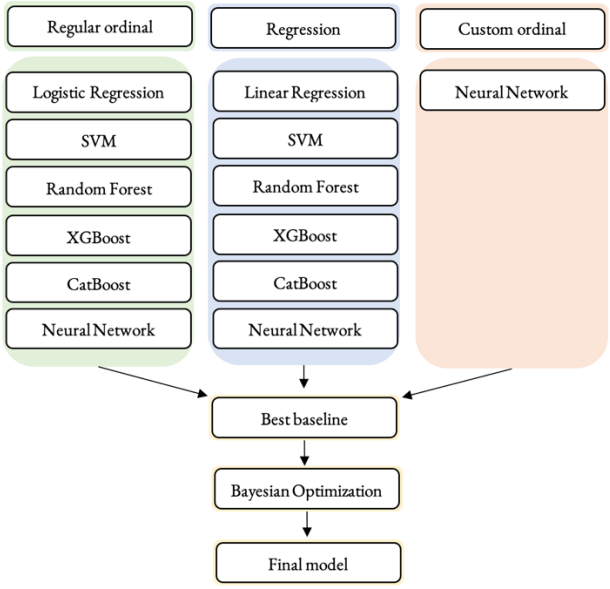

Figure 4. Modelling pipeline with each of the three tasks

6. Results

The results of the baseline models are shown below (see figure 5). The regular ordinal classification, where the ratings are labeled from 1 to k where k is the number of classes, received the highest performance of the three tasks across all metrics. The encoding used in the regression task and the custom ordinal task does not seem to improve the performance. Random forest and XGBoost using regular ordinal classification achieved the highest accuracy score on the validation set and highest average cross validation score among the models using any form of ordinal classification. The random forest was chosen as the contender for final hyperparameter tuning. I also optimized the XGBoost, see notebook for implementation if curious. Although the performance of the random forest was similar to the XGBoost, I decided to go for the random forest as it allows for parallel computing and is slightly easier to tune. The hyperparameters that were tuned include the number of trees in the forest, the function to measure the quality of the split, maximum depth of the tree, the number of features to consider when looking for the best split, the minimum number of samples required to be at a leaf node and the minimum number of samples require to split an internal node.

	Regular ordinal classification					**Regression**					**Ordinal custom**
	CATB	**LREG**	**RF**	**SVM**	**XGB**	**CATB**	**LREG**	**RF**	**SVM**	**XGB**	**MLP**
Accuracy	0.6333	0.2622	0.6867	0.36	0.7022	0.3644	0.1844	0.4511	0.2244	0.3711	0.1267
Precision	0.6591	0.3207	0.7142	0.4647	0.7272	0.4192	0.2212	0.4863	0.2998	0.4309	0.148
F1	0.6399	0.2735	0.6951	0.3803	0.7096	0.3732	0.1781	0.4555	0.2367	0.4309	0.1107
CV	0.889	0.3709	0.9128	0.579	0.8926	-1.1656	-2.3071	-0.9894	-1.6821	-1.1469	-

Figure 5. Results from baseline models

The performance of the hyperparameter tuned random forest on the test set is shown below.

	Regular ordinal classification			
Final Model	**Accuracy**	**Precision**	**F1**	**CV**
RF	0.6785	0.6881	0.6803	0.9204

Figure 6. Results from random forest model using regular ordinal classification and Bayesian optimization

The final random forest model used Gini impurity, max depth of the tree set to 20, maximum percentage of features to consider when splitting set to 22.365 %, minimum one sample required to be at a leaf node, minimum two samples to split internal node and the number of trees in the forest were 980. Comparing the cross-validation scores of the out-of-the-box random forest classifier with the one tuned using Bayesian Optimization we see that the improvements are negligible. The predictions made by the classifier can be seen in the figure below. Furthermore, figure 7 and 8 in the appendix show a confusion matrix and a classification report. The results from the figures show that in cases where the model makes classification mistakes the mistakes it makes is close to its actual rating. This characteristic is crucial as it shows that the model has been able to understand the ordering of the data. The effect of this is that incorrect ratings are within a tolerable margin of error or at least not entirely misleading. It is important to address that deviations from the true value could in parts originate from ratings not being updated very often and that rating agencies could under- or overestimate the risk of the bonds and give the bond a rating it does in theory not deserve looking at historical data. Converting the output to a range of +/- 1 rating of what it predicted and thereby allowing for a more general rating recommendation would lead to a close to perfect prediction accuracy.

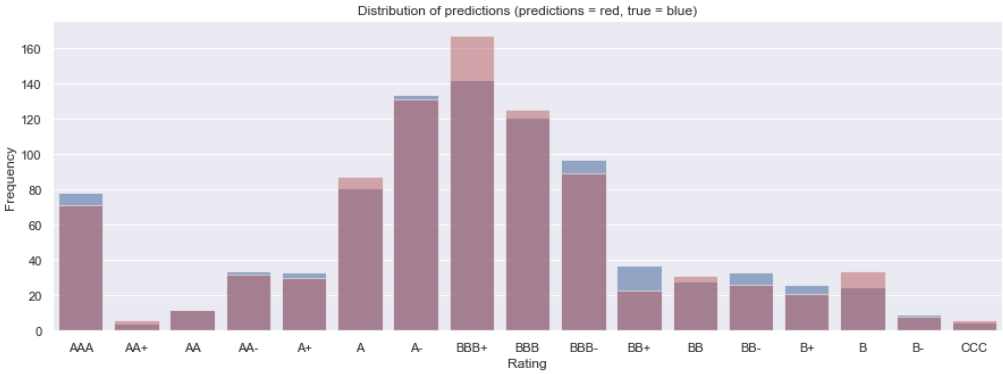

Figure 7. Distribution of final predictions on test set (predictions = red, true = blue)

7. Discussion

The results show that machine learning using rather rudimentary features can be utilized to model the risk associated with bonds with high accuracy. One argument why random forest produced such promising results could be that each decision tree is trained on a random subset of the training data. Additionally, the model is able to generalize well by only selecting a subset of all features at each split. Bayesian Optimization may have reduced the risk of overfitting by decreasing the max depths of each tree.

The promising results show that rating agencies, issuers, investment banks and investors can use models like the ones proposed in this paper to better understand the risk associated with certain bonds. Banks, credit agencies and investors are parties that undergo strict regulations and understanding how the model makes predictions is vital. Explainable AI allows for greater transparency and accountability in the decision-making processes of AI algorithms. This can help to reduce bias and errors and can also help financial institutions to comply with regulations that require them to be able to explain their risk management decisions. Random forest is a model that in terms of interpretability could be placed between simpler traditional machine learning algorithms and so-called black-box models like many deep learning models. The fact that the algorithm makes predictions based on an ensemble of 980 decision trees where each decision tree can be split into a number of leaf nodes means that understanding each prediction is somewhat convoluted. Going for a simpler model would be at the expense of quite a lot of decrease in performance (see figure 5). Scikit-learn offers a method to visualize the feature importance of random forest classifiers and thus the global interpretability of the model. For each feature in each split the method calculates the decrease in impurity. The average over all trees in the forest is the feature's importance. The results from the feature importance plot for the model train in this paper are shown below.

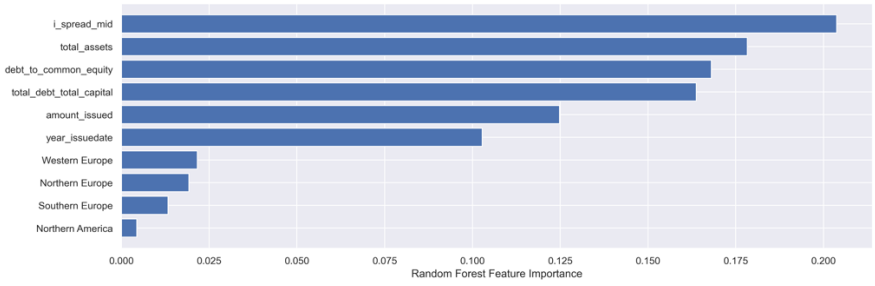

Figure 8. Feature importance plot (larger values = more important for the average prediction)

Furthermore, Shap values can be used to understand both global and local interpretability and thereby how changes to each feature affects a particular prediction. Shap values are generated by seeing how the prediction of a model changes if the values of the features change. The absolute Shap value tells how much a single feature affected the prediction. Positive Shap value means positive impact on prediction, leading the model to predict the class. A benefit of using Shap values is that they are model agnostic and can be applied to complex models (Shap, n.d). The results are shown below.

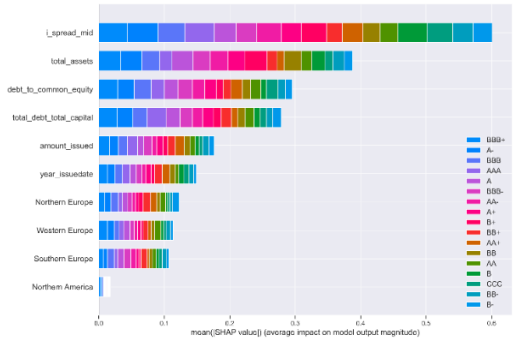

Figure 9. Shap summary plot (larger values = more important for the prediction)

The results from figures 8 and 9 show that the most important features in terms of global interpretability are, maybe not too surprisingly, the I-spread, total assets, and debt to common equity. This can help decision makers trust that the model is picking up the important features in the dataset. Partial dependence plots can be used to understand how varying values of one feature effects on single prediction (Shap, n.d). In the below figures, partial dependence plots are shown for the features "amount issued" and "Southern Europe" in cases where the predicted class is "AAA" .

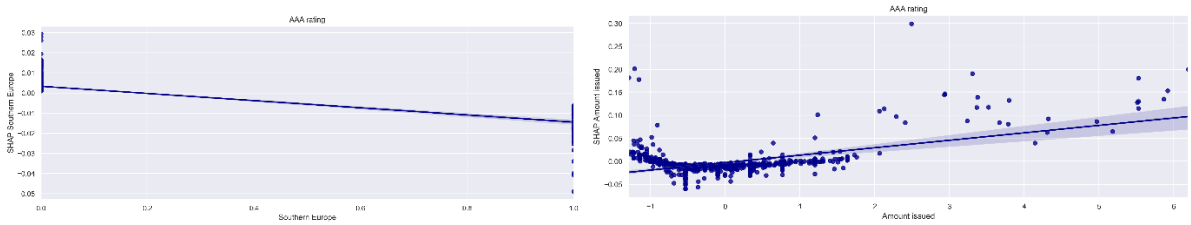

Figure 9. Shap partial dependence plots for Southern Europe (left) and Amount Issued (Right) (larger values = more important for the "AAA" prediction)

The selected features were cherry-picked to highlight the power of local interpretability through Shap values. The results from figure 9 show that issuers originating in Southern Europe decrease the probability of the bond receiving the rating "AAA". Also, the more the issuer is issuing, the higher the probability that the model predict "AAA". The results could be reasonable considering that governments and supranationals like the European Union often issue very large amounts and also have strong ratings.

Lastly, it is possible that using machine learning to better understand risks could introduce additional risks that need to be managed. This could occur if the machine learning model is not properly trained, tested, and validated, or if the data used to train the model is biased or inaccurate. In order to mitigate these risks, it is important to have strong governance processes in place for the development and deployment of machine learning models. This includes establishing clear goals and objectives for the use of the model, selecting, and preparing the data carefully, and implementing robust testing and validation procedures to ensure the accuracy and reliability of the model. The cost of an incorrect prediction by a machine learning model used to predict the credit risk of bonds could be significant. If the model incorrectly predicts that a bond has a low credit risk, investors may be more likely to purchase the bond, only to find out later that the borrower is unable to make the required interest and principal payments. This could result in a loss of capital for the investors, as well as damage to their reputation and trust in the financial system. On the other hand, if the model incorrectly predicts that a bond has a high credit risk, investors may be less likely to purchase the bond, even if it is actually a safe investment. This could result in missed opportunities for investors, as well as potential losses for the borrower who is unable to access the necessary capital.

8. References

Bond ratings. Fidelity.com, n.d. https://www.fidelity.com/learning-center/investing-basics/fixed-income-investing/bond-ratings.

Chawla, Nitesh V and Bowyer, Kevin W and Hall, Lawrence O and Kegelmeyer, W Philip. SMOTE: Synthetic Minority Over-sampling Technique. arXiv preprint arXiv:1106.1813, 2011.

Choudhry, M. (2010). Fixed-Income Securities and Derivatives Handbook: Analysis and Valuation (2nd ed.). John Wiley & Sons.

Frank, Eibe and Hall, Mark. A simple approach to ordinal classification. In: Machine Learning: ECML 2001, 145-156. Springer, 2001.

Géron, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. O'Reilly Media, 2019.

Investopedia. (https://www.investopedia.com/terms/b/bond-valuation.asp). Bond Valuation

Kilcher, Yannic. "Deep Learning for tabular data." arXiv preprint arXiv:1901.07884, 2019. https://arxiv.org/pdf/1901.07884.pdf.

Murphy, Kevin P. Probabilistic Machine Learning. MIT Press, 2022.

Neptune AI. "When to Choose CatBoost Over XGBoost or LightGBM." Neptune AI, 2020. https://neptune.ai/blog/when-to-choose-catboost-over-xgboost-or-lightgbm.

Optuna. "Optuna: A Next-generation Hyperparameter Optimization Framework." Optuna, 2020, https://optuna.org/.

Papers with Code. "Classification Tasks and Top Performing Models." Papers with Code, n.d. https://paperswithcode.com/task/classification.

Reforming the Ratings Agencies: Will the U.S. Follow Europe's Tougher Rules? knowledge at Wharton, May 27, 2009. https://knowledge.wharton.upenn.edu/article/reforming-the-ratings-agencies-will-the-u-s-follow-europes-tougher-rules/

SHAP documentation. (https://shap.readthedocs.io/en/latest/#)

Scikit-learn. "sklearn.preprocessing.StandardScaler." scikit-learn: Machine Learning in Python, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

Stack Exchange. "How to set up neural network to output ordinal data? - Cross Validated." Stack Exchange, 2019. https://stats.stackexchange.com/questions/140061/how-to-set-up-neural-network-to-output-ordinal-data/324879#324879.

9. Appendix.

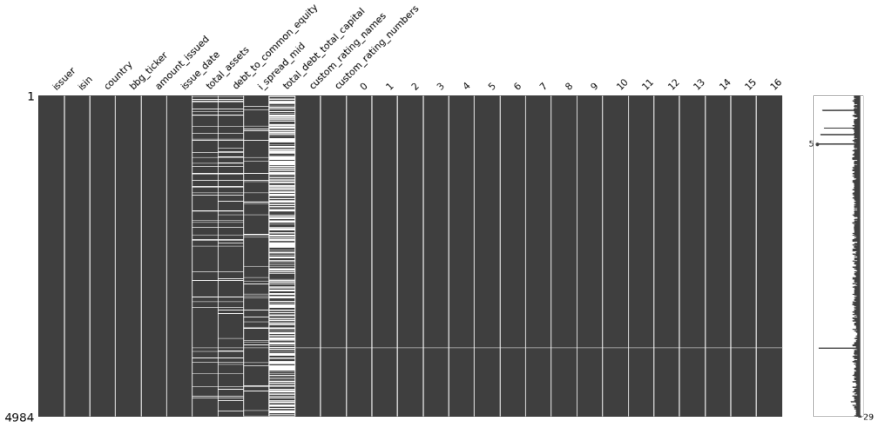

Figure 1. Distribution of nulls

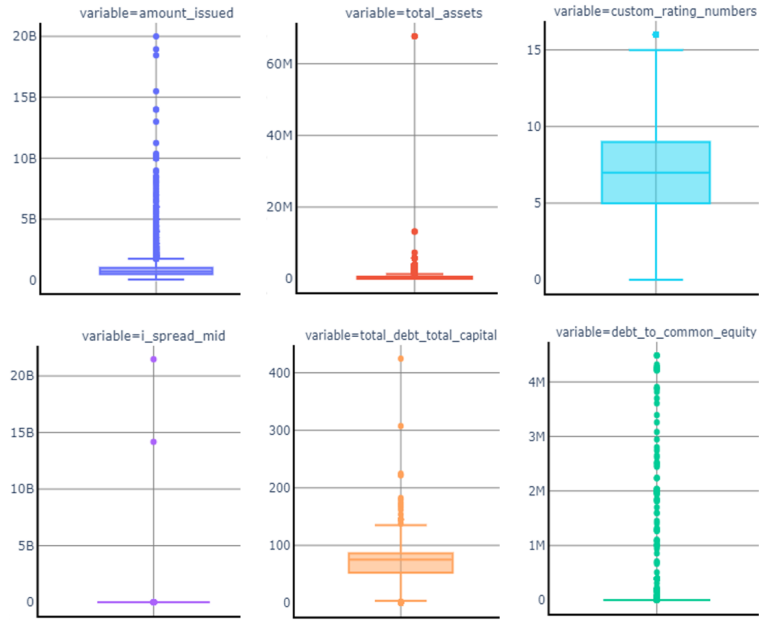

Figure 2. Boxplot for numerical features and the custom rating (0 = least risky, 16 = riskiest)

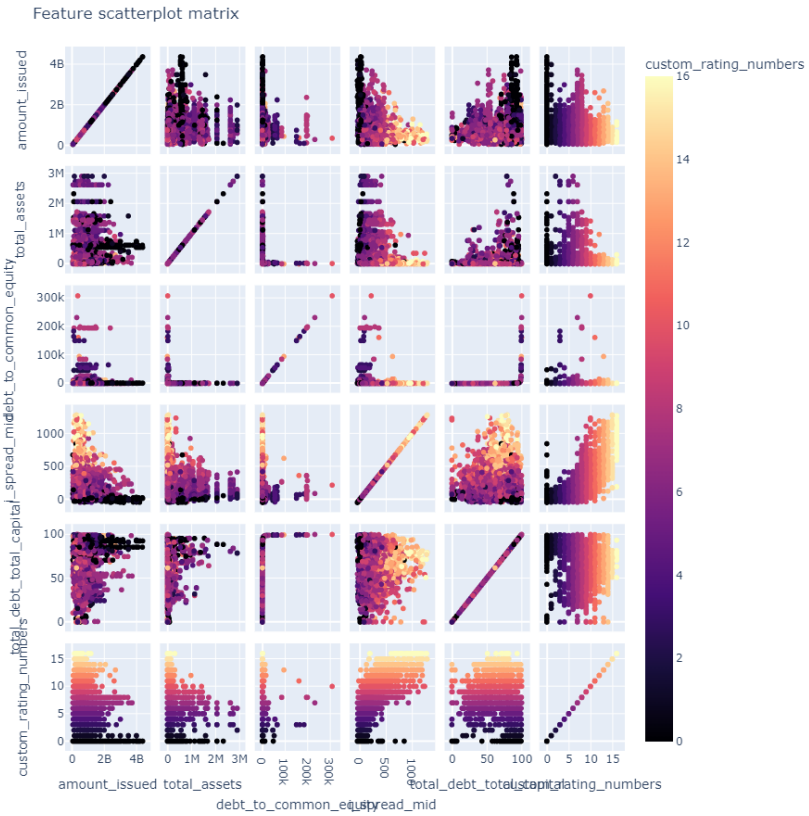

Figure 3. Feature scatterplot matrix (0 = least risky, 16 = most risky)

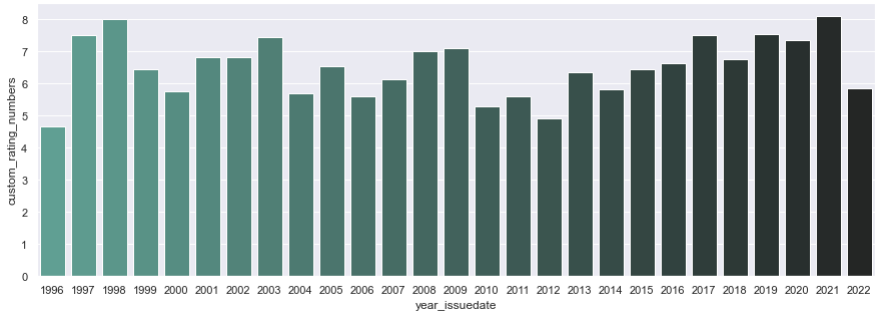

Figure 4. Average bond rating (numeric representation) by year (0 = least risky, 16 = most risky)

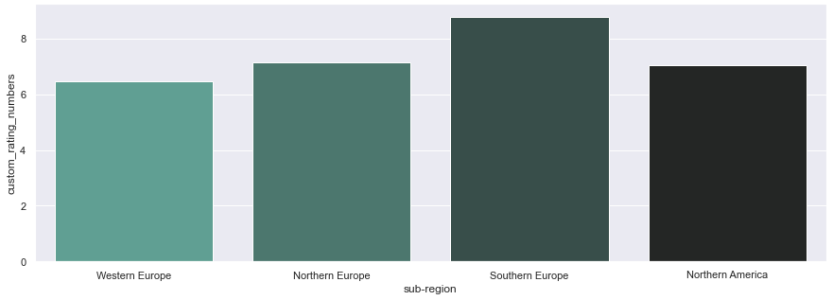

Figure 5. Average bond rating (numeric representation) by year (0 = least risky, 16 = most risky)

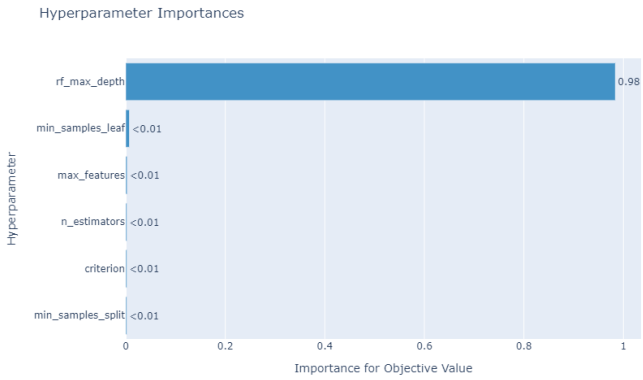

Figure 6. Hyperparameter Importance using Bayesian Optimization of a random forest classifier with regular ordinal classification

	Precision	Recall	F1	Support
AAA	0.9	0.99	0.94	71
AA+	0.25	0.17	0.2	6
AA	0.75	0.75	0.75	12
AA-	0.76	0.81	0.79	32
A+	0.64	0.7	0.67	30
A	0.77	0.71	0.74	87
A-	0.73	0.75	0.74	131
BBB+	0.8	0.68	0.73	167
BBB	0.71	0.69	0.7	125
BBB-	0.58	0.63	0.6	89
BB+	0.38	0.61	0.47	23
BB	0.36	0.32	0.34	31
BB-	0.45	0.58	0.51	26
B+	0.27	0.33	0.3	21
B	0.64	0.47	0.54	34
B-	0.33	0.38	0.35	8
CCC	0.6	0.5	0.55	6
Accuracy			0.68	899
Macro avg	0.58	0.59	0.58	899
Weighted avg	0.69	0.68	0.68	899

Figure 7. Classification report

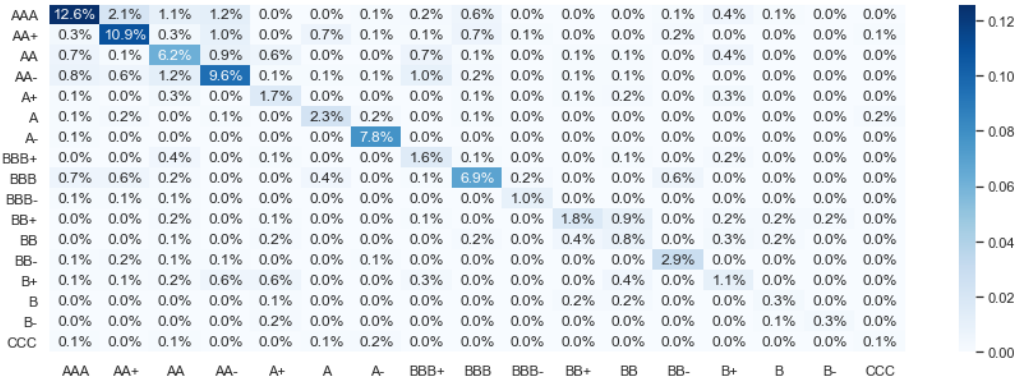

Figure 8. Confusion matrix