Machine Learning for Corporate Credit Rating Prediction

Data Scientist Udacity Nanodegree: Capstone Project

11th March 2023

Beth Barlow

1. Introduction

1.1 Project Overview

The prediction of corporate credit rating is one of the most sought-after applications of machine learning in finance. Credit ratings assess the creditworthiness of debt issuers and gauge the risk associated with purchasing debt. The ability to predict such ratings is of interest to several market participants.

Firstly, employing domain experts to manually perform ratings analysis can be costly and time-consuming. The ability to gain an in-house estimate enables banks and other lenders to set strategies to manage credit risk and minimize loss. Secondly, issuers themselves may be interested in a preliminary estimate of credit rating before entering capital markets. Finally, the ability to forecast credit ratings based on publicly available financial datasets improves the transparency of the rating process and may reduce conflicts of interest between the rated corporation and the credit rating agency, something widely regarded as an underlying driver of the 2008 financial crisis.

Traditional statistical methods such as linear regression have been applied to this problem, however, are difficult to implement due to complex dependencies between financial data used to predict the final rating.

1.2 Problem Statement

The purpose of this work is to build a simple machine learning prediction model to predict the credit rating of a company based on several company financial indicators that may impact the credit rating assigned by a ratings agency. This will be achieved by developing and testing a number of popular machine-learning models and comparing the performance of each. The hyperparameters of the best performing model will be tuned to try to improve the final performance.

The problem will be addressed in Python using multiple python libraries such as pandas for data wrangling, sci-kit learn to implement the model and Seaborn and Matplotlib for visualisation.

1.3 Metrics

The F1-score is an appropriate single number metric to evaluate performance of a machine learning model in this context. As the harmonic mean of precision and recall, the F1-score provides robust results for imbalanced datasets, and places equal importance on reducing false positives and false negatives. This is important for the prediction of credit ratings since ratings are essential in determining interest rates on debt instruments. Whether over- or under-estimated, inaccurate ratings could lead to distorted prices of debt instruments and ultimately disruption to financial markets.

The F1-score can be defined as follows:

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

Where:

TP = number of true positives
FP = number of false positives
FN = number of false negatives

This is an unbalanced multiclass classification problem, and the class F1-scores can be averaged by using the number of instances in a class as weights. This was achieved by setting the 'average' argument equal to 'weighted' in the Python sci-kit learn library.

2. Methodology and Results

2.1 Exploratory Data Analysis

The dataset is publicly available from Kaggle and contains records of credit ratings awarded to 2029 companies. Each row represents a company, and a number of key financial indicators are represented by the columns. The dataset contains the following columns, with the data type shown in brackets:

- 'Rating': company credit rating (string)
- 'Name': company name (string)
- 'Symbol: 'company name symbol' (string)
- 'Rating Agency Name': rating agency (string)
- 'Date': date of credit rating (datetime)
- 'Sector': company sector (string)
- 'currentRatio': current ratio (float)
- 'quickRatio': quick ratio (float)
- 'cashRatio': cash ratio (float)
- 'daysOfSalesOutstanding': number of days sales outstanding (float)
- 'netProfitMargin': net profit margin (float)

- ‘pretaxProfitMargin’: pre-tax profit margin (float)
- ‘grossProfitMargin’: gross profit margin (float)
- ‘operatingProfitMargin’: operating profit margin (float)
- ‘returnOnAssets’: return on assets (ROA) (float)
- ‘returnOnCapitalEmployed’: return on capital employed (float)
- ‘returnOnEquity’: return on equity (ROE) (float)
- ‘assetTurnover’: asset turnover (float)
- ‘fixedAssetTurnover’: fixed asset turnover (float)
- ‘debtEquityRatio’: debt-equity ratio (float)
- ‘debtRatio’: debt ratio (float)
- ‘effectiveTaxRate’: effective tax rate (float)
- ‘freeCashFlowOperatingCashFlowRatio’: ratio of free cash flow to operating cash flow (float)
- ‘freeCashFlowPerShare’: free cash flow per share (float)
- ‘cashPerShare’: cash per share (float)
- ‘companyEquityMultiplier’: equity multiplier (float)
- ‘ebitPerRevenue’: EBIT per revenue (float)
- ‘enterpriseValueMultiple’: enterprise value multiple (float)
- ‘operatingCashFlowPerShare’: operating cash flow per share (float)
- ‘operatingCashFlowSalesRatio’: ratio of operating cash flow to sales (float)
- ‘payablesTurnover’: payables turnover (float)

The aim is to use the financial data for each individual company to predict the credit rating given in the ‘Rating’ column.

Exploratory data analysis (EDA) is important to understand key characteristics of features in the data as well as relationships between features. It helps to uncover areas of the dataset that require cleaning prior to analysis and can be used to identify unexpected characteristics of the dataset such as outliers and missing values.

The features were categorized as numerical or categorical based on data type. Categorical features included the rating agency name, company name and symbol, and company sector. The name and symbol for each company clearly do not hold predictive power and were identified as variables to remove. Rating agency name and company sector were identified as features to be one-hot encoded. Numerical features included all financial data as well as ‘Date’. 'Rating' is the target variable and was identified as a variable to be label-encoded to ensure that sci-kit learn algorithms can be applied.

The distribution of each numerical variable was displayed, which revealed that most variables had extremely high maximum values and likely contained outliers. This required further investigation as outliers in financial data may be legitimate and add valuable information to the model.

There were no missing values in the dataset, and as such, imputation was not required.

Data visualization can be achieved in Python using packages such as Pandas, NumPy, Seaborn and Matplotlib.

Figure 1 shows how ratings are distributed across companies. The most common ratings are those around the middle of the range, with a BBB rating awarded to over 650 companies (around one-third of companies in the dataset). The dataset also shows a high class-imbalance, something that should be dealt with prior to model training to avoid poor predictive performance for the minority classes. This can be partially addressed by creating a new class that combines ratings of CCC, and also combining ratings 'AA' and 'AAA' into a new class. Resampling techniques were also investigated, and further discussion is given in Section 2.3.

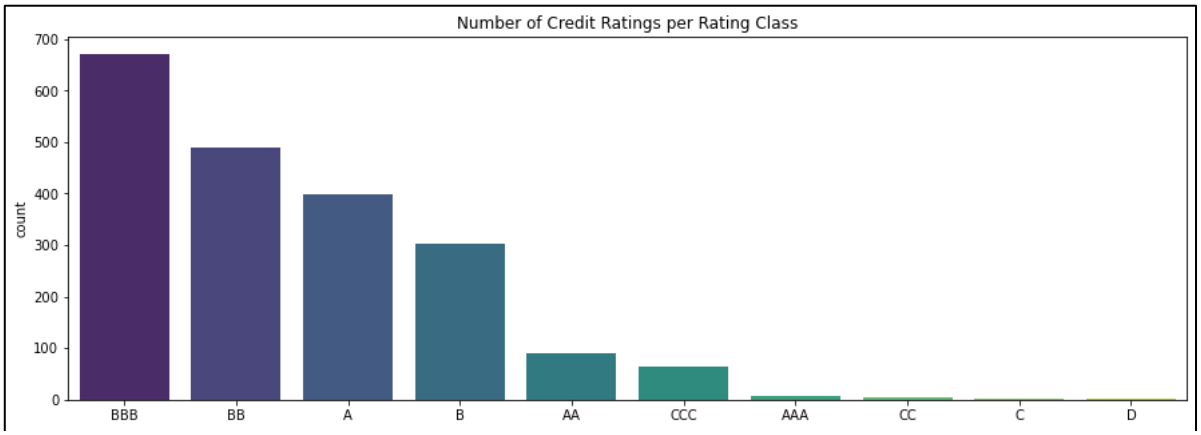

Figure 1: Number of Credit Ratings Per Rating Class

Figure 2 shows differences in the distribution of credit ratings between companies. Whilst Fitch Ratings and Moody's Investor Services assign credit ratings of BBB to over 40% of companies rated, Egan-Jones and S&P display more balanced distributions. These observations indicate potential predictive power of this feature, however, could also be attributed to nuances of the particular dataset.

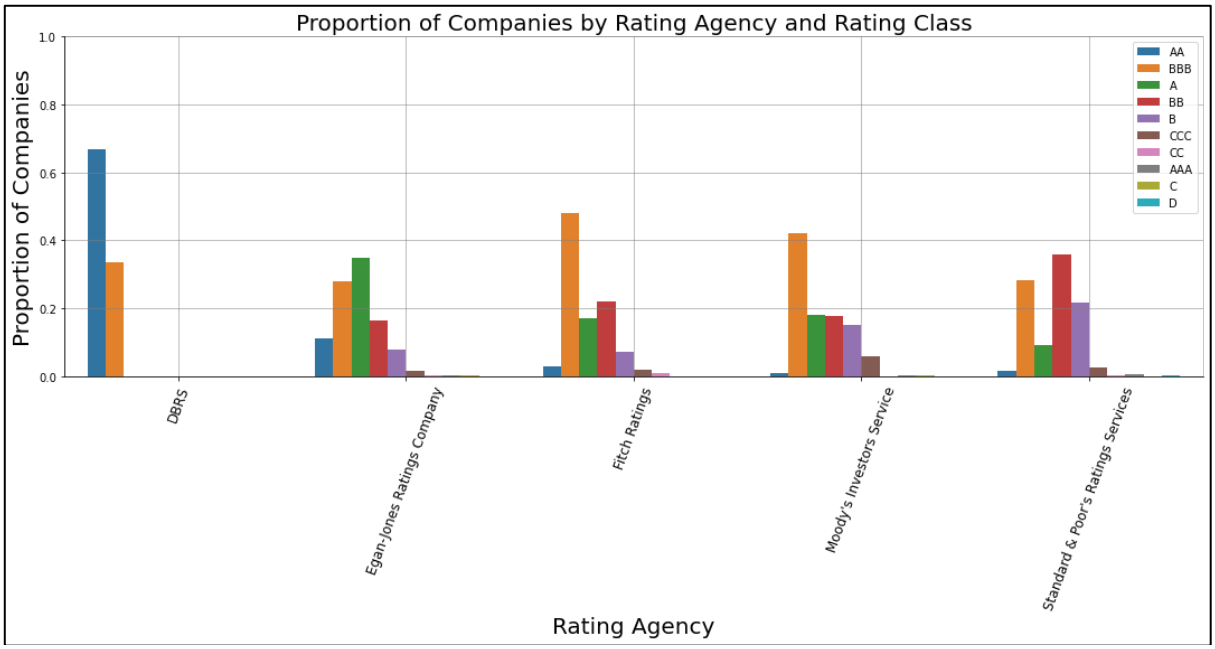

Figure 2: Comparison between Rating Agencies of Distributions of Rating Classes

The number of categories in 'Rating Agency Name' and Sector' was investigated to ensure that this would not add too many new features when one-hot encoding is applied. The results are displayed in Figures 3 and 4 below. Too many categories would create a sparse feature matrix, something that could lead to overfitting and poor model performance. It was noted that the Rating Agency DBRS rated very few companies. However, other features associated with these companies may still provide valuable information to the model. Companies rated by DBRS were therefore retained in the model.

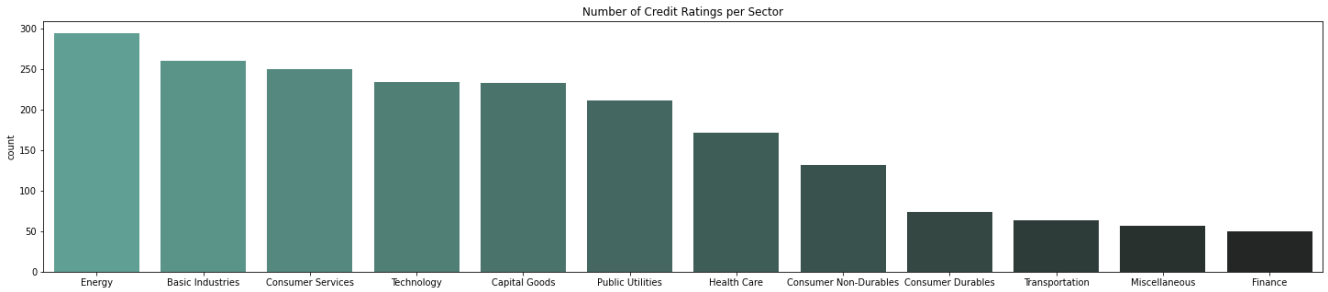

Figure 3: Number of Credit Ratings per Sector

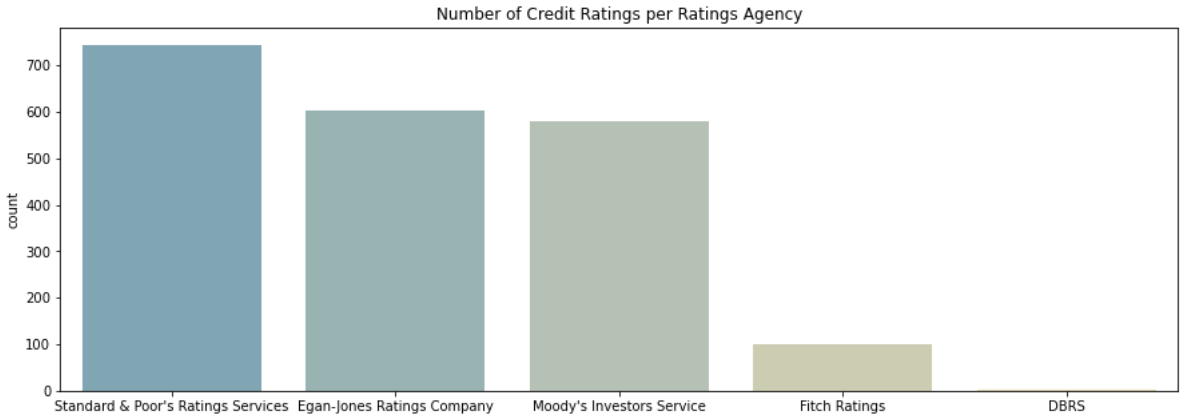

Figure 4: Number of Credit Ratings per Ratings Agency

The extent to which the date is related to the rating was also of interest. Figure 5 shows the change in distribution of ratings across ratings class on each day of the month.

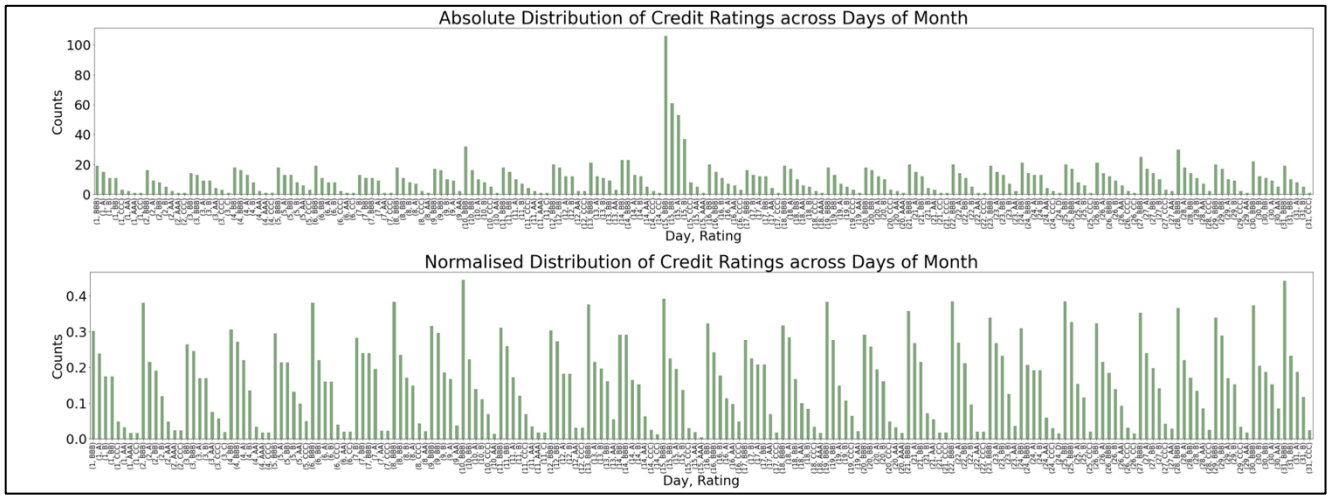

Figure 5: Daily Distribution of Credit Ratings: Comparison across Days of Month

The distribution of ratings across companies does not appear to change significantly depending on the day of the month, an expected observation. Although a larger number of credit ratings are assigned mid-month (15th), the distribution of ratings on this day is not significantly different to other days, so 'Day' should not have a significant impact on the model. Nevertheless, it will be retained, and its predictive power investigated.

Figure 6 shows the change in distribution of ratings across ratings class each month.

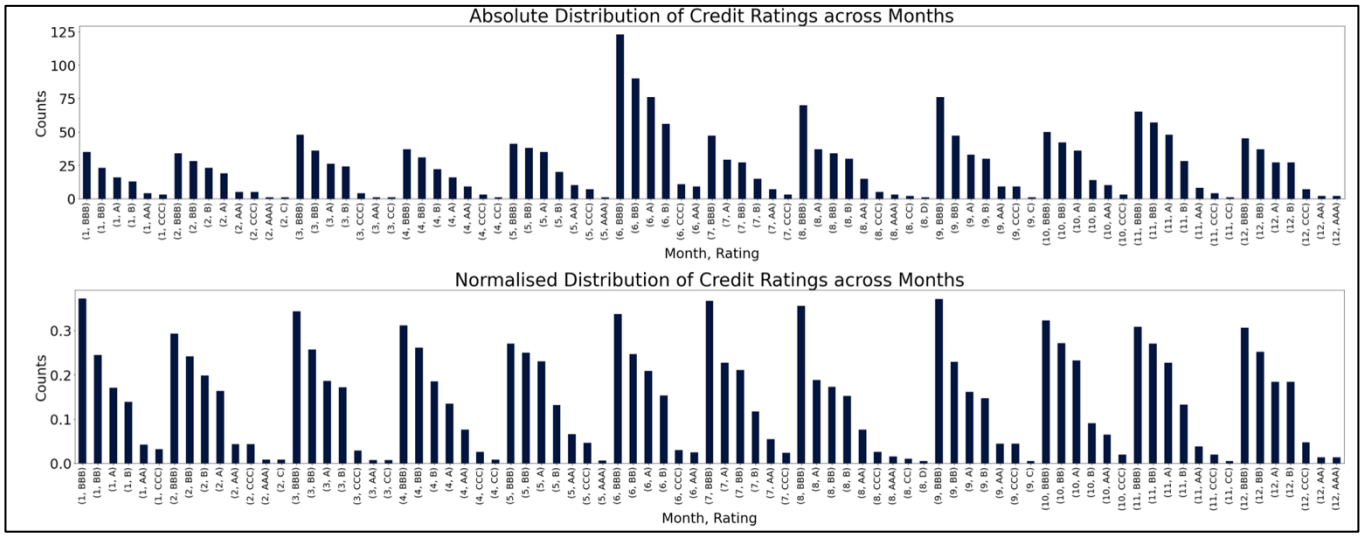

Figure 6: Monthly Distribution of Credit Ratings: Comparison across Months

A similar result to Figure 5 is found, with a larger number of ratings made in June relative to other months, yet with the normalised distribution similar to other months. The overall distribution appears slightly more skewed towards BBB ratings in the summer months, but this may simply be due to the nuances of the given dataset and may not indicate a pattern across a broader population.

It was noted that 'Month' and 'Day' are cyclical features and should be encoded to reflect this. 'Year' should be removed from the dataset. This feature may hold significant predictive power on a backwards-looking model; however, the purpose of this model is to predict future credit ratings, and as such, the model performance may be lower in practice.

The distribution of numerical variables was investigated via box and whisker plots, the first five of which are displayed in Figure 7.

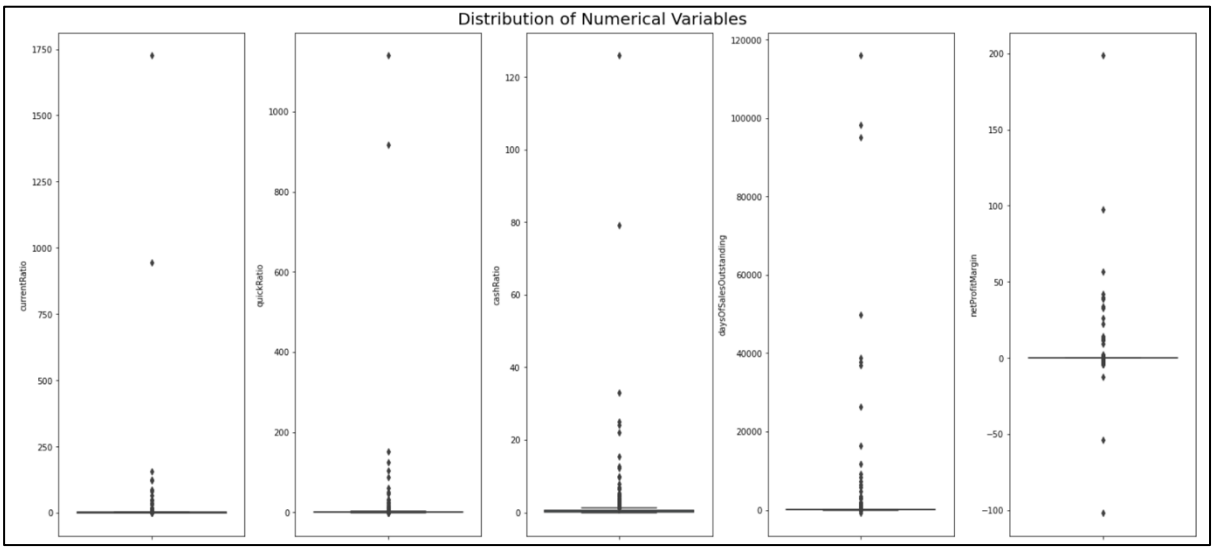

Figure 7: Box and Whisker Plots showing Distribution of Numerical Variables

It is clear that outliers are a generalized problem. However, in a financial context, these may be legitimate as long as such data points are associated with the same company. This was investigated by checking that outliers were consistent across similar financial metrics for a given company. As a result, no data points were dropped in the dataset used for modelling. However, the presence of a small number of outliers greatly reduces the ability to visualise the overall distribution of the variables.

To address this, a winzorized dataset was created whereby outliers were defined as points lying outwith 1.5 times the interquartile range. This dataset was used to re-plot the distribution of numerical variables, this time according to Rating as shown in Figure 8.

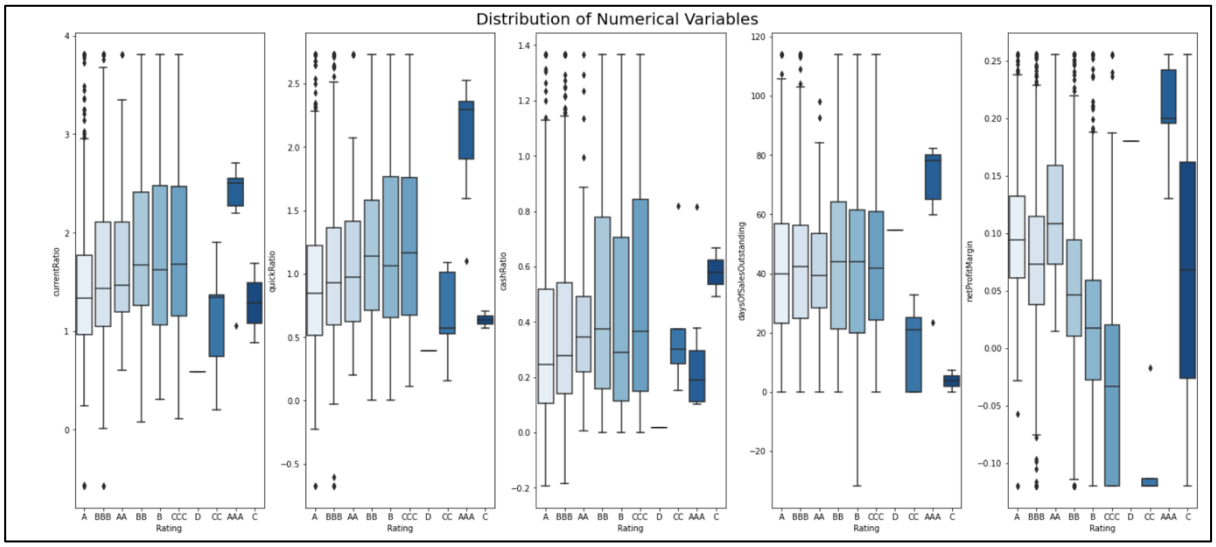

Figure 8: Box and Whisker Plots showing Distribution of Numerical Variables

This shows a difference in medians according to the rating (i.e. solvency risk of the company), which indicates that these numerical variables may hold predictive power in a classification model.

The correlation between features can be seen by plotting a correlation plot as displayed in Figure 9.

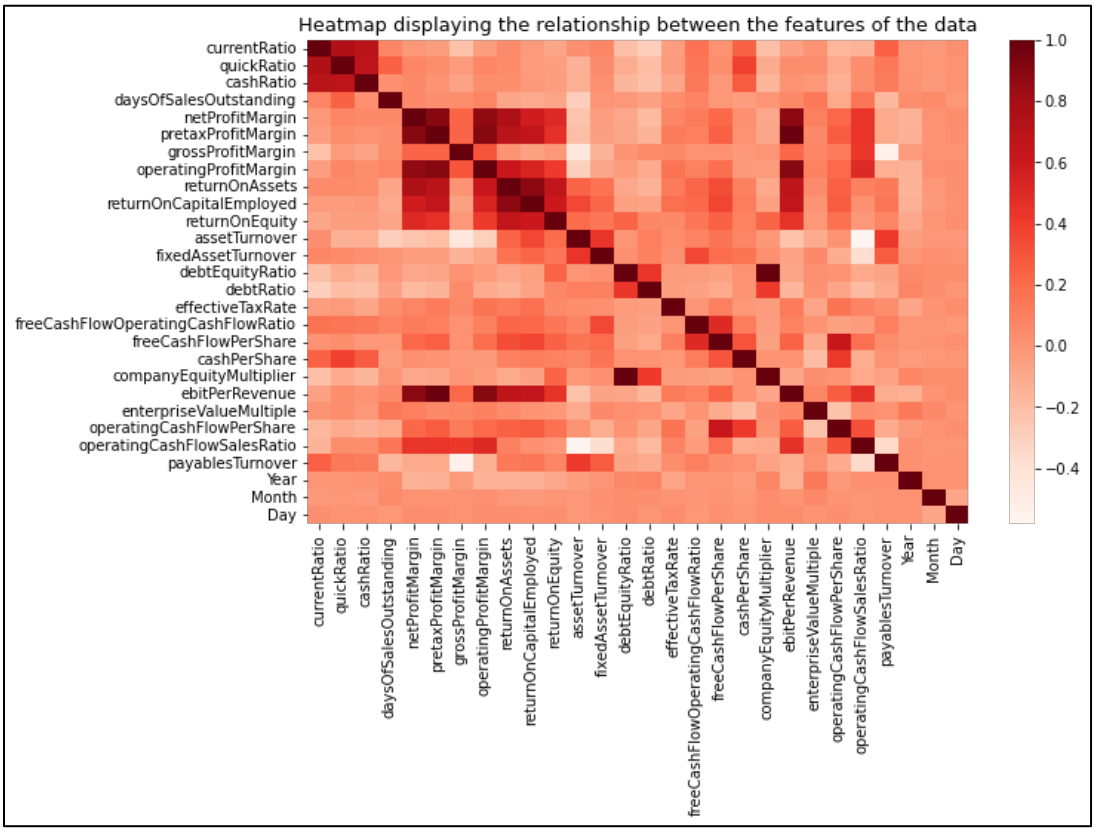

Figure 9: Heatmap showing correlation between numerical features

There is a near perfect correlation between several variables for which a direct relationship exists. For example, the company equity multiplier is equal to 1 + debt equity ratio and these two variables exhibit a correlation of 0.994. Whilst algorithms based on boosted trees are immune to multicollinearity by nature, KNN and Naïve Bayes cannot handle highly correlated features. In the case of the former, correlated features will carry a higher weight on the distance calculation. For the latter, the fundamental assumption is that features are independent, thus correlated features violate this assumption. As such, one variable in a pair of highly correlated variables must be removed from the dataset. An upper correlation limit of 0.8 was set to decide which variables should remain in the final dataset.

2.2 Data Preprocessing

In this step, undesirable characteristics of the data that were identified through EDA were addressed.

A number of steps were undertaken:

1. Replace 'Date' with individual features for 'Month' and 'Day' and encode these to ensure the algorithm understands that these features occur in cycles
2. Remove 'Symbol' and 'Name' – this data will have negligible predictive power
3. Address high correlations by removing 'companyEquityMultiplier', 'pretaxProfitMargin', 'returnOnCapitalEmployed', 'ebitPerRevenue' and 'quickRatio'
4. Reclassify 'Rating' by grouping ratings of CCC and below, and ratings of AA and AAA
5. Apply label encoding to the target variable 'Rating'
6. Apply one-hot encoding to 'Sector' and 'Rating Agency Name'

2.3 Model Implementation and Refinement

The dataset was split into train, validation, and test sets. A 70%–20%-10% split was chosen, as the dataset is relatively small, and it is important to maximise the proportion of data used for training. The argument 'stratify = y' was included to ensure an equal proportion of classes in each of train and test set, and 'shuffle' was set to True to reduce overfitting. The training set is used to train the model, with validation set used for hyperparameter tuning and test set to evaluate the performance of the final model.

The task of predicting a credit rating for a new company translates into a multiclass classification problem. The 'Rating' variable is the target, stating which class of rating is assigned to each company. Popular algorithms for this type of problem include K Nearest Neighbours (KNN), Naïve Bayes and Decision Tree methods.

Prior to fitting these models, the data is scaled using MinMaxScaler() from sci-kit learn. This scales the data to between 0 and 1 whilst retaining the original shape of the distribution. Scaling is required to avoid bias towards features with higher magnitudes in distance-based algorithms. The technique is also useful for gradient descent-based models such as logistic

regression, as it accelerates convergence to the global minima and ensures a consistent step size between features. Classifiers based on graphical models such as Random Forest, XGBoost and Naïve Bayes are invariant to the scale of features, so scaling should not affect the model performance either way.

Each of the following models were fitted to the train set, using the default parameters for each. The approach of 'fail fast and iterate' was taken, so no tuning was attempted at this stage. The models fitted are listed below. It is assumed that the reader has a basic understanding of how these models work and they will not be discussed in detail.

- Multinomial Logistic Regression
- Naïve Bayes
- KNN
- Random Forest
- XGBoost

The fitted models were applied to both training and validation set, and the resulting F1-scores are shown in Figure 10.

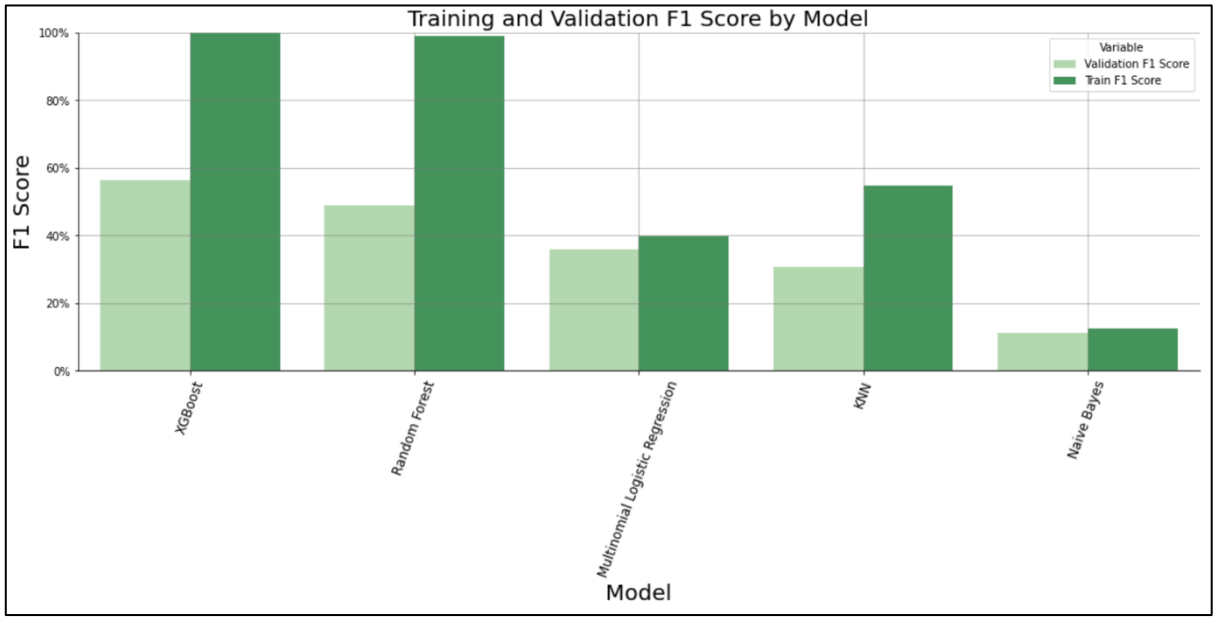

Figure 10: Comparison of F1-scores for baseline models

An ideal result is that the model displays low bias and low variance. The F1-scores of the training and validation dataset should be high, and close to each other, indicating that the model is both fitting well, and generalizing well to unseen data.

Naïve Bayes performed extremely poorly with a validation F1-score around 7%. However, a key assumption underlying the algorithm is that predictors are independent, conditional on class. Given the very high correlations observed in this dataset, this assumption is not met, and a poor performance is not unexpected.

Poor performance of KNN and the logistic regression model can be attributed to the inability to handle outliers and collinearity. Furthermore, logistic regression relies on a fundamental assumption of a linear relationship between the target and features, which may not hold for this dataset.

The fitted Multinomial logistic regression and Naïve Bayes models are typical of underfitting models, with high bias and low variance. This contrasts with the two best performing models, XGBoost and Random Forest. Both algorithms exhibited a training F1-score of almost 1, with validation F1-scores of 0.59 and 0.5 respectively. This behaviour is strong indication that the models have high variance, i.e., are greatly overfitting and cannot generalize well.

There are several possible remedies. Ideally, a larger and more diverse dataset would be collected. However, for the given dataset, resampling and data augmentation techniques could be performed to address the class imbalance. Furthermore, it is highly likely that most default hyperparameter values are inappropriate. For example, high values for the 'subsample' and 'colsample_bytree' parameters (i.e., fraction of training rows and columns used to train each tree) can produce an excessively complex model. Similarly, a high learning rate updates the prediction more with each successive tree, leading to a model that is highly tuned to the training set.

This can be addressed by *hyperparameter tuning,* and the following discussion will detail the results of this method when applied to the best performing model, XGBoost.

Optuna is a popular method for identifying the optimal set of hyperparameters. The machine evaluates the performance for each combination and returns the trial with the best performance. By using Bayesian optimization to find the best set of hyperparameters, it is much faster than the exhaustive GridSearchCV method.

In this project, a particular area of interest was the comparison of resampling techniques and the hyperparameter tuning procedure discussed shortly was repeated for three techniques: random oversampling, random undersampling and synthetic minority oversampling technique (SMOTE). These were compared to a base scenario that performed no resampling.

Random oversampling was performed using RandomOverSampler() from sci-kit learn. This involves randomly duplicating samples from the minority classes. Random undersampling involves randomly removing samples from the majority classes and was performed using RandomUnderSampler() from sci-kit learn. Both processes should be repeated until the desired class balance is achieved, in this case equal split across classes. Finally, SMOTE is a data augmentation technique that oversamples the minority classes by creating synthetic data points. In contrast to RandomOverSampler(), this approach is effective as it creates new plausible data points that add new information to the model.

For each of these resampling techniques, hyperparameter tuning was undertaken by first defining an objective function. This function trains the model on a training set and applies the trained model to a validation set, returning a performance evaluation metric, in this case F1-score.

Within the objective function, a hyperparameter space was defined, which was searched over to identify the optimal hyperparameter set. The size of search range for each parameter was chosen to strike a balance between computational efficiency and maximising possibility of finding optimal parameter value.

Stratified 5-fold cross validation was applied in training and validation. Given this model is prone to overfitting, a smaller number of folds was chosen to reduce variance error, as well as reduce computation time and memory required. The data was split in such a way to ensure that each fold has the same proportion of observations of a given class. It is critical to ensure that information is not leaked whilst carrying out re-sampling techniques alongside cross validation. Resampling should not be conducted before cross validation as duplicated data may appear in the validation and training set at the same time. Therefore, the training data was resampled in each fold of cross validation after splitting the training and validation set. Furthermore, resampling was not applied to the validation set as this represents real data for which the imbalance always exists.

The final validation F1-score returned by the objective function is the mean of the individual scores for each cross-validation fold. Each individual F1-score is itself a weighted average of the F1-score for each class.

The objective function was optimized over 50 trials to produce 50 sets of hyperparameters and associated F1-scores. Figure 11 displays the optimization history for the best performing resampling technique, RandomOverSampler(). This is useful to ensure that a sufficient number of trials have been run to reach convergence.

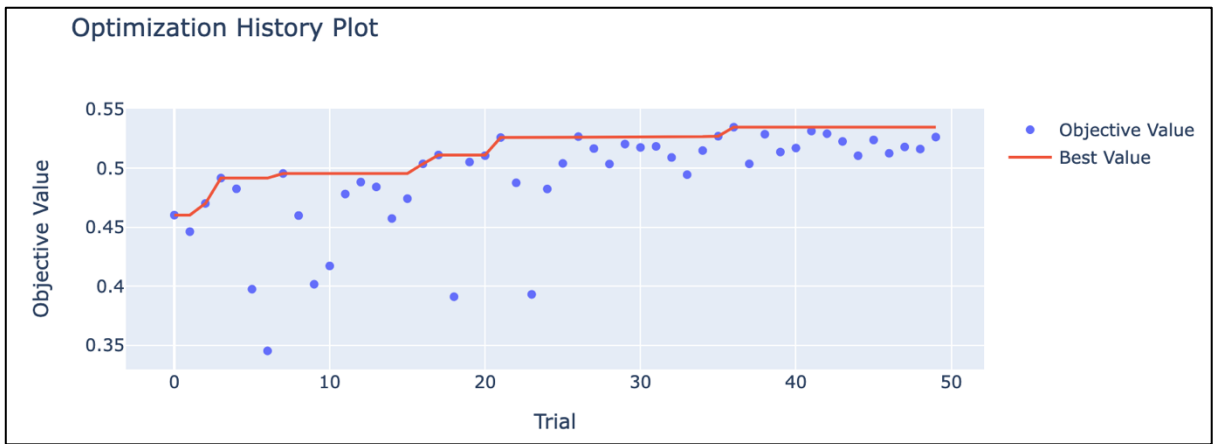

Figure 11: Optimisation history plot for XGBoost trained with randomly oversampled data

The final model hyperparameters for each resampling technique are shown in Table 1. The results for RandomOverSampler() that produced the highest F1-score are highlighted in green.

Sampler	Trial	F1-score	Parameters
None	21	0.5311	max_depth: 8 n_estimators: 400 eta: 0.09860320469273205 gamma: 0.038378995161902935 subsample: 0.6673019646162675 colsample_bytree: 0.8749660641256987 reg_alpha: 0.6123959265237534 reg_lambda: 0.20303066320725047
RandomUnderSampler	9	0.3666	max_depth: 7 n_estimators: 550 eta: 0.012597327632011712 gamma: 0.1350268210205262 subsample: 0.953923226905159 colsample_bytree: 0.2793331656089445 reg_alpha: 0.4175342758614018 reg_lambda: 0.2980421193494581
RandomOverSampler	36	0.5348	max_depth: 7 n_estimators: 300 eta: 0.07211077129241486 gamma: 0.06462288809847885 subsample: 0.7907459887880808 colsample_bytree: 0.6143382176315698 reg_alpha: 0.32362436244976384 reg_lambda: 0.7309082608142007
SMOTE	21	0.5189	max_depth: 9 n_estimators: 450 eta: 0.11211624549947179 gamma: 0.3877969341459817 subsample: 0.6713436671804262 colsample_bytree: 0.5878689371510991 reg_alpha: 0.4774716553566396 reg_lambda: 0.1844873104163657

Table 1: Hyperparameter Tuning Results for Three Resampling Techniques and Baseline

The tuned hyperparameters control the level of conservatism of the model and can help to prevent overfitting.

The parameter *max_depth* controls the size of the decision tree and thereby the extent to which the algorithm learns the peculiarities of a given dataset. Too high a maximum depth represents a complex model that fails to generalize. The parameter *n_estimators* controls the number of trees to be boosted. It is expected that by increasing the depth of the tree, fewer trees are required to produce a similar model performance. In this model, *max_depth* and

n_estimators were increased to 7 and 300 from the default values of 6 and 100 respectively indicating that these parameters were tuned to increase model complexity.

Gamma defines the minimum loss reduction required to make a node split. Higher values lead to fewer splits and a shallower tree, thus a more conservative model. A default of 0 was increased to 0.06, which although lower than other models, still contributes to reduction of overfitting.

Eta controls the learning rate, i.e, the amount of correction applied at each step. An optimal *eta* of 0.07 is significantly below the default value of 0.3, as well as values chosen with other resampling methods. This indicates that the model is more robust to overfitting.

Subsample and *colsample_bytree* corresponds to the fraction of observations and features respectively to subsample at each step. The default fraction is 1. In this case, these parameter values were reduced to 0.79 and 0.61 indicating that the model is less likely to overfit to a particular sample or feature.

Alpha is the L1 regularization parameter on weights with a default of 0. The best performing model chose values of 0.32, a choice that increases the level of conservatism in the model.

Lambda is the L2 regularization parameter on weights with a default of 1. Increased values of these parameters make the model more conservative and reduce overfitting. The best performing model had a value of 0.73, indicating that the amount of regularization is reduced.

There is little difference between all three resampling techniques, with random oversampling leading to a slightly improved model by a small margin. However, this is useful to note as seemingly irrelevant differences between models can translate into huge losses by investors if an incorrect credit rating was predicted.

Figure 12 displays hyperparameter importances for the random oversampling model.

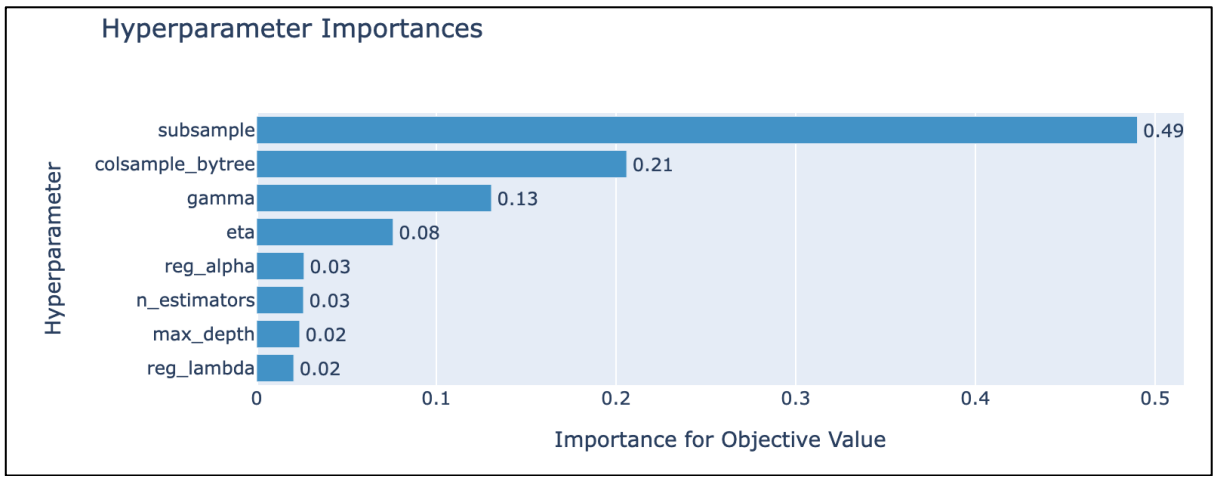

Figure 12: Hyperparameter importances for XGBoost trained with randomly oversampled data

The two parameters with the highest importance are 'colsample_bytree' and 'subsample', which together account for 70% of the overall importance. It is promising to see that the values of these parameters were tuned in favour of reducing overfitting.

2.4 Model Implementation and Refinement

The XGBoost classifier was retrained using random oversampling and the associated hyperparameter set, and the final model was applied to the test set, producing an overall weighted f1-score of 0.536. To evaluate the final model, a classification report and confusion matrix was produced. A classification report provides summary information on precision, recall and F1-score. This is useful to describe the type of errors made by the model but fails to provide specifics. Viewing a confusion matrix in conjunction with the classification report provides a more nuanced view of how the model performs and helps to avoid the common pitfalls associated with considering single number metrics in isolation.

The result of the final model is displayed in Table 2.

	precision	recall	f1-score	support
A	0.62	0.70	0.66	40.00
AAA/AA	0.43	0.30	0.35	10.00
B	0.34	0.33	0.34	30.00
BB	0.56	0.47	0.51	49.00
BBB	0.61	0.67	0.64	67.00
CCC and below	0.14	0.14	0.14	7.00
accuracy	0.54	0.54	0.54	0.54
macro avg	0.45	0.44	0.44	203.00
weighted avg	0.54	0.54	0.54	203.00

Table 2: Classification Report for XGBoost model with random oversampling and optimized hyperparameters

There is no clear pattern to how the model performs for different classes. For classes BBB and A, recall is higher than precision, indicating that false negatives are lower. On the other hand, precision is higher than recall for B-rated companies such that the model was careful to avoid incorrectly assigning this label. However, recall is much lower indicating that the model also misses B and BBB-rated companies as it is being too careful. The F1-scores reflects this imbalance. Minority classes 'AAA/AA' and 'CCC and below' show poor predictive performance due to the lack of training samples.

Figure 13 displays the related classification matrix.

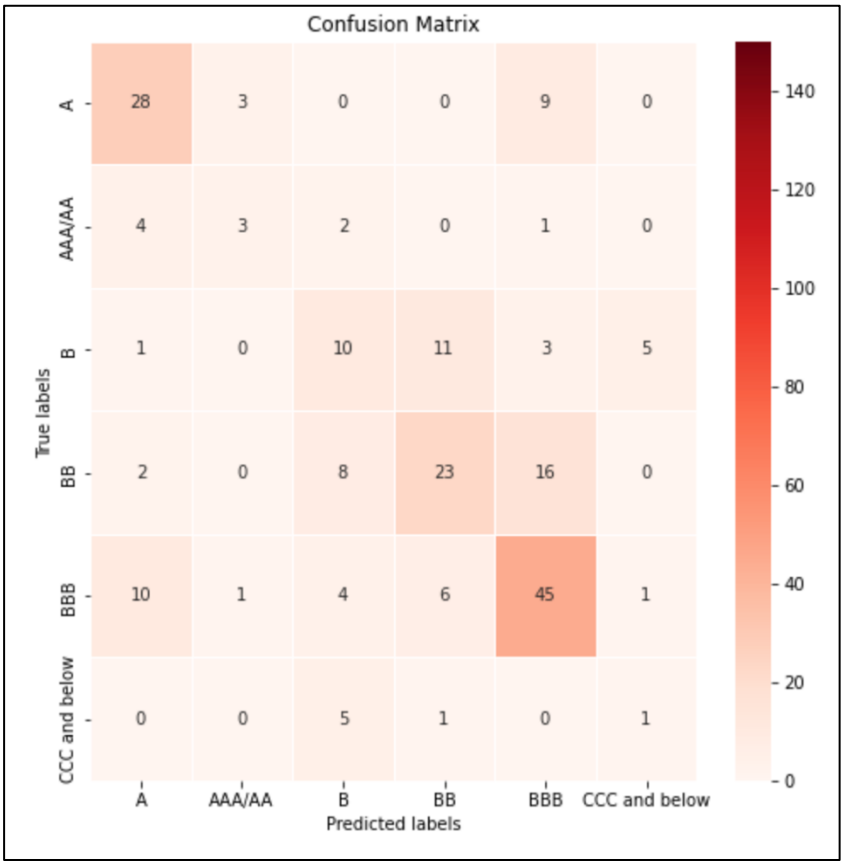

Figure 13: Confusion Matrix for XGBoost model with random oversampling and optimized hyperparameters

Apparent from the confusion matrix is the lack of separability of classes. For example, 6 BBB rated companies were rated BB, and 16 BB rated companies were rated BBB. Although inaccurate classification is detrimental in all cases, the second value here is of greater concern, since a credit rating of BB or below is considered non-investment grade, and the company at risk of default. Incorrectly attributing an investment grade rating to non-investment grade companies underestimates default risk and could lead to a loss of trust in the model, misguided pricing decisions and ultimately financial losses.

Overall weighted precision and recall for the model are identical, indicating that the model attempts to reduce false positives and false negatives in equal measure.

Figure 14 displays the feature importance in the final model.

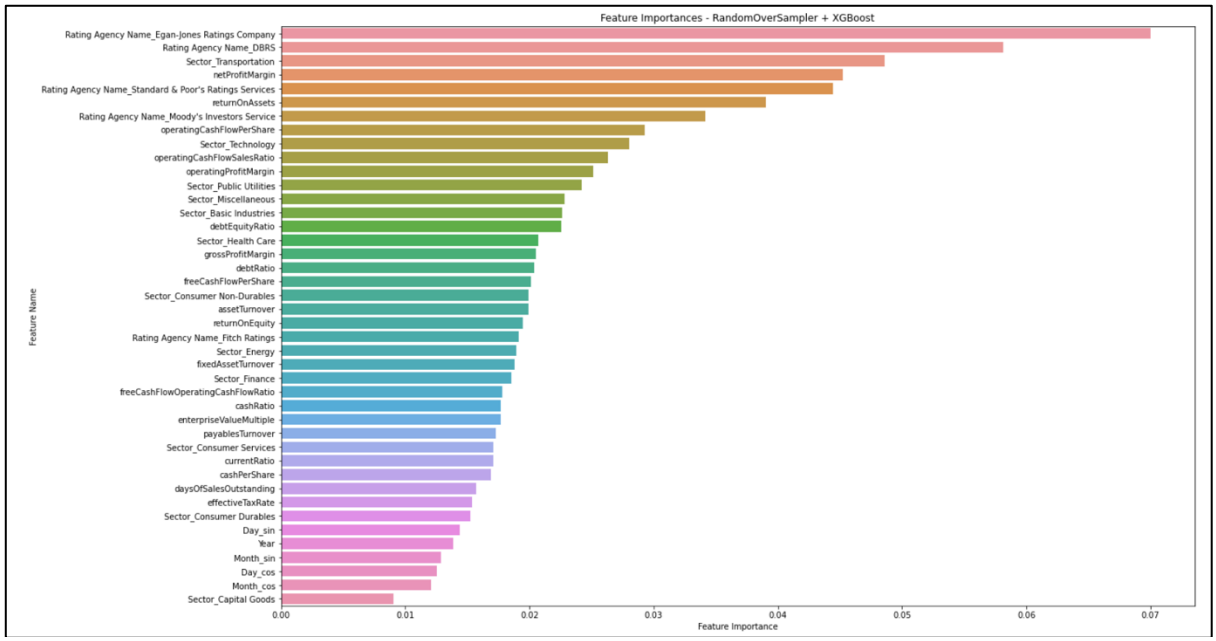

Figure 14: Feature Importance for XGBoost with random oversampling

Feature importance is often used to remove irrelevant features from the model and retain only those with the highest predictive power. It is also useful to communicate to stakeholders which features carry the greatest importance from a business perspective. In this case, the rating agency name appears to heavily influence model prediction. This is not unexpected given the observation made from Figure 2.

For interest, multiple thresholds were tested for selecting features by feature importance. This was tested by incrementally reducing the subset from the full input set to a set with only the most important feature, and each time calculating the F1-score. Figure 15 shows the result of this analysis.

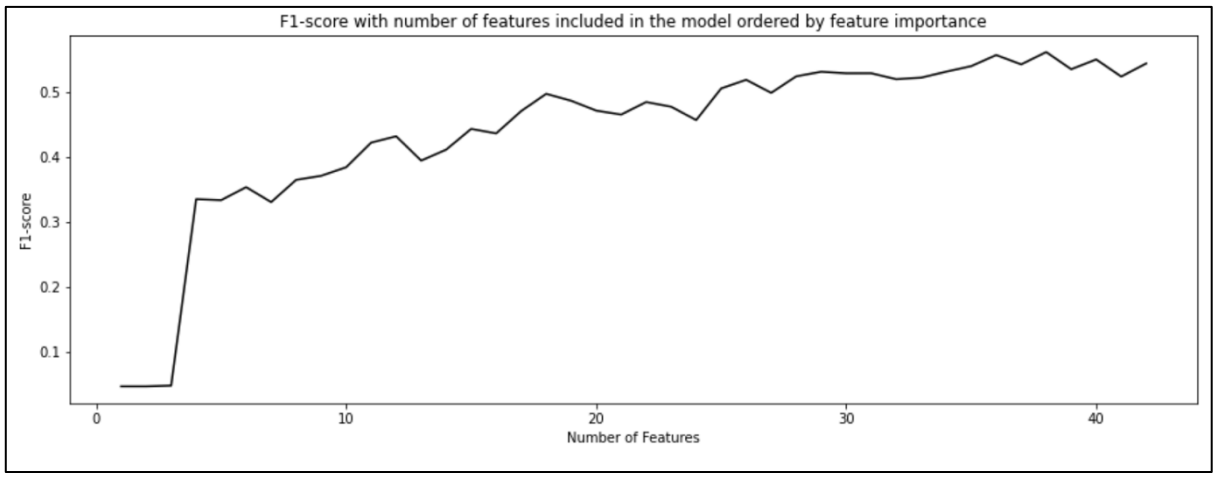

Figure 15: F1-score with number of features included in the model, with features added in order of importance.

The model performance increases with number of features, leading to a trade-off between model complexity and model performance. It appears that the optimal number of features is below the maximum, indicating that feature selection would be beneficial. In practice, a discussion with stakeholders would be valuable at this stage to determine which features it is preferred to retain in the model.

3. Discussion

3.1 Justification

A number of characteristics of the dataset may explain the relatively weak out-of-sample performance of this model.

Firstly, the dataset used in this project suffers from a class imbalance, an issue that is magnified by the lack of training samples. This negatively impacts the model's ability to represent combinations of features and their mapping to class labels. Without a sufficient number of samples from the minority class, the model was unable to generalize a class boundary to categorise new examples.

Secondly, the issue of class separability is highly relevant to this dataset and is compounded by the limited type of information present in the dataset. That is, the financial risk profile captured in the dataset is only one component in the credit rating decision. Other aspects such as the business risk profile (competitive positioning, industry-specific risks, operating risk, governance policies etc.) must also be considered. From a modelling perspective, this translates into poor class separability, given that companies with similar financial statements may receive very different ratings. Imbalanced data exacerbates this issue due to a lack of examples in the minority class. For example, the class named 'CCC and below' represents the 72 least creditworthy companies. There may be several drivers of poor creditworthiness and given enough data, this may form multiple clusters of observations within the feature space. However, in this case, the class is so small that any groupings would not be apparent; rather observations would appear as one sparse group that mixes in with creditworthy companies such that class separability is difficult.

XGBoost is a good choice of algorithm for this dataset, as it produces the best predictive ability on future datasets of the models compared. The algorithm was able to handle characteristics of the data such as collinearity and outliers that hamper the ability of other ML algorithms. However, an important consideration when using XGBoost in a financial context is the lack of explainability of the results. In practice, there is likely to be interest from stakeholders in factors driving the results; it is crucial to be able to break down the classification of ratings by financial indicator. Algorithms such as logistic regression may be preferable for this reason.

3.2 Conclusion and Evaluation

This project aimed to develop a classification model used to predict corporate credit ratings based on key financial indicators. Initial EDA was used to describe the corporate credit ratings dataset and uncover key characteristics of the data. A number of machine learning models were applied to the dataset and the performance compared. XGBoost was found to produce the highest performance as measured by weighted F1-score and was chosen to refine via hyperparameter tuning. Three resampling techniques were compared in conjunction with hyperparameter tuning, and the highest performance was observed using random oversampling, with a tuned hyperparameter set that produced a weighted F1-score of 54%. As a final step, the model was applied on unseen test data and a final weighted F1-score of 54% was achieved.

It was interesting to observe that neither hyperparameter tuning nor resampling had a significant impact on the final model performance. It could be the case that the distribution of train and test set differ significantly, although this is unlikely given that shuffling was applied during the train/test split. It is more likely that because the dataset size relative to number of features is small, overfitting is still present to a large extent. This is not unusual for this type of problem, but severely limits the performance of the algorithm.

A number of improvements could be made to the model implementation.

Firstly, in this project, hyperparameters were tuned in a single step to simplify the procedure. However, hyperparameters may be adjusted sequentially and repeatedly to achieve optimal results. Other automatic hyperparmeter tuning methods such as HyperOpt or Ray Tune could be investigated, or manual tuning may be attempted. This would provide greater control over the process but would likely be more time consuming, particularly when tuning several hyperparameters.

Secondly, cross validation could be employed to a greater extent, to prevent the precise train/test split that occurs having a great impact on the final F1-score estimate. This could involve a greater number of folds or repeated cross validation.

Several major machine learning algorithms were compared before choosing to further refine XGBoost. However, other machine learning tools such as deep learning could be explored. As described earlier, the size of the dataset greatly prohibits the ability of the algorithm to produce a strong performance. However, huge volumes of financial data and qualitative data exist, which if harnessed, could lead to significant gains in accuracy. Deep learning algorithms are particularly well-equipped to handle big data. For example, Convolutional Neural Networks have been explored in this space, in part due to the ability to perform feature selection. Given a large number of financial performance indicators, feature selection could be a useful step in the prediction of credit ratings. Algorithms such as Recursive Neural Networks may also hold potential since they are able to connect temporal data. This is useful since financial performance in the quarter preceding the rating assignation is a good indicator of rating.

It would be interesting to explore whether the outcome for this type of problem is driven mainly by the choice of data, or level of sophistication of the model. However, it is clear that any machine learning model should be monitored and revised as data on more companies becomes available.

4. References

Baeldung. 2022. 'F-1 Score for Multi-Class Classification.' Available at https://www.baeldung.com/cs/multi-class-f1-score

Bhandari, A. 2020. Analytics Vidhya. 'Feature Engineering: Scaling, Normalization, and Standardization (Updated 2023)'. Available at https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/

Brownlee, J. 2020. Machine Learning Mastery. 'Why Is Imbalanced Classification Difficult?' Available at https://machinelearningmastery.com/imbalanced-classification-is-hard/

Clarke, M. 2022. Practical Data Science. 'How to Use Optuna for XGBoost Hyperparameter Tuning'. Available at https://practicaldatascience.co.uk/machine-learning/how-to-use-optuna-for-xgboost-hyperparameter-tuning

Hajek et al. 2013. Feature Selection in corporate credit rating prediction. Knowledge-Based Systems, Volume 51, 2013, Pages 72-84, ISSN 0950-7051. Available at https://doi.org/10.1016/j.knosys.2013.07.008

Mazumder, S. 2021. Analytics Vidhya. '5 Techniques to Handle Imbalanced Data For a Classification Problem'. Available at https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/

Pol et al. 2022. 'Predicting Credit Ratings using Deep Learning Models – An Analysis of the Indian IT Industry'. AABFJ Volume 16 Issue 5 2022. Available at https://ro.uow.edu.au/cgi/viewcontent.cgi?article=2282&context=aabfj

Roy, B. 2019. Towards Data Science. 'All about Categorical Variable Encoding'. Available at https://towardsdatascience.com/all-about-categorical-variable-encoding-305f3361fd02

Varghese, D. 2018. Towards Data Science. 'Comparative Study on Classic Machine Learning Algorithms'. Available at https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222

Wang et al., 2021. "Harnessing Machine Learning Emerging Technology in Financial Investment Industry: Machine Learning Credit Rating Model Implementation", *Journal of*

Financial Risk Management, *Vol.10 No.3, 2021.* Available at https://doi.org/10.4236/jfrm.2021.103019

XGBoost Developers. 2022. Read The Docs. XGBoost Parameters. Available at https://xgboost.readthedocs.io/en/stable/parameter.html