"A Machine Learning Journey into Corporate Bond Liquidity Score Estimation"

By Leonardo Bosco, Greta Kichelmacher and Emanuele Carosi

E-mails:

leonardo.bosco2608@studenti.luiss.it

greta.kichelmacher@studenti.luiss.it

emanuele.carosi@studenti.luiss.it

Table of Contents

Introduction

The aim of the project was to develop a regression model that could accurately predict the Liquidity Score of corporate bonds. The Liquidity Score is a synthetic indicator used to represent the ease of buying and selling a bond in the market without significantly affecting its price, and investors use it to estimate the liquidity of a corporate bond. In recent years, several leading financial service providers, such as Bloomberg, Markit and Ice, have developed their own liquidity score, using machine learning methodologies. This project focused on the prediction of the Liquidity Score evaluated by Bloomberg, which has a range of values between 1 and 100.

The final dataset was created by aggregating three subsets of data collected by Bloomberg. Each subset contained information on bonds that were available on the specific day the Bloomberg terminal was consulted. All bonds in the dataset shared a set of financial variables, and some additional variables were present in only one of the subsets. The initial dataset had 17 variables, but through financial and analytical analysis, it was reduced to 7 variables that were included in the final models. These variables are Coupon Type, Outstanding Amount, Time to Maturity (years), Yield, Mid Price, LQA Price Volatility, LQA Expected Volume, and LQA Liquidity Score.

Methods

Financial Understanding

After collecting the data, it was necessary to understand the financial meaning of each variable. Below, a brief explanation will be provided for each variable.

- Coupon Type: it refers to the type of interest or coupon rate. It can assume 5 different values: Fixed, Floating, Step CPN, Variable, Zero Coupon.
- Outstanding Amount: it represents the total amount of the bond that is still outstanding and has not yet been redeemed or settled by the issuer.
- Duration: it indicates the weighted average duration of the bond's expected cash flows, including interest payments and principal repayment, expressed in years.
- Time to Maturity: it indicates the time remaining in years until the maturity of the bond.
- Yield: it is the expected yield of the bond, expressed as a percentage of its nominal value.
- Mid Price: it refers to the average price between the bid and ask price of a corporate bond.
- Lowest Quoted Ask (LQA) Price Volatility: it refers to the lowest selling price (the 'ask price') of a bond in a given time period.
- Lowest Quoted Ask (LQA) Expected Volume: it indicates the expected daily trading volume for a particular bond.
- Lowest Quoted Ask (LQA) Liquidity Score: it corresponds to the Liquidity Score evaluated for a bond by Bloomberg.

Cleaning and Exploratory Data Analysis

As previously mentioned, although each dataset contained the variables reported above, there were also additional variables that were not present in the other two datasets. To address this asymmetry, two courses of action were taken. On the one hand, because the joint variables were the most financially critical for the analysis, we decided to clean up each dataset and create a new one with only those. On the other hand, an investigation was carried out to determine if there were any differences or biases among the three datasets, or if the additional variables were important for the analysis. Therefore, both each independent dataset and the one consisting of all three were analysed.

The analysis phases for each dataset were very similar and can be summarized in 5 steps:

1) Data cleaning:
    a. Exclusion of duplicates bonds.
    b. Drop not significant features for the analysis: ID, Security, Issuer name, Ticker, S&P Rating, ESG Disclosure Score, Country of Incorporation.

c. Drop null values.
d. Drop of error values (values out of feature range).
e. Rename variables to make them homogeneous in all three datasets.
f. Creation of merged dataset containing all the others.

2) Correlation Matrix and exclusion of variables with high correlation. There were several variables that were highly correlated, which potentially might have led to higher complexity and overfitting. Therefore, it was necessary to drop the ones less significative. ‘Duration’ and ‘Time to maturity’ were highly correlated. Since the former presented high correlation with Mid-price and LQA Price Volatility as well, it was dropped. Also ‘Ask price’, ‘Bid price’, ‘Bid Ask spread’, and ‘Mid-price’ had the same problem. Of these four, the first three were dropped in favour of ‘Mid-price’. ‘Mid price' was retained as it was considered to be a more informative feature than the other three variables because it represents the average price between the Bid price (the price at which a buyer is willing to buy the stock) and the Ask price (the price at which a seller is willing to sell the stock).

3) Stratification of the dataset. As the distribution of the Liquidity Score was not constant among different values, a data binning methodology was implemented to partition the original dataset into a training and test set containing the same proportion of Liquidity Score values. This involved dividing the original dataset into 10 different subsets, or 'bins', where each bin contained observations in a specific range of values of the LQA Liquidity Score. Each bin spanned 10 values of the LQA Liquidity Score, with the first bin ranging from 1 to 10, the second from 11 to 20, and so on. To create the train and test sets, 80% of each bin was used for training and the remaining 20% was used for testing.

4) Encoding of categorical variable. Since the coupon type is a categorical variable, it was necessary to transform its values. Specifically, it was used the dummy variables approach which resulted in the creation of five new columns that were added to the datasets, each representing one of five possible values of the coupon type.

5) Normalization of numerical variables. To avoid any potential bias and ensure that all variables had a similar impact on the model’s performance, it was necessary to standardize all the variables of the training and test set, except for the Liquidity Score, which is the response variable. To achieve this, the standard scaler on the matrix (X) was fitted with the training set, and then, the test set was normalized using the same standard scaler. This was crucial to ensure that the normalization process

did not introduce data leakage, which occurs when information from the test set is used to scale the training set, leading to overly optimistic evaluation metrics.

With respect to the models, the starting point was to develop more interpretable models and thus, began by implementing a Multiple Linear Regression and a Support Vector Regressor model. However, due to their scarce accuracy, the use of three different Ensemble methods, namely Random Forest, Gradient Boosting Regressor and Extreme Gradient Boosting Regressor, and a Neural Network were necessary to improve the results. To evaluate the performance of each model root mean square error (RMSE) and R coefficient (R-squared) were calculated on the test set of the aggregated dataset.

Finally, in order to gain a better understanding of variable contribution, the 'Shap' library in Python was utilized. This library employs a mathematical method to provide explanations for the predictions made by machine learning models to evaluate the feature importance and the financial insights of each predictor.

Below is reported the diagram with the various stages of the project.

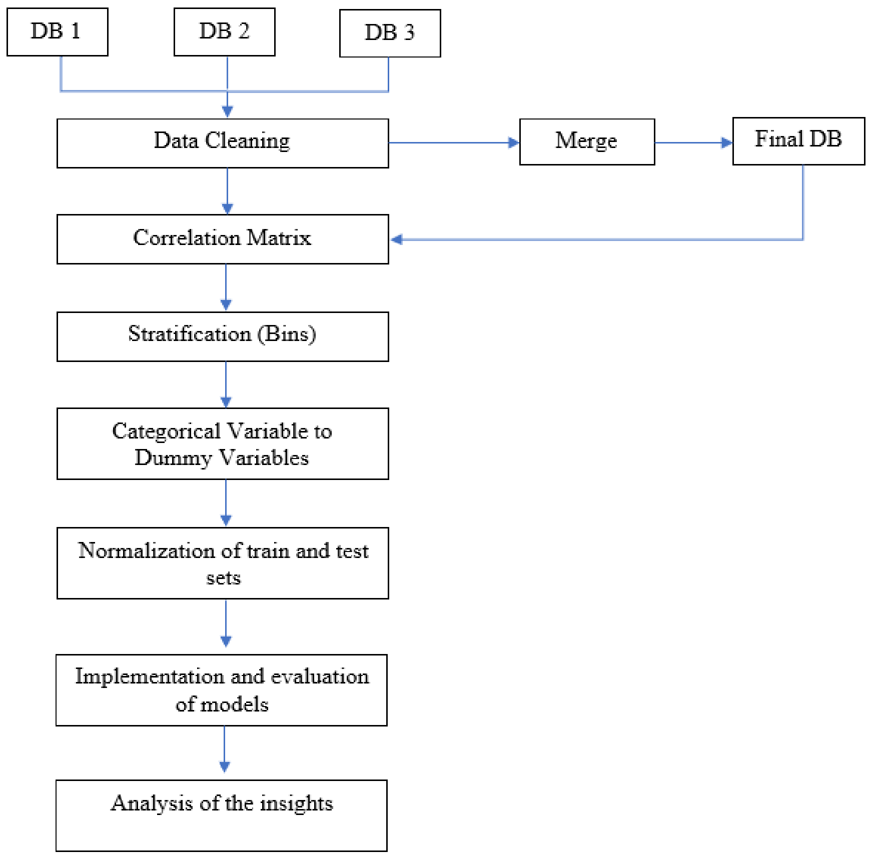

DB 1
DB 2
DB 3
Data Cleaning
Merge
Final DB
Correlation Matrix
Stratification (Bins)
Categorical Variable to Dummy Variables
Normalization of train and test sets
Implementation and evaluation of models
Analysis of the insights

Fig.1 Stages of the project

Experimental Design

Introduction

In this section, a description of all the implemented models will be presented. For each model, there will be a brief description of how it works, the results of the performance metrics and the tuning methodology. For models that have the possibility of performing a grid search, the characteristics of the grid search will also be provided.

As performance metrics both R-squared and RMSE were chosen to provide a more comprehensive evaluation of the performance of each model. R-squared measures the proportion of variation in the target variable that the model explains, while RMSE measures the magnitude of the error between the model's predictions and the actual values. Evaluating both metrics allows for a better understanding of the strengths and weaknesses of each model and enables the selection of the best performing one.
On the one hand, RMSE is advantageous because it maintains the same scale as the predicted variable, allowing for a more intuitive interpretation of the error in the model's predictions. On the other hand, R-squared provides a means of comparing the performance of different models, as it ranges from 0 to 1, with higher values indicating a better fit to the data.

Regarding the metrics used to evaluate the models, it is important to note that each model was run five times, and the reported R-squared and RMSE are the resulting means. Additionally, to ensure the absence of potential fluctuations in accuracy and the consistency of the predictions, the standard deviation of R-squared was also included.

Models

Interpretable Models

Linear Regression

The first model implemented in the study was a Multiple Linear Regression, which uses the method of minimizing the sum of squared error to determine the best line of fit. Although the distribution of the outcome variable was not gaussian, this method was utilized to establish a baseline for comparison with the results of the other models. The multiple linear regression model yielded an R-squared value of 0.4091 and an RMSE value of 21.055.

A regression line of the model plotted against the data is shown below.

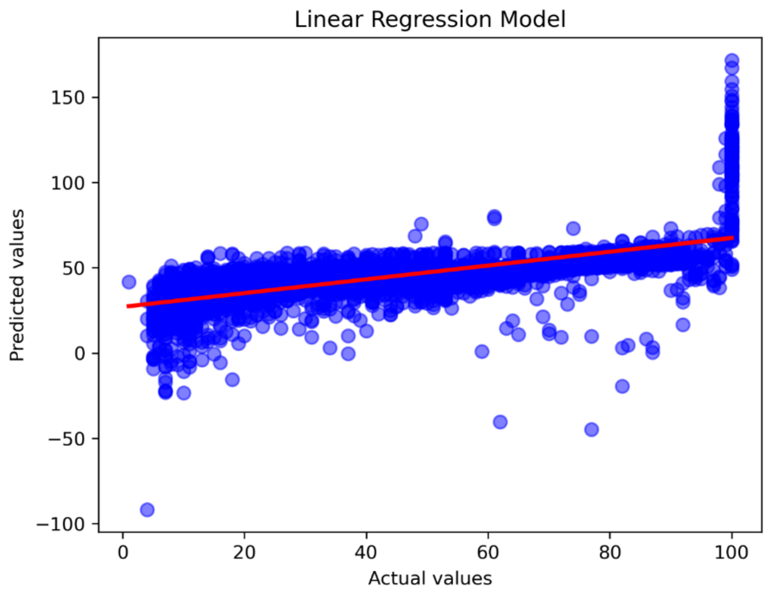


Fig 2. Linear Regression Model

Support Vector Regressor (SVR)

Support Vector Regressor (SVR) is a machine learning algorithm that can handle non-linear relationship between data by using kernel methods to transform the data into higher dimension. The SVR works by finding the hyperplane that maximizes the margin between support vectors and using this hyperplane to make predictions.

To tune the SVR, the RandomizedsearchCV method was employed, resulting in the following best hyperparameters:

- C = 100
- gamma = 0.6
- kernel = "rbf".

The SVR model achieved an R-squared of 0.9096 and an RMSE of 15.091.

In the graph below, a 2-dimensional representation of how the best calculated hyperplane approximates the observations is shown.

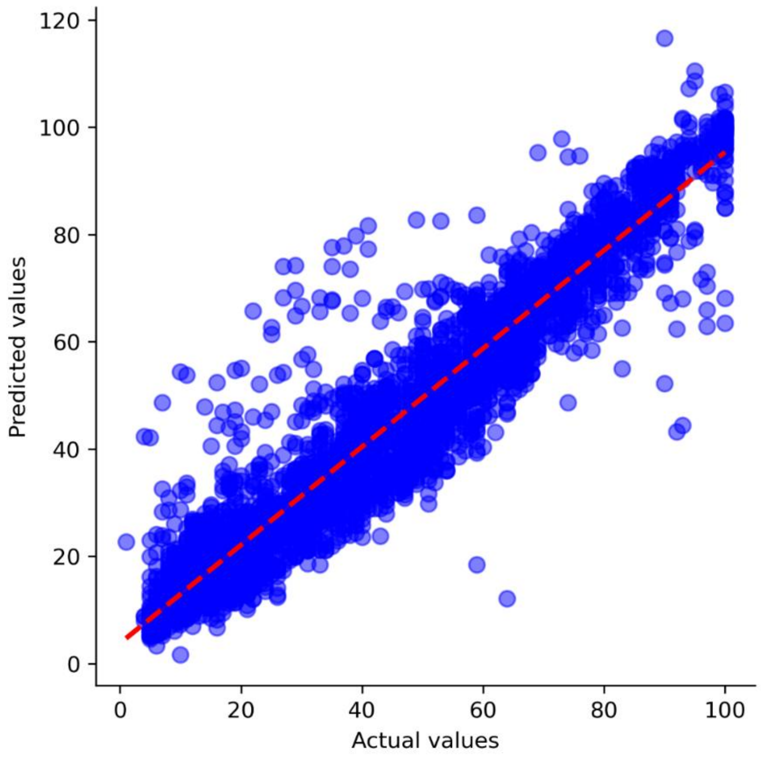


Fig. 3 Support Vector Regressor best hyperplane

Ensemble Methods

The aim of Ensemble Methods is to enhance the generalization performance of a classifier by combining multiple individual classifiers. In this study, three ensemble methods were used, each of which involved a combination of Classification and Regression Trees (CART) with a different technique for generating the training set.

To make the final predictions, each ensemble method utilized the "majority vote" approach, which consists in calculating the mean value of the predictions made by each tree. Through these methods, it was possible to take advantage of the accuracy of regression trees while limiting their drawbacks, such as the strong dependence of the model on the training dataset used.

Random Forest

The Random Forest model consists of multiple trees whose training set is evaluated through the Bagging technique. The Bagging algorithm iteratively evaluates a different sample with replacement of the same original training set for each tree.

As expected, this model achieved much better results than the previous ones, with an R-squared of 0.9814 and an RMSE of 3.7309.
The best hyperparameters found through RandomizedsearchCV were:

- n_estimators = 1000
- min_samples_split = 2
- max_features = 'log2'
- criterion = 'squared_error'
- bootstrap = False

Gradient Boosting Regressor

The Gradient Boosting Regressor is an iterative algorithm that uses a weighted version of the training set to train the algorithm at each iteration. At the first iteration, it uses all the original training set, then, at each subsequent iteration, it reassigns weights to the misclassified observations. The weights of the models are updated by computing the negative gradient of the loss function with respect to the model's prediction, which is then used as a target for the next model.
This was the best-performing model, achieving an R-squared of 0.9841 and an RMSE of 0.4569.

The best hyperparameters were found through GridsearchCV and are as follows:

- learning_rate= 0.01,
- max_depth = 10,
- n_estimators = 1000,
- subsample = 0.5

XGBoost Regressor

The XGBoost Regressor is an advanced version of the Gradient Boosting Regressor that incorporates regularization techniques, such as limiting the depth of the decision trees and introducing penalties for large weights in the model. Additionally, it uses a more recent algorithm to optimize the data splitting at each node of the decision tree, making it faster and more accurate than traditional gradient boosting algorithms.
Despite the advanced features, this model surprisingly performed the worst among the ensemble ones, probably because the parameters grid for the GridSearchCV was not good enough. This model reaches an R2 of 0.9769 and an RMSE of 4.1866.

The best hyperparameters were found through GridsearchCV and include:

- colsample_bytree = 0.5
- gamma = 0.0

- learning_rate = 0.1
- max_depth = 8
- min_child_weight = 5

Neural Network

Lastly, a Neural Network was implemented with a structure that comprised of 5 layers, consisting of 11 neurons (which represents the number of predictors used in the model), followed by 24, 12, 6, and 1 neurons respectively in each subsequent layer. Due to computational limitations, it was only possible to set the number of epochs to 10,000 to train each Neural Network, which may be considered a limitation of their potential. It is acknowledged that the model may have performed better with a higher number of epochs.

This model achieved satisfactory results, but not as good as expected, ranking fourth after the Ensemble methods, with an R-squared of 0.9525 and an RMSE of 5.9461.

Then the Optuna library was used to determine the optimal number of layers, neurons, and learning rate. However, this method did not meet the expectations, and the Neural Network built using generic parameters performed better.

Results

Insights on the three independent datasets

Through the implementation of all the regression models on the three independent datasets and to the specific feature importance chart, it was possible to deduce that the minor variables, previously mentioned, that were not common to all three datasets had close to zero contribution to the improvement of the models. Therefore, it was decided to drop them and conclude the analysis with the merged dataset.

Final insights

As it is possible to visualise in the table below, the best performing model is the Gradient Boosting Regressor, achieving a R-squared value of 0.9841 and an RMSE of 3.4569.

Model	**R2 score**	**Std R2**	**RMSE**
Linear regression	0.4091	0.0093	21.0522
SVR	0.9096	0.0047	8.2334
Random Forest	0.9814	0.0016	3.7309
XGBoost	0.9769	0.0009	4.1866
Neural Network	0.9525	0.0053	5.9461
Gradient Boosting	0.9841	0.0004	3.4569

Table 1 – Evaluation Metrics

To interpret the model results from a financial point of view, the Python library "SHAP" proved to be an extremely useful tool. This library made it easy to visualize the importance of predictor features in various types of visualizations, as shown in the Appendix A.

The SHAP value of a specific feature is a measure of its contribution to the model output, representing the difference between the predicted output when that feature is included versus when it is excluded, averaged over all possible combinations of features. It is important to note that the scale of SHAP values is dependent on the scale of the input features. If the input features are standardized to have zero mean and unit variance, then the SHAP values will be on the same scale. However, if the input features are not standardized, then the SHAP values will be in the units of the input features.

It is worth noting that the last Shap graph implemented is the one used to infer financial insights.

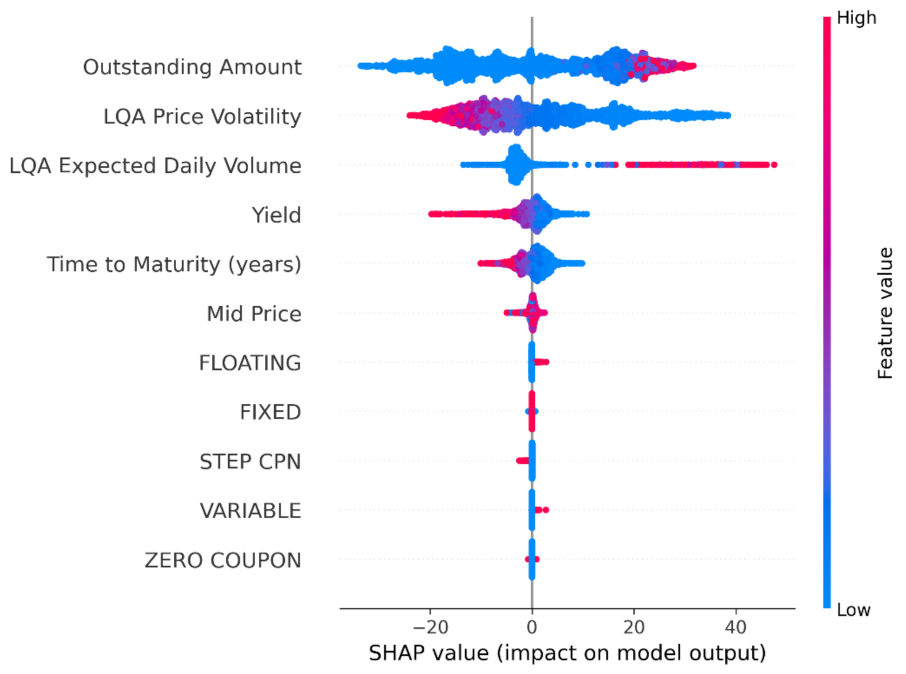


Fig. 4 Shap Values

The graph shows the predictors with the highest contribution in descending order from top to bottom. For each predictor, the distribution of SHAP values over the predictions and the individual feature's contribution to the prediction is shown. Each data point is coloured based on the value of the specific predictor. This representation allows to understand the extent to which the values of each predictor affect the final Liquidity Score prediction.

For **Outstanding Amount** the graph shows that lower values of this variable correspond to a decrease in Liquidity Score, and vice versa. Outstanding Amount represents the total amount of bonds issued by the issuing company that are still held by investors and must be returned to them upon maturity or early redemption. A corporate bond with a higher outstanding amount is generally more liquid, as it offers a greater availability of bonds in the market, allowing investors to buy or sell the bond more easily. Therefore, the insights from the model are consistent with the actual workings of the market, as a larger outstanding amount implies a greater availability of bonds in the market, making it easier for investors to buy or sell them.

LQA Price Volatility showed an inverse correlation with the outcome variable. High levels of price volatility may indicate lower liquidity, as rapid price fluctuations make it more challenging to buy or sell the bond at a fair price and this is reflected also in the graph. This is because bond price volatility refers to the fluctuations of the bond price in the market.

Regarding **LQA Expected Daily volume**, this score indicates the expected daily trading volume for that particular bond, and it is calculated on historical trading data and the bond's liquidity in the market. A higher expected daily trading volume may then indicate higher liquidity and vice versa. As we can see this is also captured perfectly by the graph.

Yield is another important predictor for the liquidity score. A corporate bond with a higher yield may be less liquid, as it may be associated with greater risk and may be less attractive to investors. The Shap graph also indicates that high values of Yield have a strongly negative impact on the Liquidity Score, while low values of Yield have a slightly positive impact.

For **Time to Maturity**, it is noteworthy to mention that it has an intuitive relationship with the outcome variable. Specifically, the longer the time remaining until the bond matures, the higher the bond's volatility and interest rate risk. This happens because bonds with longer maturities tend to be more sensitive to changes in interest rates, which in turn increases their volatility. In contrast, bonds with shorter maturities tend to be more easily exchanged, which makes them more liquid.

For **Mid Price**, the SHAP values suggest that it has a relatively low predictive power for the Liquidity Score. However, from a financial point of view, a Mid price closer to the current market value may indicate greater liquidity, as trades can be executed at prices closer to the current market value.

Finally, regarding the **Bond Types**, the model showed that it has very low predictive power. However, it is important to note that this may be due to an unbalanced distribution of classes in the training data, where almost 90% of the bonds used for training are fixed. Therefore, it is reasonable to believe that a more balanced dataset that includes a variety of bond types will be more useful for the model to understand how to deal with different types of bonds and improve the predictive power of this predictor.

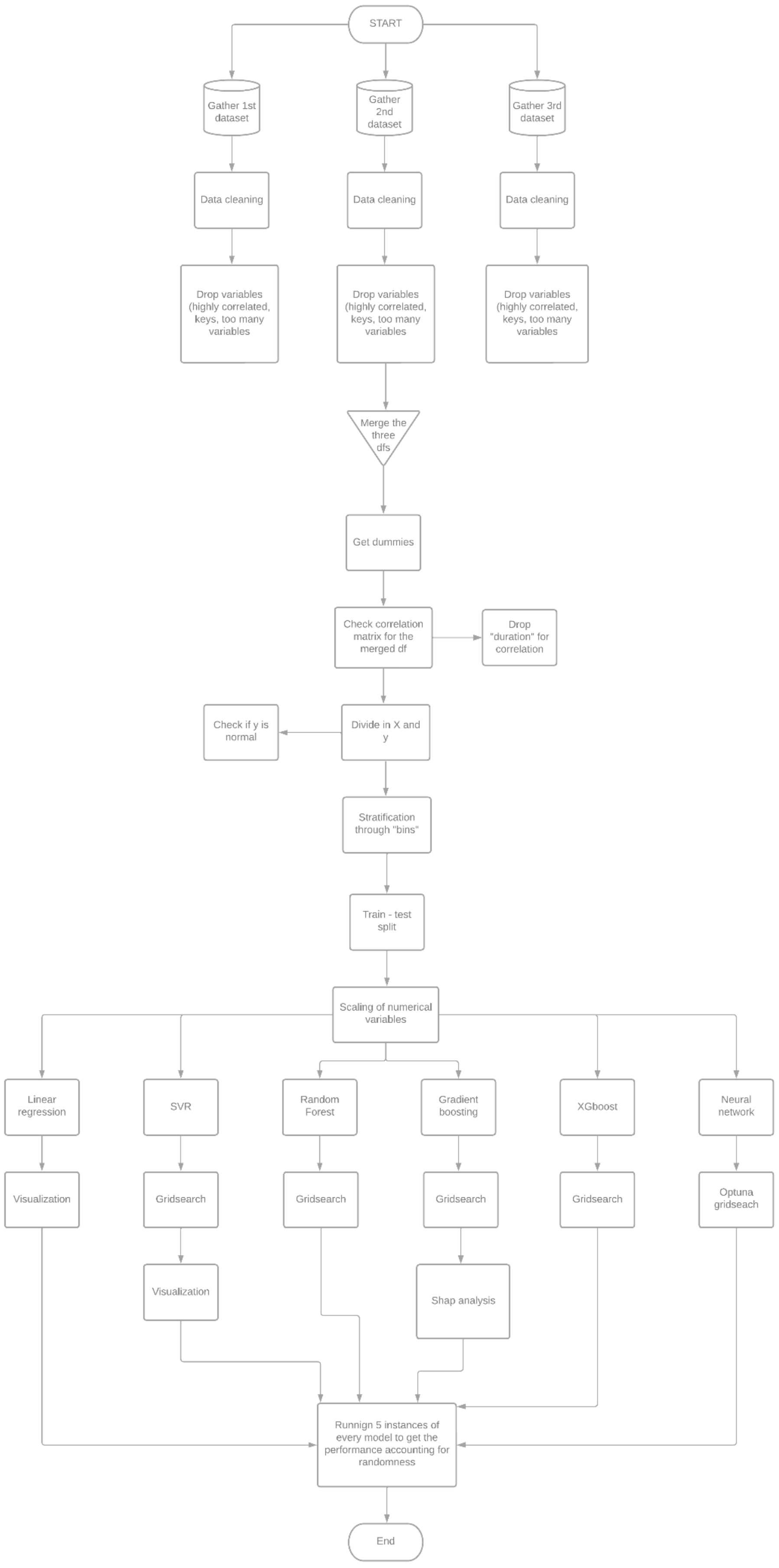
Conclusions

In the end, as one could expect, the prediction of the Liquidity Score cannot just be reduced to just a series of linear relationships, so it was necessary to develop much more complex models able to capture underneath interdependencies. The best model for estimating the Bloomberg Corporate Bond Liquidity Score was the Gradient Boosting Regressor, which thanks to parameter tuning performed with a GridSearchCV was able to achieve an R-squared of 0.9841.

Regarding the future developments it is rather important to mention that the variables "S&P Ratings" and "ESG Disclosure" were initially considered for evaluating a corporate bond's liquidity but were then drop due to insufficient data. However, including these non-financial variables could be useful in assessing the bond issuer's sustainability, reputation, and risk management practices and maybe increase the accuracy of the model.

The ESG variable could be significant in estimating the corporate bond liquidity score since ESG disclosure can influence investor perception of the issuer's sustainability and risk management practices. Moreover, ESG disclosure has become increasingly important in recent years as investors pay more attention to environmental, social, and governance issues as part of their investment risk and opportunity evaluation. Credit rating, such as "S&P Ratings," reflects the default risk associated with a bond issuer and is an objective evaluation of its creditworthiness. A higher credit rating indicates greater financial stability and a lower probability of bond default, potentially increasing the corporate bond's attractiveness to investors and improving its liquidity score. Therefore, credit rating is an important variable in estimating the corporate bond liquidity score, providing essential information on the issuer's credit quality and default probability.

Appendix A



START
Gather 1st dataset
Gather 2nd dataset
Gather 3rd dataset
Data cleaning
Data cleaning
Data cleaning
Drop variables (highly correlated, keys, too many variables
Drop variables (highly correlated, keys, too many variables
Drop variables (highly correlated, keys, too many variables
Merge the three dfs
Get dummies
Check correlation matrix for the merged df
Drop "duration" for correlation
Check if y is normal
Divide in X and y
Stratification through "bins"
Train - test split
Scaling of numerical variables
Linear regression
SVR
Random Forest
Gradient boosting
XGboost
Neural network
Visualization
Gridsearch
Gridsearch
Gridsearch
Gridsearch
Optuna gridseach
Visualization
Shap analysis
Runnign 5 instances of every model to get the performance accounting for randomness
End

Table 2 – Flow Chart of code process

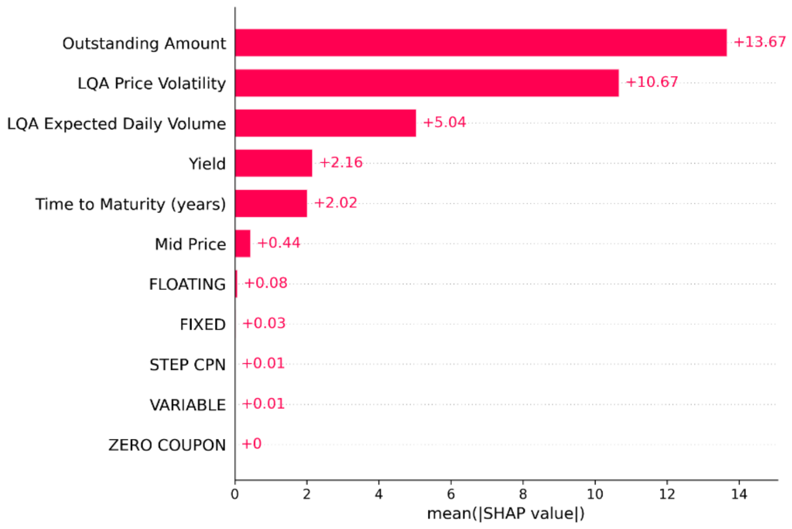


Fig. 5 Feature importance of each predictor

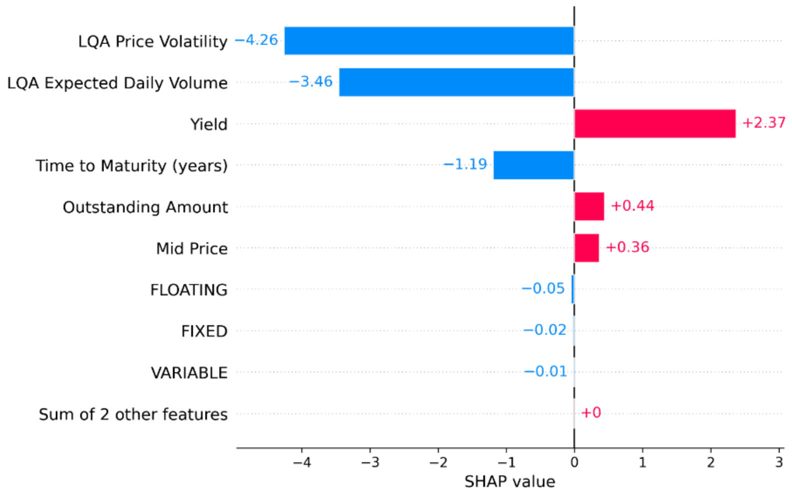


Fig. 6 Shap Feature contribution of most important predictors for a singular random observation

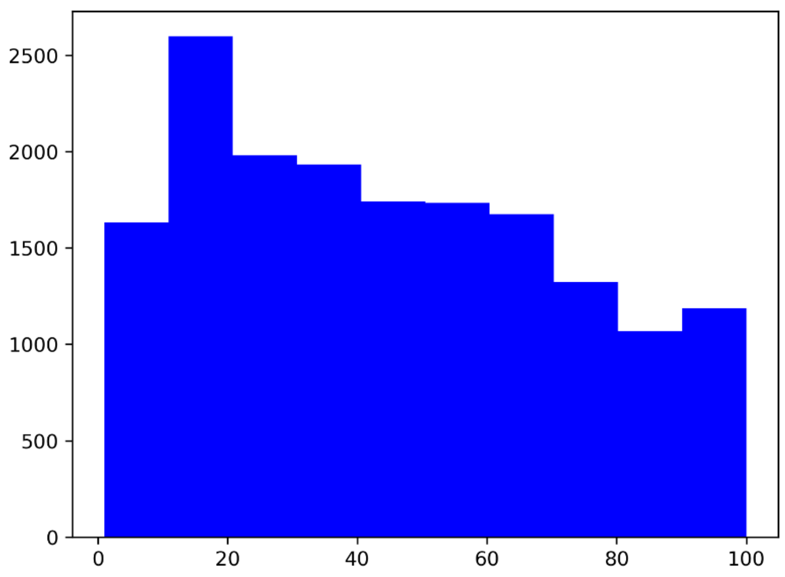


Fig. 7 Liquidity Score distribution

Appendix B

Contributions

All the members of the team gave the same amount of help in the realisation of the project. However, Leonardo Bosco is undoubtedly the master mind behind the realisation of the models, Greta Kichelmacher the one who made possible the financial insights through extensive investigations, and Emanuele Carosi the glue among the different steps of the project supporting both fields.