Corporate Credit Rating Prediction

1st Abdulaziz Alhumaidy
UC Davis
Davis, CA
alhum@ucdavis.edu

2nd Abdulah Haitham
UC Davis
Davis, CA
ahaitham@ucdavis.edu

3rd Vedant Patel
UC Davis
Davis, CA
vsxpatel@ucdavis.edu

4th Vikram Penumarti
UC Davis
Davis, CA
vpenumarti@ucdavis.edu

5th Elijah Alba
UC Davis
Davis, CA
elialba@ucdavis.edu

I. Introduction

A. What are Credit Ratings?

Credit ratings are assessments of a company's ability to meet its financial obligations, such as repaying loans or bonds. They serve as an essential measure of a corporation's financial health and reliability. These ratings are issued by specialized agencies, including Moody's, Standard and Poor's, and Fitch, and are widely used by various stakeholders. Investors rely on credit ratings to evaluate the risk associated with corporate debt instruments, while lenders use them to determine loan terms and interest rates. Additionally, corporations benefit from favorable credit ratings by securing lower borrowing costs and attracting potential investors. Credit ratings exist to provide transparency, reduce information asymmetry, and facilitate informed decision-making in financial markets. By offering a standardized evaluation, they play a critical role in maintaining trust and efficiency within the global financial ecosystem.

B. Problem Statement

While credit ratings are integral to financial decision-making, the traditional processes behind their assignment face significant challenges. These processes often rely on manual, qualitative analyses conducted by specialized agencies. This reliance leads to issues such as opacity, inefficiency, and susceptibility to biases, making it difficult to adapt to the dynamic nature of financial markets. Additionally, these limitations hinder the scalability and timely updating of ratings, leaving a critical gap in accurately reflecting evolving financial risks and complexities.

C. Background

Recent advancements in machine learning (ML) have demonstrated their potential to address inefficiencies in credit rating prediction. ML models, such as decision trees, support vector machines (SVMs), and neural networks, excel in identifying complex patterns in financial data, enabling rapid and accurate predictions. Studies like [9] and [11] have shown that ML approaches outperform traditional methods in predictive accuracy while adapting to diverse datasets. However, a critical limitation remains—many ML models function as "black boxes," offering little interpretability for financial analysts. Explainable AI (XAI) and hybrid approaches that blend accuracy with interpretability are emerging as solutions but are not yet widely implemented in credit rating prediction.

D. Motivation and Scope

This research seeks to bridge the gap between predictive accuracy and model interpretability by leveraging advanced ML techniques for corporate credit rating prediction. By addressing challenges such as class imbalance, data sparsity, and variability, this study aims to develop a robust, transparent, and scalable framework. Specifically, we focus on feature selection, ensemble learning, and explainable AI methods to ensure that predictions are not only accurate but also actionable for financial analysts. This work contributes to the growing body of research by providing a comprehensive solution that balances technical rigor with practical usability.

II. Relevant Literature

Recent research demonstrates the potential of machine learning in predicting corporate credit ratings, with varied methodologies. Golbayani et al. (2020) used U.S. corporate data to compare models like decision trees and random forests, introducing the "Notch Distance" metric [9]. Yu et al. (2021) applied tree-based models to Eurozone firms, emphasizing sustainability metrics [12]. Makwana et al. (2024) focused on explainable AI with decision trees, highlighting interpretability and temporal stability in predictions [13].

Specifically, the differences we found are:

A. Datasets

- Golbayani et al. (2020): Used U.S. corporate data from quarterly financial statements with features like debt-to-equity ratio, ROA, and profitability indicators [9].
- Yu et al. (2021): Focused on Eurozone firms ranked by climate change scores, incorporating macroeconomic variables and firm-level climate impact data [12].
- Makwana et al. (2024): Curated a large dataset of U.S. companies, combining financial ratios and a time-based train-test split for temporal stability analysis [13].

B. Selection Technique

All studies emphasized financial ratios as primary predictors, with variations:

- Golbayani et al.: Focused on traditional financial ratios like ROA and debt-to-equity [9].
- Yu et al.: Incorporated sustainability-related factors like carbon neutrality scores [12].
- Makwana et al.: Blended domain knowledge with correlation analysis to identify key features [13].

C. Machine Learning Models

- Golbayani et al.: Explored decision trees, random forests, support vector machines (SVM), and neural networks, with decision tree-based models performing best [9].
- Yu et al.: Compared classification and regression trees with random forests for credit rating predictions, favoring tree-based models for their precision [12].
- Makwana et al.: Focused on explainable AI (decision trees) to enhance interpretability, balancing accuracy and explainability [13].

D. Evaluation Metrics

- Golbayani et al.: Introduced a novel "Notch Distance" metric alongside accuracy to quantify prediction deviations from actual ratings [9].
- Yu et al.: Emphasized robustness of predictions across investment and speculative-grade categories [12].
- Makwana et al.: Evaluated precision and temporal stability of derived decision-tree rules [13].

III. Dataset Description

We used the same data set used by [13]. The dataset provides comprehensive information on corporate credit ratings and associated financial ratios for 7,805 corporations. It includes metadata such as rating agency, corporation details, and financial metrics. With 25 different attributes, this dataset provides an extensive set of variables essential for analyzing corporate solvency and credit risk. Below are the key attributes included in the dataset.

A. Metadata Columns

- **Rating Agency:** The agency that provided the credit rating.
- **Corporation:** Name of the corporation being evaluated.
- **Rating:** Credit rating assigned to the corporation ranging from AAA (most promising) to D (most risky).
- **Rating Date:** The date the rating was assigned.
- **CIK:** Central Index Key, a unique identifier for corporations.
- **Binary Rating:** A binary categorization of the credit rating (investment-grade vs. non-investment-grade).
- **SIC Code:** Standard Industrial Classification code of the corporation.
- **Sector:** Sector or industry the corporation belongs to (e.g., Utilities, Business Equipment).
- **Ticker:** Stock ticker symbol of the corporation.

B. Financial Ratios

The dataset includes 17 financial ratios to provide insights into different aspects of financial health and performance to assess creditworthiness:

- **Current Ratio:** Liquidity ratio, calculated as current assets divided by current liabilities.
- **Long-term Debt / Capital:** Measures long-term debt as a percentage of total capital.

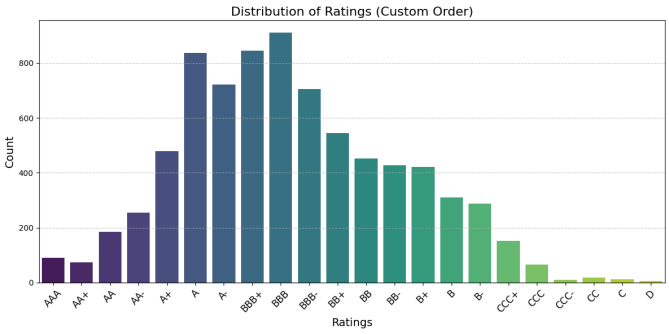

Fig. 1: Distribution of corporate credit ratings, showing the frequency of each rating category

- **Debt/Equity Ratio:** Leverage ratio, showing the proportion of debt to shareholders' equity.
- **Gross Margin:** Profitability ratio, calculated as gross profit divided by revenue.
- **Operating Margin:** Operating profit as a percentage of revenue.
- **EBIT Margin:** Earnings before interest and taxes as a percentage of revenue.
- **EBITDA Margin:** Earnings before interest, taxes, depreciation, and amortization as a percentage of revenue.
- **Pre-Tax Profit Margin:** Profit before taxes as a percentage of revenue.
- **Net Profit Margin:** Net income as a percentage of revenue.
- **Asset Turnover:** Measures how efficiently a company uses its assets to generate revenue.
- **ROE (Return on Equity):** Net income divided by shareholders' equity.
- **Return on Tangible Equity:** Return on equity considering only tangible equity.
- **ROA (Return on Assets):** Net income divided by total assets.
- **ROI (Return on Investment):** A broader profitability metric.
- **Operating Cash Flow Per Share:** Operating cash flow on a per-share basis.
- **Free Cash Flow Per Share:** Free cash flow on a per-share basis.

IV. Exploratory Data Analysis

A. Distribution and Frequency Analysis

To understand the distribution of corporate credit ratings, we visualized the frequency of each rating category (Figure 1). The bar chart highlights the uneven distribution of ratings, with categories like "A" and "BBB" being the most frequent, while lower ratings such as "CCC" and "D" are significantly underrepresented. This imbalance is critical as it directly influences model performance and necessitates preprocessing techniques like oversampling or class weighting to address the bias during training.

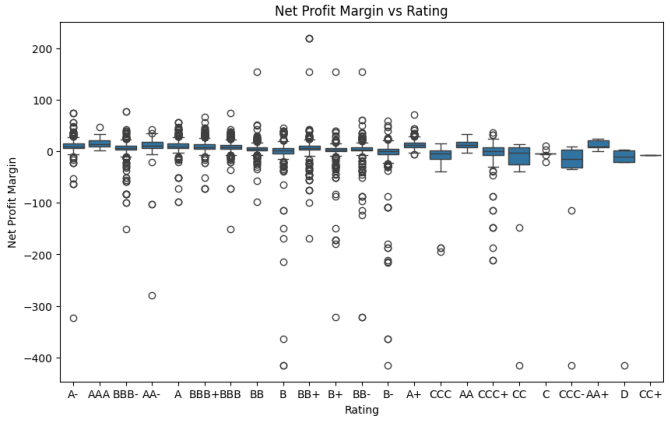

Fig. 2: Boxplot analysis of key financial features, identifying outliers in the "Net Profit Margin" Variable

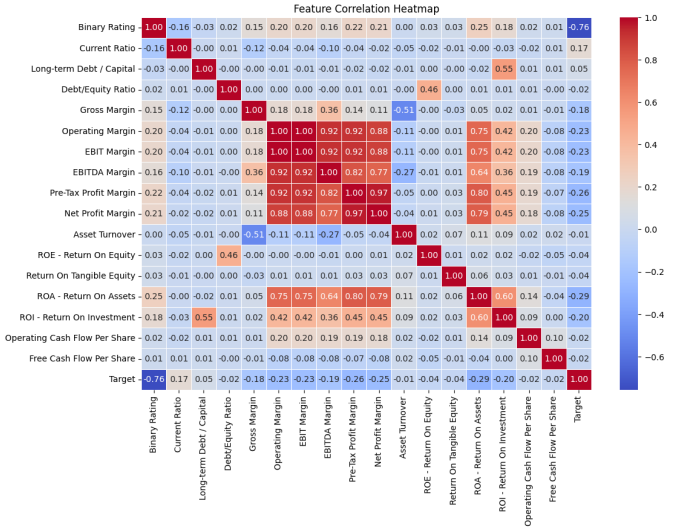

Fig. 3: Correlation heatmap of financial features, emphasizing highly correlated variables like "EBITDA Margin" and "Operating Margin" and their relationship with the target variable

B. Preprocessing

The dataset was analyzed for outliers and missing values to ensure data quality before training the models. A boxplot analysis (Figure 2) revealed the presence of outliers in features like "Net Profit Margin" These outliers were addressed using a combination of z-score filtering and robust scaling methods to minimize their impact on the model's predictions. Additionally, features were normalized to handle the varying scales of financial ratios and macroeconomic data, ensuring compatibility with distance-based algorithms like KNN.

C. Feature Selection

Feature selection was conducted based on the correlation heatmap (Figure 3), which highlights the relationships between financial features and the target variable. Highly correlated features, such as "EBITDA Margin" and "Operating Margin," were prioritized for inclusion due to their predictive strength. Conversely, features with minimal or no correlation to the target, such as "Asset Turnover," were excluded to streamline the model and reduce computational overhead. This process ensured that the model focused on the most relevant features, improving accuracy and efficiency.

V. Methodology

A. Overview

The methodology focuses on developing machine learning models to predict corporate credit ratings using financial ratios and macroeconomic variables. The process involves data preprocessing, feature selection, and implementing models such as KNeighbors Classifier, Random Forest, and Neural Network. This approach emphasizes accuracy and interpretability, aiming to address limitations identified in prior research.

B. Data Collection

The dataset was sourced from Kaggle [14], containing financial ratios such as debt-to-equity, return on assets (ROA), and profitability indicators for U.S. companies. Unlike the datasets used by Golbayani et al. [9] and Yu et al. [12], this dataset combines attributes that align with both traditional financial metrics and modern analytical needs. To ensure data integrity, missing values were addressed, and features were standardized.

C. Data Preprocessing

- **Handling Missing Data**: Missing values were handled by mean imputation, a technique also used in Golbayani et al. [9].
- **Normalization and Scaling**: Features were normalized to ensure compatibility with distance-based algorithms like KNeighbors Classifier.
- **Outlier Detection**: Boxplot analysis was performed to identify outliers, which were capped at the 1st and 99th percentiles to minimize their impact.
- **Target Variable Reduction**: We reduced the class size from over 20 classifications to 13, allowing the model to more accurately classify data. [12].

D. Feature Engineering and Selection

Feature selection was guided by exploratory data analysis and domain knowledge. A correlation heatmap was utilized to identify highly correlated features, similar to the approach in Makwana et al. [13]. Key features like debt-to-equity ratio and ROA were selected for their predictive significance, diverging from Yu et al. [12], which emphasized sustainability metrics.

E. Model Development

- **KNeighbors Classifier**: Implemented as a non-parametric model, with hyperparameters such as the number of neighbors (k) tuned using grid search. Weighted distance metrics were applied to prioritize closer neighbors, following best practices from Divekar and Agarwal [14].

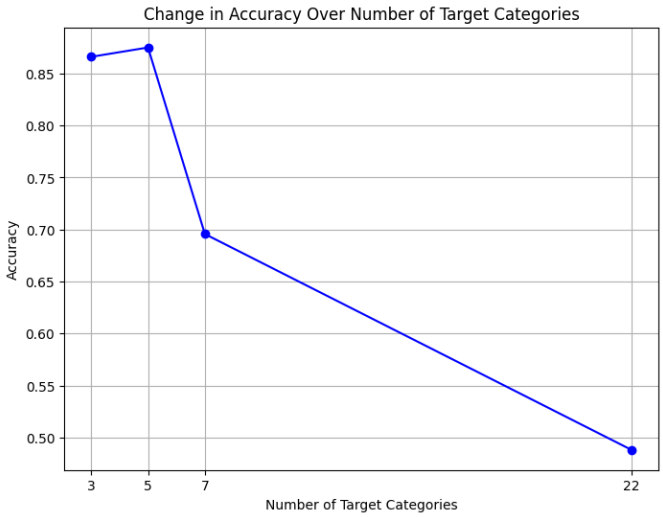

Fig. 4: KNN Model Accuracy over different target variable categories

- **Random Forest**: Designed as an ensemble model to handle feature interactions and reduce overfitting. Hyper-parameters, including the number of estimators and max depth, were optimized using grid search. This differed from Golbayani et al. [9], where simpler decision trees were emphasized.
- **Neural Networks**: Although mentioned in the research outline, they were not prioritized due to interpretability challenges highlighted in Makwana et al. [13].

F. Evaluation Metrics

The models were evaluated using metrics such as accuracy, precision, recall, and F1 score. Golbayani et al. [9] introduced a "Notch Distance" metric, but this was not applicable due to differences in data granularity. Instead, the focus was on class-level performance, emphasizing interpretability and robustness in predictions.

G. Workflow Summary

The overall workflow included the following steps:

1) Data preprocessing: Handling missing values, scaling, and balancing classes.
2) Feature selection: Identifying key financial metrics using correlation analysis.
3) Model implementation: Developing and optimizing KNeighbors Classifier and Random Forest.
4) Evaluation: Comparing models based on defined metrics and reflecting on differences from prior work.

VI. Experimental Results and Evaluation

A. Neural Networks

Neural networks, inspired by the human brain's architecture, are computational models designed to recognize patterns and relationships within data. They consist of interconnected layers of nodes, or neurons, that process input data to produce outputs. Each connection between neurons has an associated weight, which adjusts as the network learns, enabling the model to make accurate predictions or classifications [1].

1) Structure of Neural Networks:

- **Input Layer:** This layer receives raw input data. Each neuron in this layer can be considered a feature or an attribute from the data with a corresponding value.
- **Hidden Layers:** These layers are placed between input and output layers and performs complex calculations on the input data.
- **Output Layer:** This layer is the final layer of the neural network that produces final output values such as classification labels or continuous values for each neuron in the output layer based on the given problem.

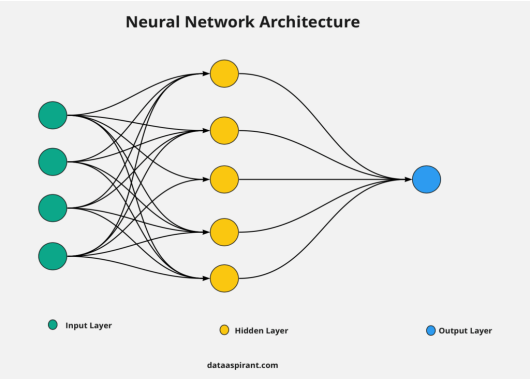

Fig. 5: Neural Network Architecture

Activation functions are used by each neuron to introduce non-linearity into the model enabling it to capture complex and intricate patterns in the data.

2) Learning Process: Neural networks rely on training data to improve its learning accuracy over time. This improvement is brought upon by adjusting the weights of connections between neurons and minimizing the loss function (difference between predicted and the actual outputs).

3) Why Neural Networks are a Good Choice: Financial datasets, such as those used in credit rating prediction, often exhibit intricate interactions between features, making neural networks a suitable candidate for this task. Neural networks with a softmax activation in the output layer are well-suited for multi-class classification problems which is a major requirement for this project. Financial datasets often have features with significant interdependencies (e.g., profitability ratios vs. leverage ratios). Neural networks inherently model these interactions without requiring manual feature engineering [2]. Lastly, it is proven in other adjacent researches that neural networks have performed well with financial tasks such as credit scoring, fraud detection, risk assessment *etc.*

These characteristics make neural networks a powerful tool for corporate credit rating prediction.

4) Initial Model Development: Our initial neural network model was developed according to standard machine learning steps as discusses in detail below:

a) Target Variable Encoding: The target variable, "Rating," consisted of multiple class labels of credit ratings. To facilitate numerical processing by the neural network we utilized scikit-learn's LabelEncoder to convert these class labels into numerical format.

b) Data Splitting: We partitioned the dataset into training and testing sets using an 80-20 split.

c) Model Architecture: The neural network was constructed using the Sequential API from TensorFlow's Keras library. The architecture included:

- An input layer consisting of same number of neurons as the number of features present in the preprocessed dataset.
- Three hidden layers with 128, 64, and 32 neurons, respectively, each utilizing the ReLU activation function. The ReLU function introduces non-linearity, enabling the network to learn complex patterns.
- Dropout layers with a rate of 0.2 to prevent overfitting by randomly setting a fraction of input units to zero during training.
- For the output layer we used softmax activation funtion to produce probability distributions suitable for multi-class classifications [3].

d) Compilation and Training: We used Adam optimizer for compiling the model known for its efficiency and low memory requirements [4], and the sparse categorical cross-entropy loss function, appropriate for multi-class classification problems where the target variable is encoded as integers [5]. We trained the model over 50 epochs with a batch size of 32 which was a tried and tested number for convergence of the model for our dataset.

e) Evaluation: Model performance was evaluated on the test set and we obtained an accuracy around 33% which was extremely below our expectations indicating the need for further refinement and optimization.

B. KNeighbors Classifier

The K-Nearest Neighbors (KNN) algorithm was selected for its straightforward and interpretable framework in classifying financial data. Its non-parametric nature, which does not assume any underlying data distribution, makes it well-suited for financial ratios and macroeconomic variables. Research by [15] demonstrated an accuracy of 0.9213 using KNN for similar tasks, and this study aimed to replicate that performance.

KNN operates on the principle of proximity, classifying a data point based on the majority class among its k nearest neighbors. These neighbors are identified using a distance metric, typically Euclidean distance. To enhance performance, an inverse distance weighting scheme can be applied, giving higher importance to closer neighbors. This is especially effective for financial datasets, where localized relationships between features are critical.

The algorithm functions in two phases: (1) *Training Phase*, where all training examples (feature vectors and labels) are stored, and (2) *Classification Phase*, where the k closest neighbors of a given input are identified, and the majority class is assigned as the prediction. The choice of k is critical; smaller k values may lead to overfitting, while larger values can oversmooth decision boundaries. Cross-validation was used to optimize k and other hyperparameters.

a) Parameter Selection::

- **Neighbors:** The number of neighbors (k) was tuned via cross-validation for optimal performance.
- **Algorithm:** The KDTree algorithm was used for efficient computation of nearest neighbors.
- **Distance Metric:** Euclidean distance was the primary metric, with alternatives like Hamming distance explored for robustness.

By applying these techniques, the KNN classifier effectively captured localized patterns in financial data, making it a robust choice for predicting corporate credit ratings.

C. Random Forest

Random Forest is an ensemble learning algorithm that combines weak predictions of several decision trees into a singular strong prediction. This method of predicting can improve accuracy and mitigate overfitting [14].

1) Structure of Model:

- **Bagging**: Each tree is built using a bootstrap sample of training data. This ensures different subsets of the data are used for different trees [16].
- **Ensemble Learning**: The predictions of all trees constructed are averaged together to create a strong, holistic prediction using different features and sections of the dataset [16].
- **Random Feature Selection**: A random subset of features is considered for each node, reducing the correlation between trees, decreasing the overall variance [16].

2) Why Random Forest is a Good Choice: Random Forest is less sensitive to noise in the data due to decision trees being constructed on different subsets of the data (bagging) and individual predictions being averaged together (ensemble learning) to generate predictions which are not skewed by individual sections of data. The random feature selection of the model ensures that correlation between decision trees are minimized, creating more generalizable predictions. Overall, Random Forest is suitable for credit score classification due to its ability to capture complex patterns while reducing overfitting through ensemble learning. Research [9] indicates its superior performance over Neural Networks and SVMs attributable to its ability to handle noisy, unbalanced datasets effectively.

3) Parameter Selection:

- **Estimators**: The number of trees in the forest.
- **Max Depth**: The maximum depth of each tree.
- **Min Samples Split**: The minimum number of samples required to split an internal node.
- **Min Samples Leaf**: The minimum number of samples required in a leaf node.

By fine tuning these parameters, bias, variance, and overfitting can be reduced significantly.

D. Insights and Observations

Initially, the models that we traind achieved extremely low accuracies which were below our expectations. This led us to review of other thoroughly prepared corporate credit rating projects and we discovered:

1) Impact of Class Imbalance: Class imbalance occurs when certain classes are underrepresented which might deteriorate the performance of neural network. This is because models may become biased towards majority classes hindering their ability to generalize eventually affecting minority classes. Addressing class imbalance became crucial to improve our model accuracy.

2) Effect of Reducing the Number of Classes: Simplifying the classification task by reducing the number of target classes can improve model performance. Fewer classes can lead to better discrimination and higher accuracy, as the model focuses on more distinct categories.

VII. Model Optimization and Hyper-parameter Tuning

Optimization is an essential step towards building a machine learning model so that the model can try and figure out what exact parameters it needs to effectively learn complex patterns from the data while also avoiding over-fitting. Despite efforts, it was observed that many of the hyperparameters had minimal to no impact on the final model's performance. Below we discuss the hyperparameters explored, their role in the model, and the observed results.

A. Neural Networks

We employed the GridSearchCV optimization algorithm for hyperparameter tuning our neural network and incorporated **Dropout** as a regularization technique.

1) GridSearchCV for Hyperparameter Tuning: **GridSearchCV** systematically evaluates combinations of hyperparameters to identify the configuration that yields the best performance results for the machine learning model. The following hyperparameters were optimized:

- **Batch Size:** Values of 16, 32, and 64 were tested to determine the most efficient mini-batch size for training.
- **Epochs:** Training durations of 50, 100, and 150 epochs were evaluated.
- **Learning Rate:** The learning rate controls the size of the step taken in the direction of the gradient during backpropagation. Learning rates of **0.001, 0.01, and 0.1** were tested.
- **Momentum:** Momentum works as inertia for the model making that higher values of momentum forces the model to make updates to the weights during backpropagation in the same direction as the previous weights. Values of **0.0, 0.5, and 0.9** were explored, but it only applies to **SGD**.
- **Optimizers:** *Adam*, *RMSprop*, and SGD optimizers were considered, given their proven effectiveness in neural network training [6].
- **Dropout Rate:** Rates of 0.2 and 0.3 were tested to assess the impact of regularization.
- **Weight Initialization:** Strategies like *glorot_uniform* and *he_uniform* were explored to ensure stable model convergence [7].

Cross-validation: GridSearchCV inherently uses cross-validation (with *cv=3*) to evaluate model performance across multiple data splits, ensuring the generalizability of the selected hyperparameters.

2) Regularization with Dropout: Dropout regularization randomly deactivates neurons during training to reduce model's dependency on specific neurons, increasing robustness in the model [8]. Dropout layers with rates of 0.2 and 0.3 were included between dense layers, balancing regularization with the network's ability to learn.

3) Additional Regularization Techniques: :

- **Early Stopping:** Early stopping was intended to halt training once validation loss failed to improve for 10 consecutive epochs. However, since validation loss consistently decreased, early stopping was never triggered, and training proceeded for the full 150 epochs.
- **L2 Regularization:** L2 regularization penalizes large weights in neural networks to improve generalization. Despite testing L2 coefficients of **0.001, 0.01, and 0.1**, no significant improvements were observed. This is likely because **other regularization techniques, such as Dropout and Early Stopping, were already sufficient**, rendering L2 regularization redundant.

4) Results of Hyperparameter Tuning: Best parameters identified by GridSearchCV are as follows:

- **Optimizer:** RMSprop
- **Batch Size:** 16
- **Dropout Rate:** 0.2
- **Epochs:** 150
- **Weight Initialization**: Glorot Uniform

5) Performance Improvements: After reducing the number of target classes to simplify the classification task and allow the model to focus on more distinct categories followed by application of GridSearchCV optimization algorithm integrated with regularization techniques, the test accuracy improved significantly from the intial 33% to approximately 74%.

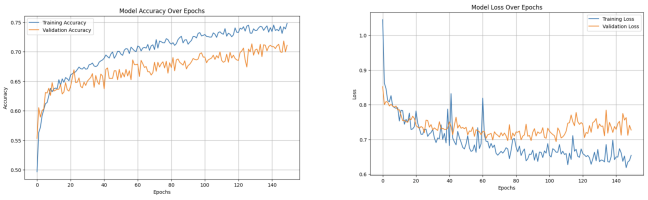

Fig. 6: Performance metrics of the optimized model.

B. KNN

1) Optimization Techniques: To enhance the predictive performance of the K-Nearest Neighbors (KNN) classifier, we

TABLE I: Class Encoding Scheme used for the categorical data.

Class Group	Encoded Value
{AAA, AA+, AA-, AA}	0
{A, A+, A-}	1
{BBB, BBB+, BBB-}	2
{BB, BB+, BB-}	3
Others	4

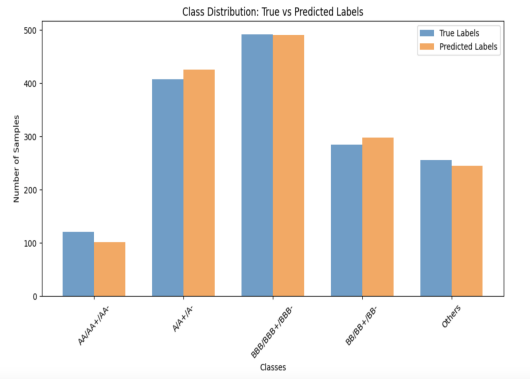

Fig. 7: Class Distribution of True and Predicted Labels

employed a multi-step optimization strategy involving several techniques. These techniques ensured that the model generalized well to unseen data while maintaining computational efficiency.

- **Hyperparameter Tuning with Grid Search**: We conducted an exhaustive *Grid Search* over a predefined hyperparameter space to identify the optimal configuration for our model. The hyperparameters tuned included:
  - **Number of Neighbors (*n_neighbors*)**: We evaluated values ranging from 3 to 20 to identify the best trade-off between flexibility and stability in decision boundaries.
  - **Weights**: We compared two schemes:
    - **Uniform**: All neighbors contributed equally to predictions.
    - **Distance**: Closer neighbors were given more weight, enhancing localized predictions.
  - **Distance Metric (*metric*)**: We explored multiple distance metrics:
    - **Manhattan (L1 distance)**: Measuring absolute differences between coordinates.
    - **Euclidean (L2 distance)**: Calculating straight-line distances.
    - **Minkowski**: Generalizing L1 and L2 metrics for different values of p.
  - **Algorithm (*kd_tree*)**: We specified the *kd_tree* algorithm for efficient nearest-neighbor searches, particularly suited for high-dimensional data.

  Grid Search was conducted using 5-fold cross-validation, and weighted accuracy was used as the scoring metric to ensure fair evaluation across class distributions.
- **Cross-Validation**: We utilized 5-fold cross-validation during the Grid Search process to assess model performance across multiple train-validation splits. This technique allowed us to evaluate how well the model generalized to unseen data. Cross-validation reduced the risk of overfitting by averaging performance across different subsets of the training data.
- **Test Set Splits**: We experimented with different test set split ratios (*test_size*), including 0.15, 0.20, 0.25, and 0.30, to assess the robustness of the model under varying train-test distributions. This experimentation revealed that a test size of 0.15 provided the best balance between training data availability and test set evaluation; see Figure 8.
- **Weighted Metrics**: Since our dataset contained imbalanced classes, we evaluated the model using weighted metrics, including weighted precision, recall, and F1 scores. These metrics ensured that all classes contributed proportionally to the evaluation, avoiding bias toward majority classes.
- **Distance-Based Weighting**: The *weights* hyperparameter in KNN was tuned to enhance the model's ability to handle localized patterns. Using *distance*-based weighting allowed the classifier to prioritize closer neighbors, improving classification accuracy in regions with dense data points.
- **Preliminary Manual Tuning**: Before conducting Grid Search, we performed initial experiments to manually test different configurations of hyperparameters such as *n_neighbors*, *weights*, and *metric*. These experiments provided insights that informed the design of the Grid Search parameter grid.

2) Performance Before and After Optimization: The performance of the KNN classifier improved significantly after optimization. Table II summarizes the key performance metrics before and after hyperparameter tuning. Before optimization, the model achieved a weighted F1 score of 0.7796, with precision and recall of 0.7821 and 0.7509, respectively. After optimization, the F1 score increased to 0.7815, indicating better generalization to unseen data.

TABLE II: Performance Metrics Before and After Optimization

Metric	Before Optimization	After Optimization
Accuracy	0.7783	0.7823
Precision	0.7821	0.7886
Recall	0.7783	0.7823
F1 Score	0.7796	0.7815

Figure 8 illustrates the variation in accuracy across different numbers of neighbors (*n_neighbors*) and test set splits. We found that the configuration with *n_neighbors* = 5, *metric* = *'manhattan'*, and *weights* = *'distance'* consistently outperformed other combinations.

3) Reflections on Optimization: The optimization process provided several key insights:

- **Key Hyperparameters**: The number of neighbors (*n_neighbors*) had the most significant impact on accuracy. A value of 5 provided the best trade-off between overfitting (small n) and underfitting (large n). Additionally, *distance*-based weighting consistently improved performance in regions with dense data points. The Manhattan distance metric was the most robust, outperforming Euclidean and Minkowski metrics for this dataset.
- **Challenges Encountered**: The computational cost of Grid Search was a limiting factor, particularly when exploring large parameter grids. Additionally, imbalanced class distributions required careful evaluation using weighted metrics to avoid skewed results.
- **Future Considerations**: Incorporating feature selection techniques or dimensionality reduction, such as PCA, could enhance computational efficiency without compromising performance. Exploring ensemble methods or hybrid distance metrics could also lead to additional improvements.

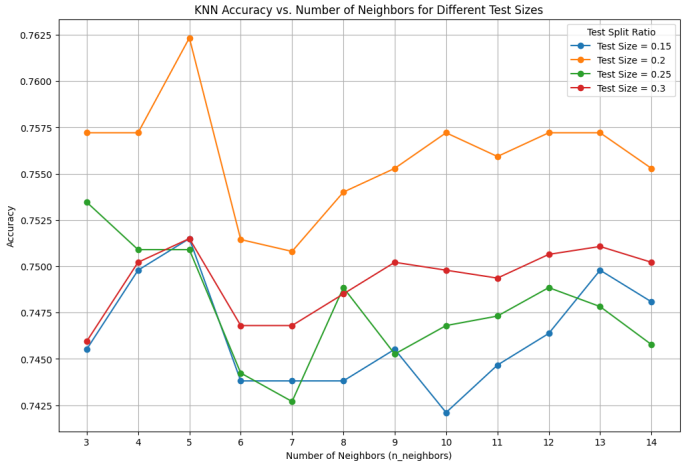

Fig. 8: Accuracy vs. Number of Neighbors for Different Test Set Splits.

C. Random Forest

1) Optimization Techniques: A grid search was used to automate the search for the ideal hyperparameters for the Random Forest model.

- **Hyperparameter Tuning with Grid Search**: The hyperparameters tuned included:
  - **Estimators (*n_estimators*)**: We evaluated values ranging from 50 to 200 to identify the best trade-off between accuracy and computational complexity. We ultimately settled on 50 estimators.
  - **Max Depth (*max_depth*)**: Values None through 30 were evaluated for max depth to identify the trade-off between a more complex model with potential overfitting and a simpler model which could potentially underfit. The grid search yielded a value of 20.
  - **Min Samples Split (*min_samples_split*)**: Values 2 through 10 were evaluated for min samples split to identify the trade-off between a model which captures more detail by splitting more or may underfit by splitting less. The grid search yielded a value of 5.
  - **Min Samples Leaf (*min_samples_leaf*)**: Values 1 through 4 were evaluated for min samples leaf to identify the trade-off between a model which is more well fit to the data, or a more generalizable model which may perform better on unseen data. The grid search yielded a value of 5.

2) Performance Before and After Optimization: After preprocessing and hyperparameter optimization the random forest model saw a significant increase in all standard metrics.

TABLE III: Performance Metrics Before and After Optimization

Metric	Before Optimization	After Optimization
Accuracy	0.4946	0.7892
Precision	0.4335	0.7791
Recall	0.4079	0.7666
F1 Score	0.4897	0.7890
Specificity	0.9750	0.9440

As shown in Table III, before preprocessing and hyperparameter optimizations the model was drastically inaccurate. After hyperparameter and preprocessing optimizations, all relevant metrics significantly improved.

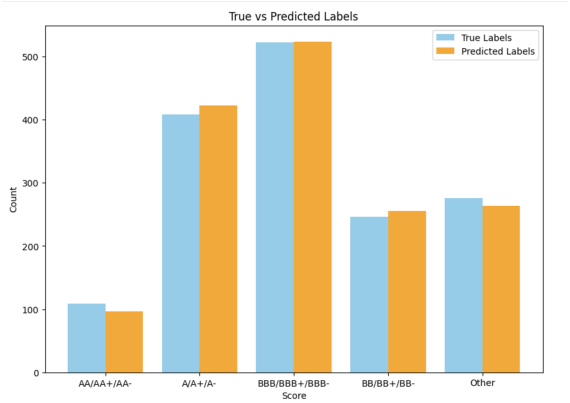

Fig. 9: True vs. Predicted for class labels

Figure 9 shows the true vs. predicted number of labels for each class. Shown by the figure, the values match closely, indicating a strongly performing model.

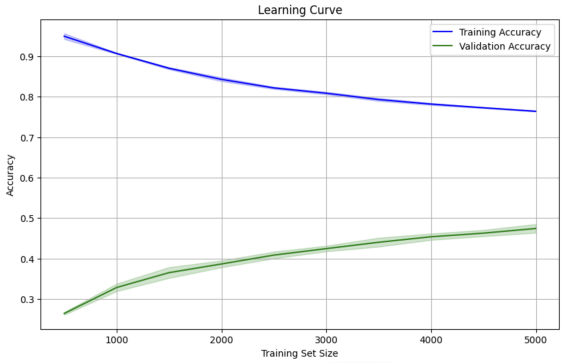

Fig. 10: Before Optimization

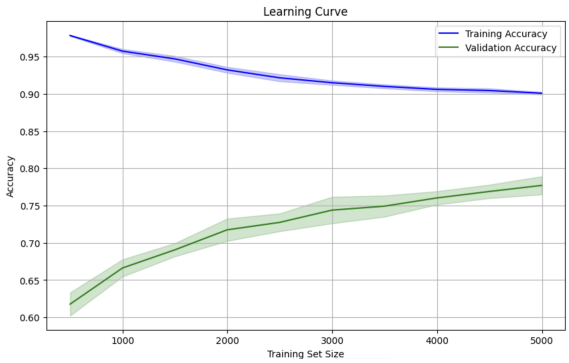

Fig. 11: After Optimization

Figures 10 and 11 show that before optimization, the model overfit severly, but after optimizations were made, the model stopped overfitting and validation accuracy started to approach training accuracy.

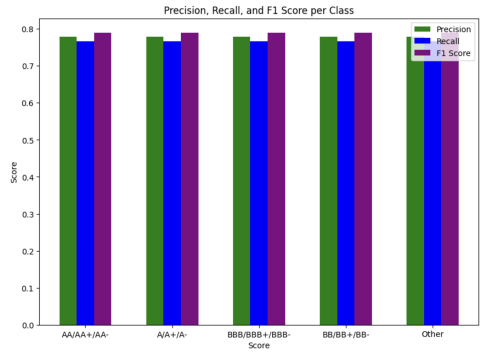

Fig. 12: Precision, Recall, and F1 Score Per Class

As shown in figure 12, precision, f1, and recall for all the classes are very similar values, indicating the model performs consistently across different classes.

VIII. Evaluation and Model Comparison

A. Evaluation Metrics

To comprehensively evaluate the performance of the models, we employed the following metrics:

- **Accuracy:** Measure of correct predictions among total predictions. This metric is particularly important for assessing the overall performance but may be biased in imbalanced datasets.
- **Precision:** Measures the ratio of true positive to the total predicted positives. This metric highlights the model's ability to avoid false positives.
- **Recall:** Measures the proportion of true positives among all actual positive cases. This metric emphasizes the model's ability to identify all relevant instances. Also known as sensitivity.
- **F1-Score:** Measures the harmonic mean of precision and recall. This metric is the most reliable even for unbalanced class distributions.

We utilized these performance metrics to ensure that our models performance is robust across various scenarios.

B. Comparative Analysis of Models

Table IV summarizes the performance of the three models based on the metrics discussed. The results indicate varying strengths and weaknesses across the models:

TABLE IV: Performance Measures for the Prediction Models.

Performance Measures	Accuracy	Precision	Recall	F1-score
Random Forest	78.92%	77.91%	76.66%	78.90%
Neural Network	74.05%	74.03%	74.05%	73.98%
KNeighbors Classifier	78.23%	78.86%	78.23%	78.15%

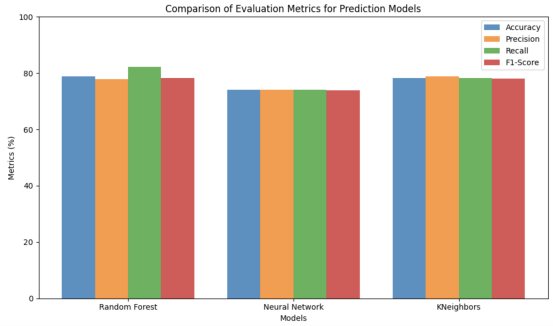

Fig. 13: Model Comparison Bar Chart

C. Insights and Final Model Selection

The Random Forest model emerged as the best-performing algorithm based on our performance metrics. This performance can be attributed to its data bagging and feature bagging approach which captures intricate patterns in complex data and also makes sure that model does not overfit.

However, the Neural Network model demonstrated significant potential in handling complex non-linear relationships between features after extended training over more epochs. It still can be considered as a scalable solution for larger and more diverse datasets. The KNeighbors classifier, while efficient for smaller subsets, struggled with scalability and computational efficiency for larger datasets.

Final Selection: The Random Forest model was selected as the final model. Its ability to handle imbalanced datasets and provide feature importance insights aligns well with the goals of this study. Our project's online deployment for real time use also played a major role in deciding this model as our final selection owing to its computational efficiency.

IX. Conclusion

In this study, we developed and evaluated three machine learning models—Random Forest, Neural Network, and K-Nearest Neighbors—for the task of corporate credit rating prediction using financial ratios.

The Random Forest model emerged as the most effective, achieving the highest overall accuracy and recall, which is critical for ensuring that true credit ratings are identified correctly. It handled imbalanced datasets effectively. Neural Networks also demonstrated its ability to capture complex non-linear relationship. This suggests that on scaling this model even further on more diverse dataset can potentially outperform other traditional models. The K-Nearest Neighbors classifier, while competitive in accuracy for smaller datasets, showed limitations in scalability, making it less suitable for large-scale datasets.

A key insight from this research is the importance of data preprocessing and exploratory data analysis to decide on factors such as number of training epochs, model architecture and target class labels. Even though hyperparameter tuning and regularization techniques showed promise, their impact was often limited compared to fundamental design choices.

This research underscores the importance of a system designed specailly for corporate companies to predict their credit score by leveraging their financial data and ratios derived from that raw data. By incorporating more diverse data about financial ratios and improving the automated solution by advanced algorithms, the proposed models can offer immense support to make more informed decision-making in the financial sector. Future work could explore ensemble approaches that combine the strengths of these models to achieve even higher predictive performance.

X. Project Roadmap and Team Contributions

A. GitHub Link

https://github.com/Programmer7129/ecs171

B. Project Roadmap

The final project submission consists of the following components:

- Research Paper/Final Report (8-10 pages).
- Web-based interface to deploy and interact with the models.
- Source Code, including preprocessing scripts and model implementation.
- 5-minute recorded project demonstration.

The development process has been structured into the following milestones:

1) **Week 8: Initial Development**
   - Dataset Description: Analyze and document attributes, types, and distribution.
   - Literature Review: Identify relevant methodologies and datasets.
   - Exploratory Data Analysis (EDA): Perform analysis, identify outliers, and preprocess data.
   - Model Development (Part 1): Develop preliminary models and evaluate initial results.
2) **Week 10: Optimization and Evaluation**
   - Model Optimization: Fine-tune hyperparameters and optimize preprocessing techniques.
   - Model Evaluation: Evaluate model performance using metrics such as accuracy and F1 score.
   - Front-End Development: Begin creating the web-based interface.
   - Conclusion: Draft preliminary conclusions based on results.
3) **Week 10: Finalization**
   - Finalize models and integrate with the web interface.
   - Record the project demonstration video.
   - Complete the research paper and submit all components.

C. Team Contributions

The team consists of the following members and their assigned tasks:

- **Abdulaziz Alhumaidy:** Implemented the KNN model, wrote the Introduction, did the Literature Review, and wrote major parts of the EDA.
- **Abdulah:** Focused on the web GUI in order to effectively display our models capabilities. Assisted in the feature selection process and identifying relevant financial data alongside the current distribution of current corporate credit ratings.
- **Vedant:** Focused on the development and optimization of the Neural Network model. This included everything from designing the initial model architecture to fine tuning the model's hyperparameters. Found out the correct database for training the model. Worked on the coding sections for exploratory data analysis. Additionally, conducted an extensive literature review by analyzing research papers and exploring relevant GitHub repositories to incorporate best practices and innovative techniques for all the 3 models used.
- **Vikram:** Implemented the Random Forest model, along with the preprocessing and hyperparameter optimizations which were needed on it. Wrote all sections regarding the Random Forest model along with the methodology.
- **Elijah:** Contributed the Data Description to the research paper. Focused on implementing the web GUI. This is so that our models can be demonstrated, feature values can be manually input by user, and results can be seen.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[2] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. O'Reilly Media, 2019.

[3] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[5] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[7] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[9] P. Golbayani, I. Florescu, and R. Chatterjee, "A comparative study of forecasting corporate credit ratings using neural networks, support vector machines, and decision trees," *The North American Journal of Economics and Finance*, vol. 54, p. 101251, 2020. doi: 10.1016/j.najef.2020.101251.

[10] J. Li, N. Mirza, B. Rahat, and D. Xiong, "Machine learning and credit ratings prediction in the age of fourth industrial revolution," *Technological Forecasting and Social Change*, vol. 161, p. 120309, 2020. doi: 10.1016/j.techfore.2020.120309.

[11] V. Moscato, A. Picariello, and G. Sperlí, "A benchmark of machine learning approaches for credit score prediction," *Expert Systems With Applications*, vol. 165, p. 113986, 2020. doi: 10.1016/j.eswa.2020.113986.

[12] B. Yu, C. Li, N. Mirza, and M. Umar, "Forecasting credit ratings of decarbonized firms: Comparative assessment of machine learning models," *Technological Forecasting and Social Change*, vol. 174, p. 121255, 2021. doi: 10.1016/j.techfore.2021.121255.

[13] R. Makwana, D. Bhatt, K. Delwadia, A. Shah, and B. Chaudhury, "Understanding and attaining an investment grade rating in the age of explainable AI," *Computational Economics*, 2024. doi: 10.1007/s10614-024-10700-7.

[14] Vens, C. (2013). Random Forest. In: Dubitzky, W., Wolkenhauer, O., Cho, KH., Yokota, H. (eds) Encyclopedia of Systems Biology. Springer, New York, NY. 10.1007/978-1-4419-9863-7_612.

[15] Sarthak96agarwal, "GitHub - sarthak96agarwal/Corporate-Credit-Rating-Prediction-: Predicting the credit ratings of US based companies using machine learning models," GitHub. [Online]. Available: https://github.com/sarthak96agarwal/Corporate-Credit-Rating-Prediction-/tree/master.

[16] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). 10.1023/A:1010933404324.