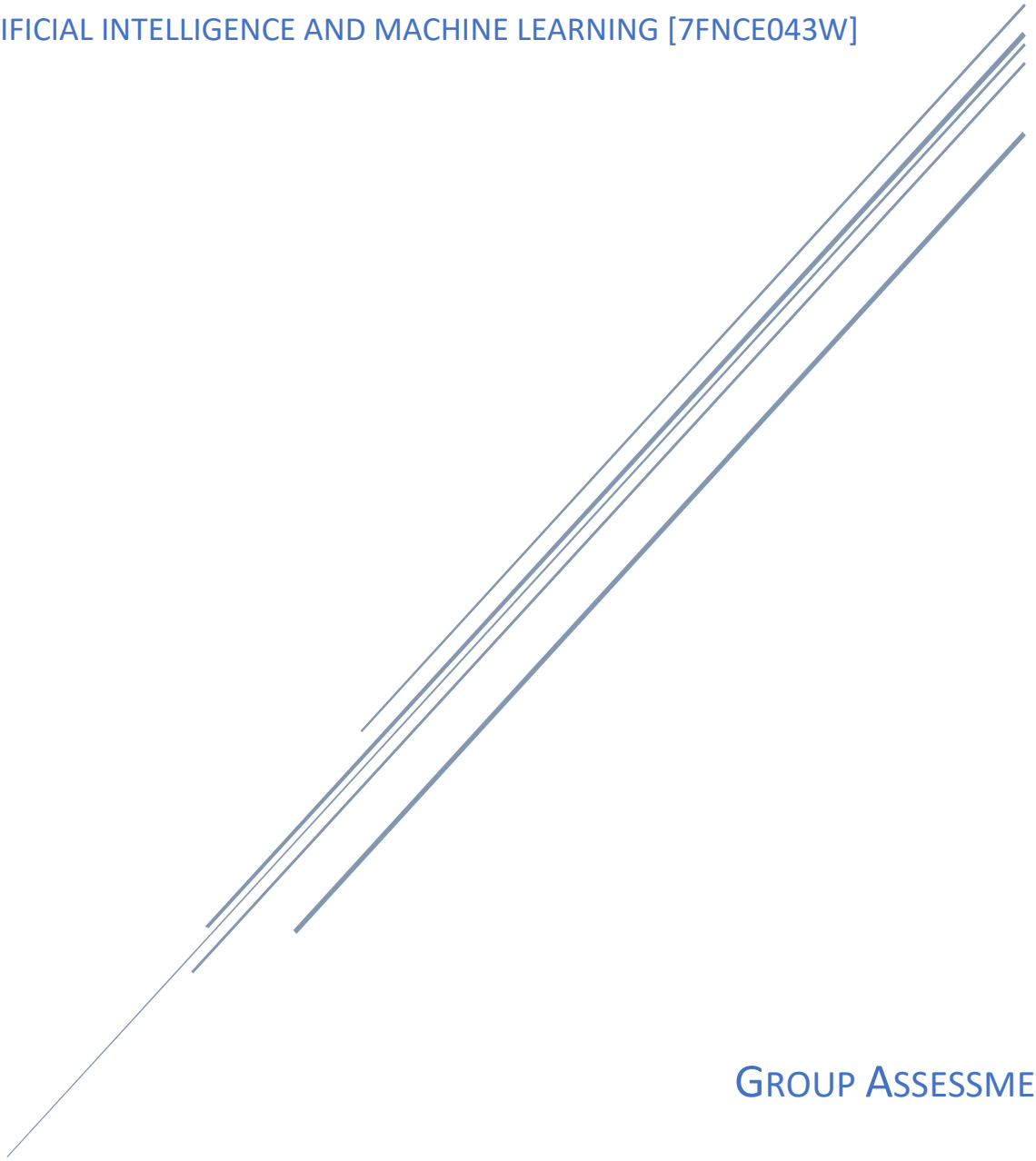
UNIVERSITY OF WESTMINSTER

MACHINE LEARNING IN CREDIT SCORE PREDICTION

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING [7FNCE043W]



GROUP ASSESSMENT

WORD COUNT: 3263, EXCLUDING COVER PAGE, TABLE OF CONTENTS AND FIGURES, ABSTRACT, FORMULAS, APPENDIX AND PEER REVIEW

Table of Contents

Figures

ABSTRACT

This report assesses the application of machine learning techniques for predicting credit level of UK firms using data from 2019 to 2020. After conducting explanatory data analysis and feature selection, the analysis applies AdaBoost and Random Forest classification methods to predict firms' credit levels. Results from the models indicate that both models accurately predict the credit level with Random Forest performing better at 99.13%. The report also highlights the challenges associated with machine learning such as overfitting, interpretability, and explainability. It suggests innovative approaches such as incorporating macro-economic variables and employing heterogeneous base classifiers in ensemble methods to improve generalization and robustness. The integration of Explainable Artificial Intelligence (XAI) techniques is also recommended to balance accuracy with transparency, providing a pathway to more dynamic and reflective credit scoring models that can adapt to real-world financial risks.

1. Introduction

Credit risk assessment is a critical tool for financial institutions as it is fundamental to base strategic and commercial decision-making processes. Discussing the importance of credit risk assessment, Amato et al (2022) highlight the importance of this practice in identification and evaluation of potential risks in lending, guide risk management practice and support the categorisation of creditworthiness of borrowers. This in turn leads to profit maximisation through balancing risk and return. It will also aid in customer/entity segmentation to develop customised offerings as well as adherence to regulatory compliance requirements.

This report explores the credit status and characteristics of firms in the UK from 2019 to 2020, with the aim of predicting the credit score level of these firms using machine learning models.

The dataset provided, "EM.csv", serves as the foundation for this analysis. It includes features such as credit score rating, various numerical and financial data along with categorical data. The primary objective is to conduct EDA, followed by creation of a new binary feature, 'CreditLevel' which categorises firms based on their credit scores to predict the 'CreditScore' higher or lower than the mean based on various explanatory features. For this purpose, two supervised machine learning methods have been employed and compared to achieve maximum accuracy in predicting the target variable. The report is concluded with the opinion that whether machine learning can or cannot predict the 'CreditLevel' of firms based on given dataset.

2. Exploratory Data Analysis of Features

The foundational principles of Exploratory Data Analysis (EDA), as outlined by Mukhiya and Ahmend (2020), underscore its importance in uncovering insights from raw data. EDA involves examining datasets to reveal patterns, anomalies, and underlying structures, thereby facilitating hypothesis generation, and informing subsequent modelling. EDA plays an important role in providing descriptive summaries, statistical analysis, and visualisation.

Mukhiya and Ahmend (2020) provide four steps in EDA processing which are problem definition, data preparation, data analysis and development and representation of results. This provides a structured approach in setting the stage for focused analysis.

In this context, the problem is identified as exploring the relationship between 'CreditScore' with various feature variables such as 'return on capital employed', 'likelihood of failure', 'credit limit GBP' 'EBITDA' and 'number of employees' over the years 2019 and 2020.

For the data analysis step of development and representation, Python data visualisation libraries were employed to generate a variety of visualization, including histograms, scatter plots, and line plots. These tools summarise the data and show the relationship between 'credit score' and other financial metrics mentioned above.

In the data preparation stage, data has been pre-processed, including creating new binary feature, handling missing values by filling NaN values with the feature mean, dropping the non-numerical columns from the dataset i.e., 'Creditscoreindicator' and 'SMEindicator', normalising data ranges by applying StandardScaler to features. The findings of EDA are discussed as follows:

2.1 Summarising the EM.csv Data

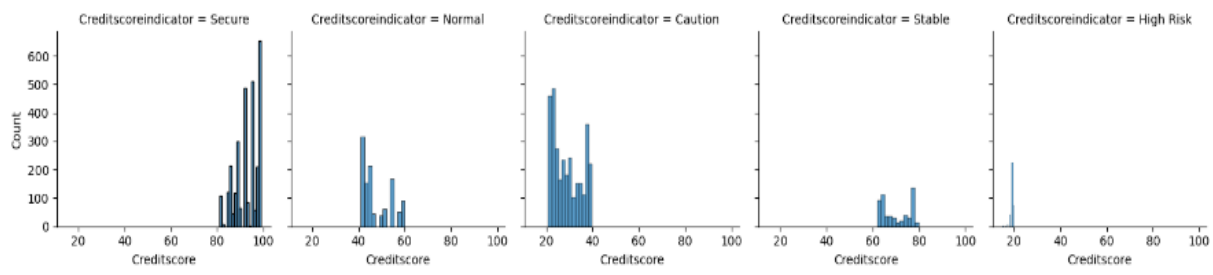
The data has a total of 8176 entries and 72 columns, with a mix of numerical and categorical data types. The key metrics for the report include 'credit score' which has an average of 57.3 with std_dev of 29.73, suggesting a range of scores form a minimum of 15 to a maximum of 99. For the 'likelihood to failure', mean is around 5.22 with values ranging from 0.9 – 19.4. Lastly, the 'Creditlimit in GBP' has a large average of around 529,300 GBP and the std_dev indicating a wide dispersion, with limits ranging from as low as 500GBP to as high as 50mn GBP. (Refer to annex 1).

2.2 Analysis of Visual Outputs

2.2.1 'Credit Score' by 'Credit Score Indicator'

FacetGrid histograms were generated to visualise the distribution of credit scores across different credit score indicators which are 'secure', 'normal', 'caution', 'stable' and 'high risk'. Figure 1 shows that when 'CreditScore' is high the firm is categorised as secure whereas it is categorised as 'caution' and 'high risk' at the lower end of 'CreditScore'.

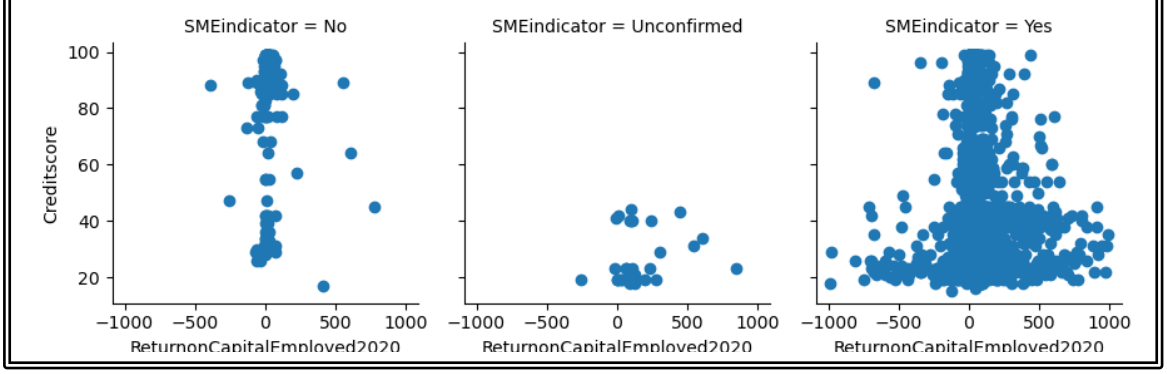
Figure 1: CreditScore by CreditScoreIndicator



2.2.2 'Credit Score' and 'Return on Capital Employed 2020' Grouped By 'SME Indicator'

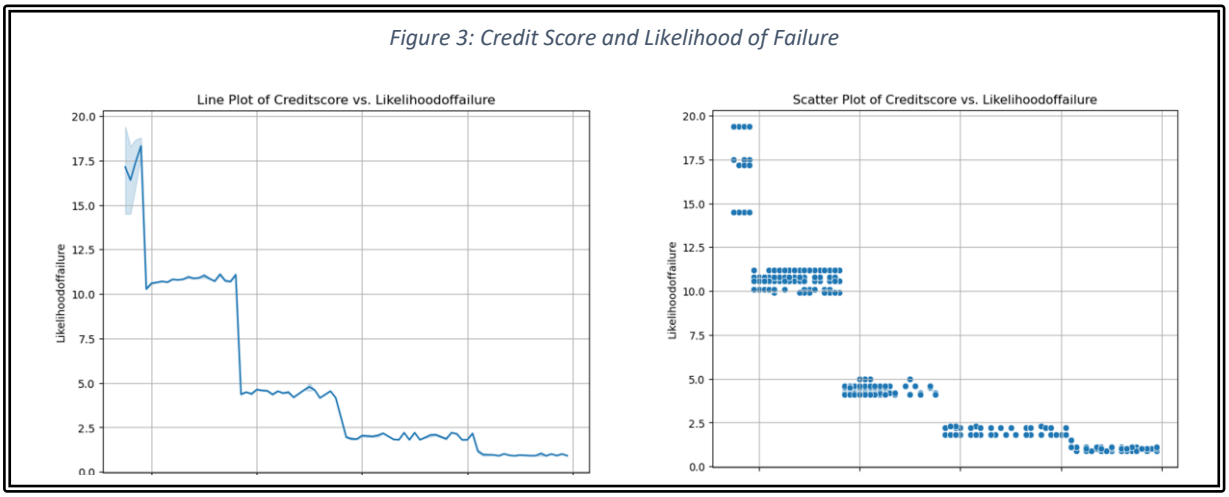
Scattered plots were used to explore the relationship between 'CreditScores' and 'ReturnonCapitalEmployeed2020' segmented by 'SMEindicators'. For non-SMEs, the data points form a small cluster implying a weak correlation between return on capital and credit score. The unconfirmed SME status show a scattered distribution suggesting less defined relationship. The confirmed SMEs, however, show a wide dispersion of data points, implying a strong correlation between the two features.

Figure 2: Credit Score, Return on Capital Employed 2020 segmented by SME Indicator



2.2.3 'Credit Score' and 'Likelihood of Failure'

Figure 3: Credit Score and Likelihood of Failure

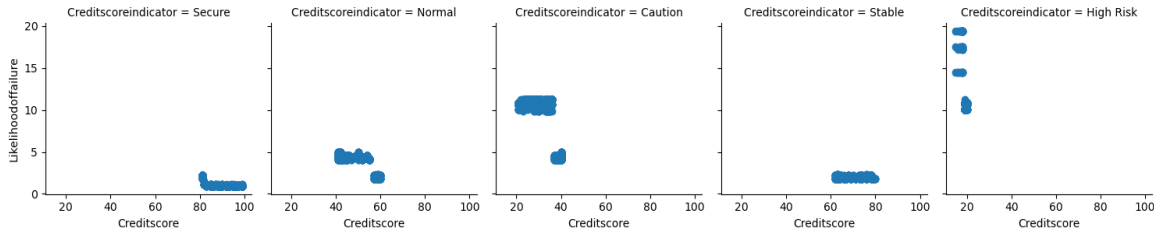


The line graph and scatter plot illustrate an inverse relationship between the two features. As the credit score increases, the risk of failure is lowered. In contrast low credit score implies high risk of failure.

Further visualisation was employed to examine the relationship between credit score and likelihood of failure across different credit score indicators. Like figure 3, the entities with high credit score show

low likelihood of failure and are considered secure. At the extreme end, high risk category has low credit score indicating higher likelihood of failure.

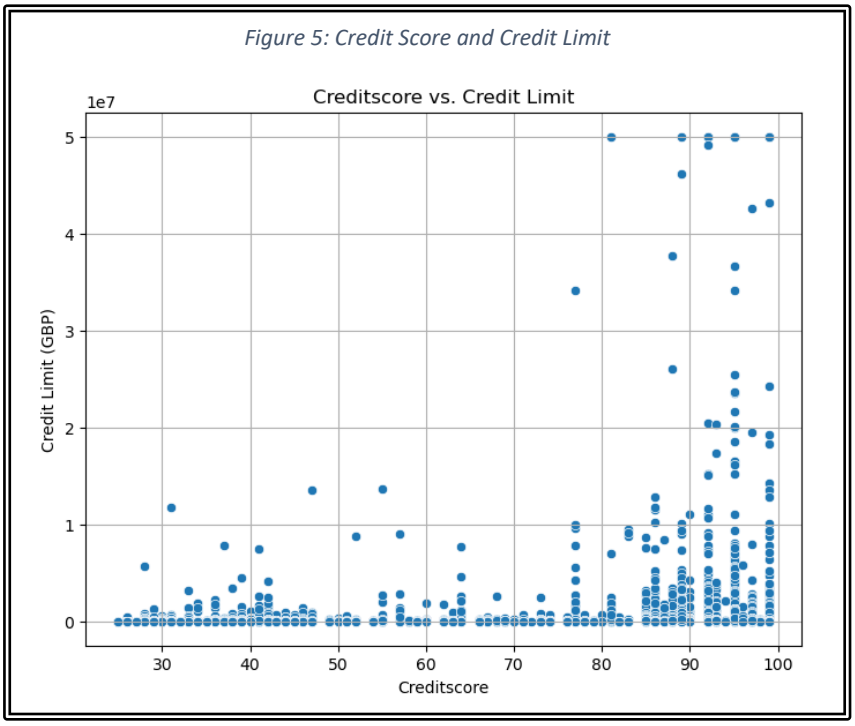
Figure 4: Credit Score, Likelihood of Failure and Credit Score Indicator



2.2.4 'Credit Score' and 'Credit Limit GBP'

The plot illustrates how credit scores, and credit limits are related, indicating the inclination for higher credit scores to correlate with increased access to credit. Additionally, the plot shows there is dense concentration of data points at the lower end of the credit limit scale which decreases as the credit limit increases. Notably, there are outliers with high credit limits at some data points.

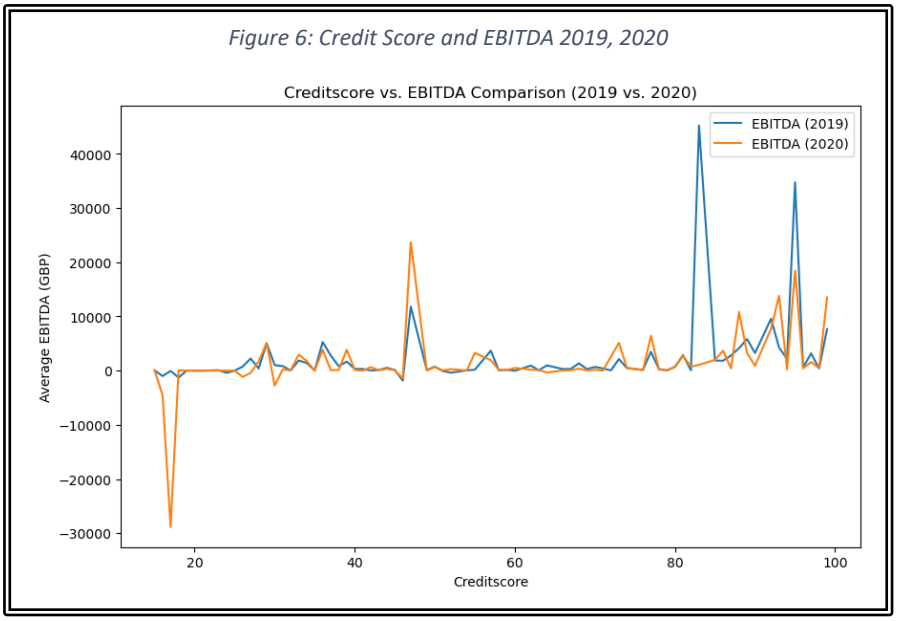
Figure 5: Credit Score and Credit Limit



2.2.5 'Credit Score' and EBITDA Comparison (2019 and 2020)

The graph provides a visual analysis of how average EBITDA correlates with credit scores over the two periods offering an insight into the financial performance trends of the entities. Compared to 2019, the 2020 EBITDA appear more volatile with higher peaks and lows. Furthermore, some credit score ranges show negative EBITDA values especially in the lower credit score range. This graph shows non-linearity between the two features.

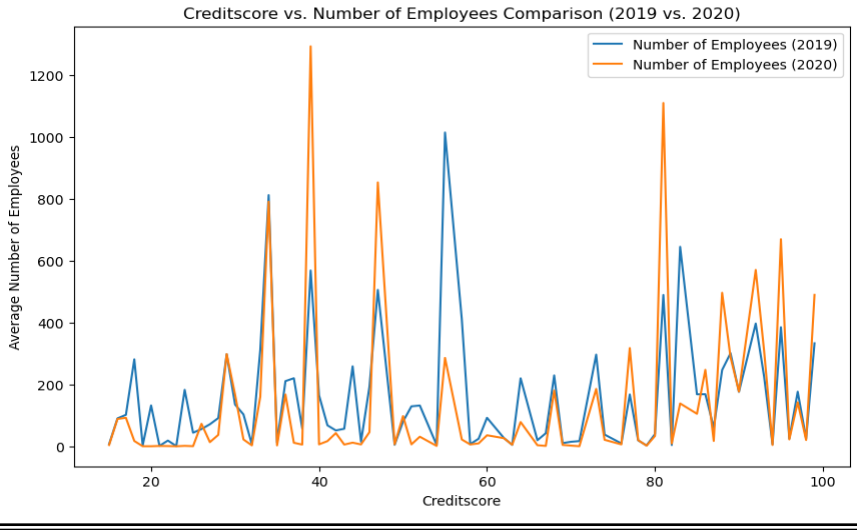
Figure 6: Credit Score and EBITDA 2019, 2020



2.2.6 'Credit Score' and Employees Comparison (2019 and 2020)

In this comparison, fluctuations are observed for both years suggesting varying employee counts across different credit score ranges. As indicated in figure 7, some credit score ranges have more employees in 2020 compared to 2019, while other ranges have fewer employees. Outliers are seen with high or low employee counts indicating entities that depart significantly from the average number of employees expected for their respective credit score range.

Figure 7: Credit Score and Number of Employees Comparison 2019, 2020



From the above EDA analysis, it is observed that the 'CreditScore' is highly related to the 'LikelihoodofFailure' and 'CreditLimitGBP' however, firm size as indicated by 'NumberofEmployees' does not have any impact on the 'CreditScore' on the entities.

2.3 Feature Creation

Feature creation, or feature engineering, enhances machine learning model performance by generating new features or extracting meaningful information from existing ones.

The first step involves calculating the mean of the 'Creditscore' feature from the dataset that provides a reference point for determining whether individual 'Creditscore' values are above or below the average. Subsquently, a new feature, 'CreditLevel', is then derived from 'Creditscore'.

- If the 'Creditscore' value is greater than the mean, the 'CreditLevel' is set to 1, indicating a high credit score.
- If the 'Creditscore' value is less than or equal to the mean, the 'CreditLevel' is set to 0, indicating a low credit level.

Following feature creation, 'CreditScore' is removed from the dataset to avoid redundancy. Subsequently, the dataset was split into feature variables (X), and 'CreditLevel' designated as the target variable (Y), representing the variable to be predicted by the machine learning models.

2.4 Data Preprocessing

2.4.1 Extracting and Keeping Numerical Columns

As part of data preprocessing, initially the non-numerical columns have been identified from the dataset. The identified non-numerical columns are 'Creditscoreindicator' and 'SMEindicator' which were then dropped from the dataset ‘X’. This step removes irrelevant or redundant features that are not required for the subsequent analysis.

```
Non-numeric columns:
Index(['Creditscoreindicator', 'SMEindicator'], dtype='object')
```

2.4.2 Filling missing values with feature means

Missing values in the dataset are filled with the mean of each respective feature. This strategy is a common approach to handling missing data and ensures that all features have valid values for further processing (Yao, 2024).

2.4.3 Splitting the data into training and testing sets

The data is divided into training and testing sets using scikit-learn's train_test_split function. Specifically, 30% of the data is allocated to the test set, and the remaining 70% for the training set. Additionally, the parameter random_state is set to '123' to ensure reproducibility of the results that

the data is split in the same way each time the code is executed, which is important for obtaining consistent and comparable results across different runs.

2.4.4 Fitting StandardScaler on the training data

The training data undergoes standardization using the 'StandardScaler' function, scaling features to a normal distribution with mean 0 and standard deviation 1. This ensures uniform scale across all features. The fitted StandardScaler transforms both training and testing sets, maintaining consistency between them by applying the same scaling parameters learned from the training data, thereby preparing them for modelling.

2.5 Correlation Matrix

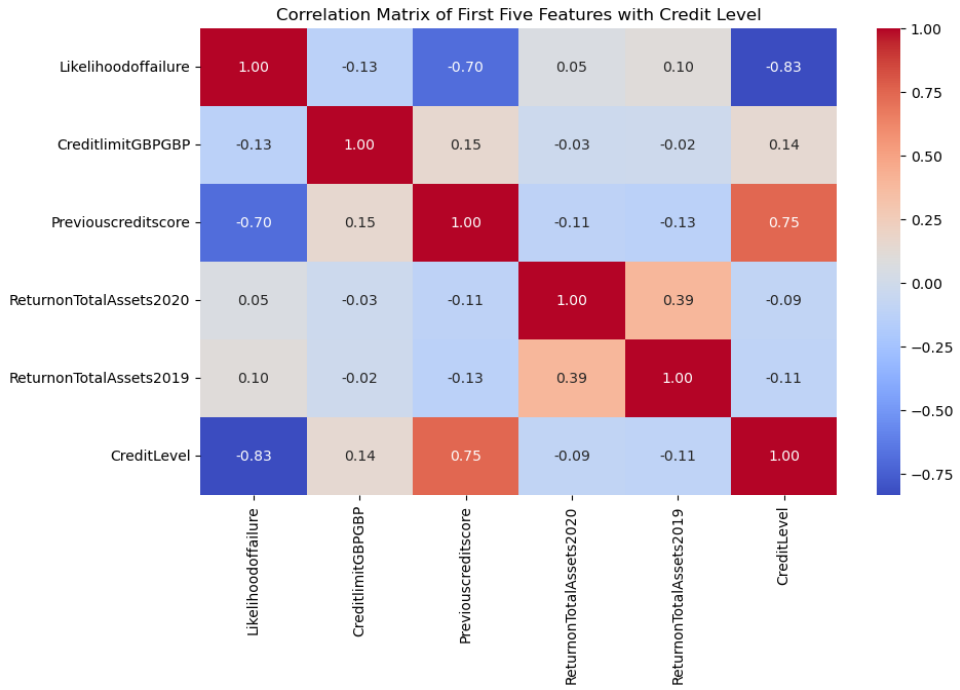
The correlation heatmap shows the correlation among different features. The features are: CreditLevel, Likelihoodoffailure, CreditlimitGBPGBP, Previouscreditscore, ReturnonTotalAssets2020, ReturnonTotalAssets2019.

The values in the heatmap range from -1 to 1. A value of 1 indicates a strong positive correlation, -1 indicates a strong negative correlation, and 0 indicates no correlation.

The following are some of the key correlations in the heatmap:

- CreditLevel is highly negatively correlates with Likelihoodoffailure (-0.83), suggesting a decrease in Likelihoodoffailure as CreditLevel rises.
- CreditLevel is highly positively related to Previouscreditscore (0.75), indicating a close relationship between current credit levels and past credit scores.
- Likelihoodoffailure is highly negatively correlated with Previouscreditscore (-0.70), implying lower likelihood of default when previous credit scores are high.
- ReturnonTotalAssets 2019 and 2020, and CreditlimitGBPGBP show no significant correlation with other features, suggesting minimal impact on CreditLevel.

Figure 8: Correlation Matrix

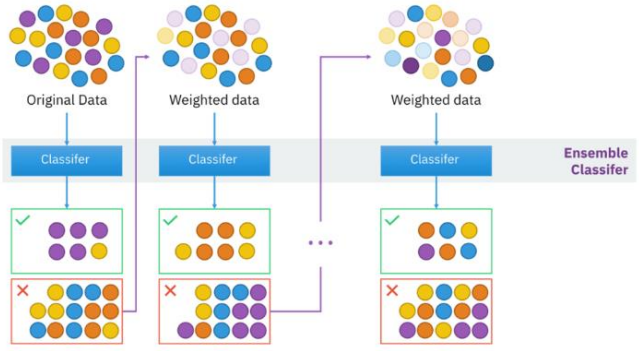


3. Machine Learning Classification Methods

3.1 AdaBoost Classification

AdaBoost (Adaptive Boosting), is a supervised ensemble machine learning technique. AdaBoost works by iteratively training a sequence of weak learners and adjusting the weights of instances in the training set based on the accuracy of previous classifications. Misclassified instances receive higher weights, while correctly classified ones receive lower weights, enabling subsequent learners to concentrate on difficult-to-classify instances. For final result predictions from all weak learners are combined using a weighted majority, with weights determined by their performance during training. (scikit-learn.org, n.d.).

Figure 9: Illustration of AdaBoost Classification Process



Source: (almabetter, n.d.)

For this dataset, the 'n-estimators' parameter has been set to 50 that specifies the number of weak learners to be used in the ensemble. The random-state parameter is set as '123' to ensure reproducibility of the results. The AdaBoost classifier is then trained on the scaled training data (X) and the trained target variable (Y), which is then used to make predictions on scaled test data.

The outcome of AdaBoost classification on the given dataset achieved an accuracy of 0.9922 on the test set, indicating that the classifier correctly predicted the 'CreditLevel' (0 or 1) for 99.22% of the instances in the test set.

```
Accuracy of AdaBoost classifier: 0.9922543823889115
```

The subsequent classification report provides more detailed metrics for each class in the target variable.

- For class 0 (firms with 'Creditscore' lower than the mean), the precision of 0.99, measuring the fraction of true negatives among predicted negatives, shows that the model AdaBoost has a low false positive rate for class 0. The recall, which measures the ratio of correctly predicted instances to the total instances, is 0.99, suggesting that it effectively captures most of the instances for credit level. The F1-score value of 0.99, as the harmonic mean of precision and recall, indicates good overall performance, with a support of 1356 instances for class 0 in the test dataset.

- For class 1 (firms with 'Creditscore' lower than the mean), the precision is 0.99, the recall is 0.99, and the F1-score is 0.99, with a support of 1097 instances indicating a good overall performance for class 1 in the test dataset.

Both macro and weighted averages are approximately 99%, indicating high performance across all classes. These are the average values of precision, recall, and F1-score, where the macro average considers each class equally, while the weighted average considers the number of instances of each class.

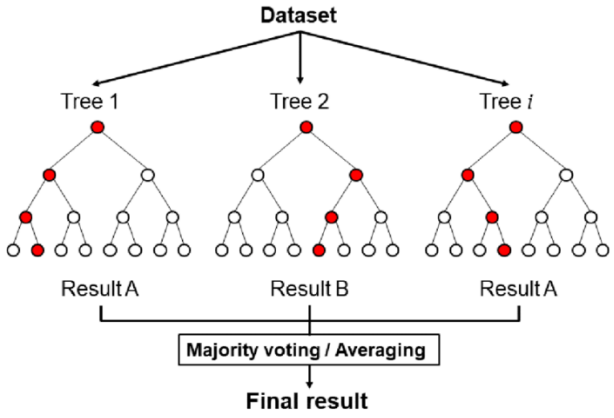
```
Classification Report of AdaBoost classifier:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1356
           1       0.99      0.99      0.99      1097

    accuracy                           0.99      2453
   macro avg       0.99      0.99      0.99      2453
weighted avg       0.99      0.99      0.99      2453
```

This suggests that the AdaBoost classifier, along with the selected features (X), was able to effectively distinguish between firms with 'Creditscore' higher or lower than the mean, based on their 'CreditLevel' target variable, and makes the accurate predictions.

3.2 Random Forest Classification

A random forest classifier, or random decision forest classifier, is an ensemble method for supervised machine learning that produces multiple decision trees, using a randomly selected subset of training samples and variables. It works by constructing multitude of decision trees during training and outputting the class, that is most frequent or by averaging of the classes, predicted by individual trees, to improve the predictive accuracy and control over-fitting (scikit-learn.org, n.d.).

Figure 10: Illustration of Random Forest Classification Process



Source: (Christiansen, 2021)

Random forest Classifier is known for its robustness to overfitting and can be used for solving regression (numeric target variable) and classification (categorical target variable) problems.

Similar to AdaBoost, the Random Forest classifier sets n_estimators to 50, here determining the number of decision trees in the ensemble, and random state to 123 for result reproducibility. Trained on scaled features (X) and target variable (Y), it predicts test data outcomes.

The output of Random Forest Classifier achieved an accuracy of 0.9931, indicating that the classifier correctly predicted the 'CreditLevel' (whether 0 or 1) for 99.31% of the instances in the test set.

```
Accuracy of Random Forest classifier: 0.9930697105584998
```

The classification report of Random Forest for each class (0 and 1) in the target variable 'CreditLevel' is as below:

- Class 0 (below mean 'Creditscore') has precision, recall, and F1-score at 0.99, with support of 1356 instances.
- For class 1 (above mean 'Creditscore') shows precision, recall, and F1-score at 0.99, with a support of 1097 instances.

Where both the macro and weighted averages are 0.99 for precision, recall, and F1-score.

```
Classification Report of Random Forest classifier:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1356
           1       0.99      0.99      0.99      1097

    accuracy                           0.99      2453
   macro avg       0.99      0.99      0.99      2453
weighted avg       0.99      0.99      0.99      2453
```

The Random Forest Classifier has performed well in predicting the 'CreditLevel' by achieving high precision, recall, and F1-score for both classes, indicating that the Random Forest algorithm, along with the selected features (X), was able to effectively distinguish between firms with 'Creditscore' higher or lower than the mean.

3.3 Other Classification Methods

Besides the above classifiers, two more classification methods were employed on the given dataset, i.e. Support Vector Machine and Logistic Regression.

3.3.1 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It works by finding the optimal hyperplane SVM aims to find the hyperplane that maximizes the margin between the classes. The hyperplane is the decision boundary that best separates the classes in the feature space. SVM is effective in high-dimensional spaces and is versatile in handling both linearly separable and non-linearly separable data using different kernel functions (Kanade, 2022). The SVM produced accuracy rate of 98.899% for predicting the data in the test set.

```
Accuracy of SVM classifier: 0.9889930697105584
```

3.3.2 Logistic Regression

The logistic regression classification is used when the target variable is categorical/binary. It models the probability that a given input belongs to a particular class based on a decision boundary, which

separates the positive class (1) from the negative class (0). The decision boundary is determined by the optimal weights learned during training that minimize the difference between the predicted probabilities and the actual class labels in the training data. The output of logistic regression is a probability score between 0 and 1 (Kanade, 2022). The Logistic Regression classification produced an accuracy rate of 98.94% of predicting the data.

```
Accuracy of Logistic Regression classifier: 0.9894007337953526
```

Since, **the accuracy rate of AdaBoost and Random Forest classifiers is higher than SVM and Logistic Regression classifiers**, therefore, further analysis have been performed AdaBoost and Random Forest only. The classification report of all classification methods is mentioned in the Appendix.

4. Feature Importance of the Chosen Classification Methods

Feature importance indicates how valuable or influential each feature is in the model's prediction process. The plot is created using a horizontal bar chart, where the x-axis represents the feature importance values, and the y-axis displays the feature names. The bars are arranged in descending order of importance of each feature in the decision-making process, with the most important feature at the top (Shin, 2023).

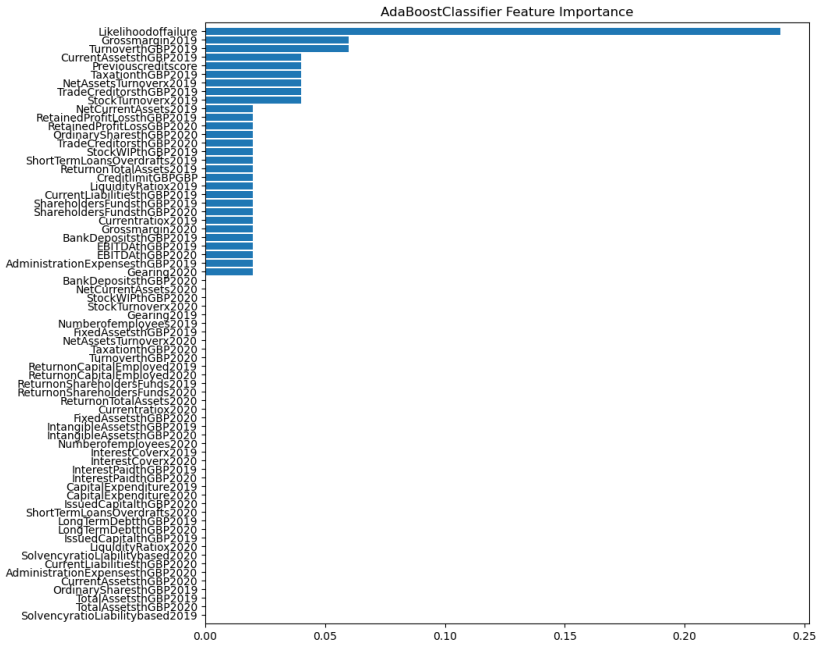
4.1 Feature Importance of AdaBoost

Figure 11 displays the feature importance plot for the AdaBoost classifier, trained on numerical variables from the dataset. AdaBoost determines feature importance based on the average contribution of each base classifier (Lejafar, 2018).

Among 70 features, only 29 are deemed important for predicting the target variable. The top five features crucial for predicting 'CreditLevel' by AdaBoost are listed below, indicating their significant contribution to the model's predictions.

1. 'Likelihoodoffailure'
2. 'Previouscreditscore'
3. 'TurnoverthGBP2019'
4. 'CurrentAssetsthGBP2019'
5. 'Previouscreditscore'

Figure 11: AdaBoost Classifier Feature Importance



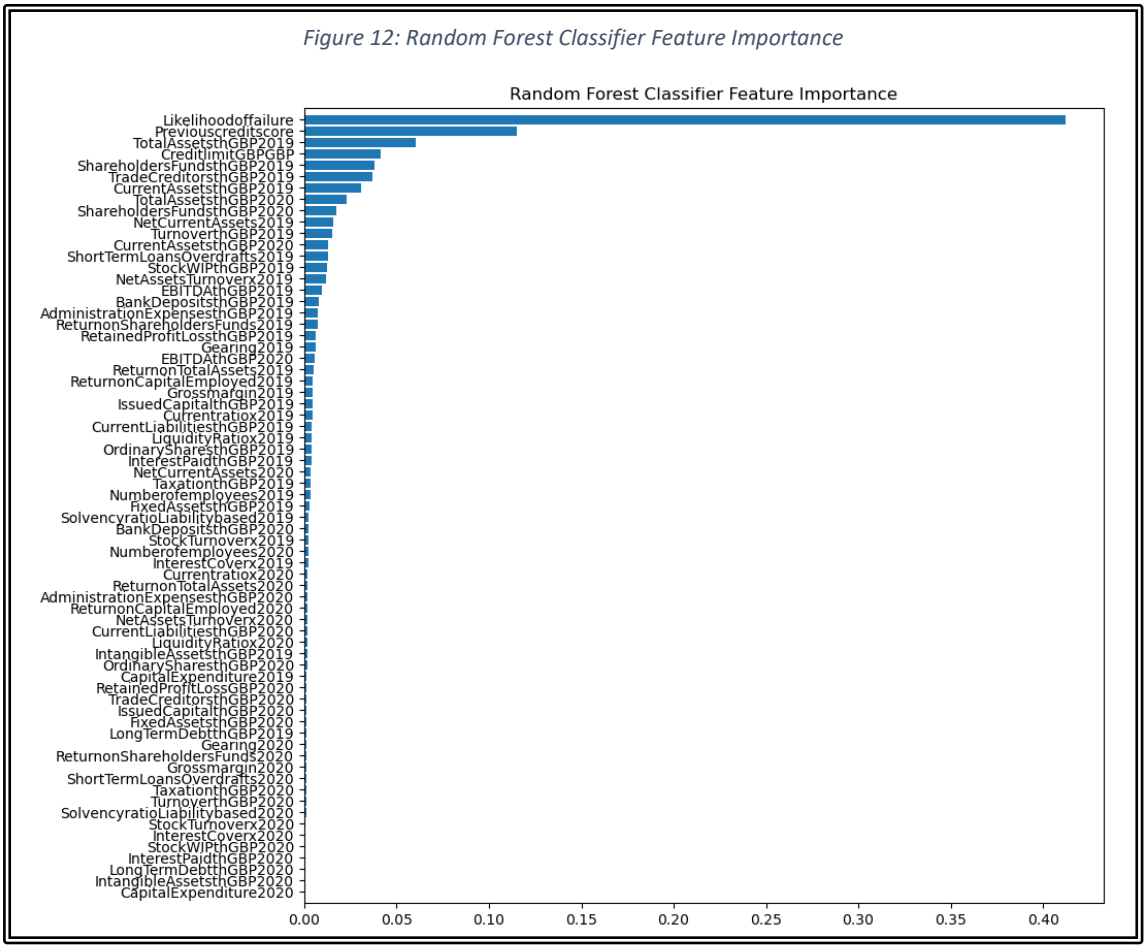
4.2 Feature Importance of Random Forest Classification

The Random Forest algorithm assesses feature importance by measuring the reduction in node impurity when splitting on a feature. Features that effectively decrease impurity are deemed more important (Filho, 2023).

While nearly all features contribute to prediction, many have minimal importance. The top five crucial features for the Random Forest classifier in predicting 'CreditLevel' are:

1. 'Likelihoodoffailure'
2. 'Previouscreditscore'
3. 'TotalAssetsthGBP2019'
4. 'CreditlimitGBPGBP'
5. 'ShareholdersFundsthGBP2019'

Figure 12: Random Forest Classifier Feature Importance

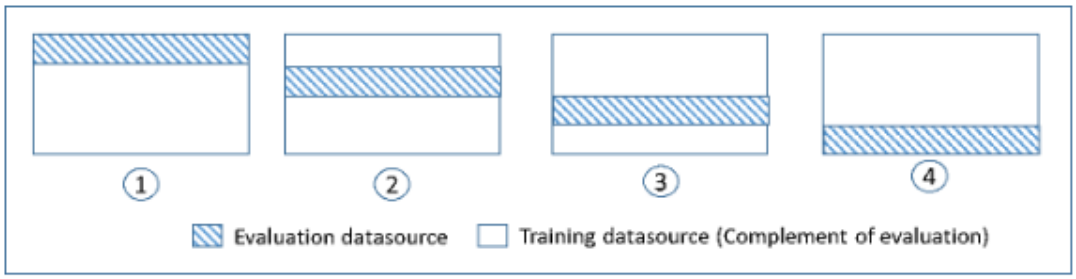


It is observed that both AdaBoost and Random Forest considers 'Likelihoodofffailure' and 'PreviousCreditScore' as most important feature, while the importance measurement for rest of the features varies between both models.

5. Cross-validation of the Classification Methods:

Cross-validation evaluates a model's performance and generalisation ability over unseen data by splitting the data into subsets or folds for training and validation. The model is then trained on a subset of the data (training set) and evaluated on the remaining subset (validation set). This process is repeated by rotating the training and validation sets across all the folds, and the performance metric (e.g., accuracy) is averaged over all the iterations (AWS, n.d.). Cross-validation helps prevent overfitting and estimates how well the model will perform on new data.

Figure 13: Cross-Validation Process



Source: (AWS, n.d.)

Both models employ 5-fold cross-validation, dividing training data into 5 subsets. The model trains on 4 folds and evaluates on the remaining fold, rotating the validation set each time. Mean accuracy is calculated from accuracy scores across all folds.

5.1 Cross-Validation of AdaBoost

The AdaBoost classifier demonstrates a mean accuracy of approximately 99.06% across all folds, indicating accurate predictions of the target variable (CreditLevel) for around 99.06% of instances in the training data.

The standard deviation of the accuracy scores is approximately 0.0026, representing the minimal variability of the accuracy scores across different folds, suggesting that the model's performance is consistent across different subsets of the training data.

```
Cross-validation scores for AdaBoost classifier: [0.98951965 0.99126638 0.9930131  0.98601399 0.99300699]
Mean accuracy of AdaBoost classifier: 0.9905640211317067
Standard deviation of accuracy of AdaBoost classifier: 0.0026173624897451817
```

5.2 Cross Validation of Random Forest

The cross-validation score of Random Forest classification ranges from approximately 0.9886 to 0.9948. The mean accuracy is approximately 0.9921, indicating that, on average, the classifier correctly predicts the target variable with an accuracy of around 99.21%. The standard deviation of the accuracy scores is approximately 0.0023, indicating relatively low variability and consistent performance of the classifier across different subsets of the data.

```
Cross-validation scores for Random Forest classifier: [0.99039301 0.99475983 0.9930131  0.98863636 0.99388112]
Mean accuracy of Random Forest classifier: 0.9921366842764223
Standard deviation of accuracy of Random Forest classifier: 0.002279678051967685
```

The high mean accuracy score (close to 1.0) and low standard deviation suggest that the Random Forest classifier is likely to fit well to new, unseen data and can be considered a reliable model for making predictions.

6. Mean Absolute Error of the Classifiers

The Mean Absolute Error (MAE) measures the average absolute difference between the predicted values and the actual values (Schneider & Xhafa, 2022). A lower MAE value indicates better performance, with a perfect score being 0. The MAE value can be calculated as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Where:
n: number of observations
y_i: the actual value of the i^{th} observation
$\hat{y}_i$: the predicted value of the i^{th} observation

```
Mean Absolute Error of AdaBoost classifier: 0.0077456176110884635
Mean Absolute Error of Random Forest classifier: 0.006930289441500204
```

For the AdaBoost classifier, the MAE is approximately 0.0077, which means that, on average, the classifier misclassifies about 0.77% of the instances in the test set.

For the Random Forest classifier, the MAE is approximately 0.0069, which means that, on average, the classifier misclassifies about 0.69% of the instances in the test set.

Comparing the two models, the Random Forest classifier has a slightly lower MAE, suggesting that, on average, the Random Forest classifier's predictions are slightly closer to the true values than those of the AdaBoost classifier.

7. Evaluation of the Classification Methods: AdaBoost and Random Forest

Based on the results, both AdaBoost and Random Forest can be compared for their performance on predicting the 'CreditLevel'.

Accuracy

- AdaBoost classifier: 0.9922 (99.22%)
- Random Forest classifier: 0.9931 (99.31%)

Both classifiers achieved very high accuracy scores on the test set, with the Random Forest classifier performing slightly better than the AdaBoost classifier.

Classification Report

The classification reports for both classifiers show similar performance metrics:

- Precision: 0.99 for both classes (0 and 1)
- Recall: 0.99 for both classes (0 and 1)
- F1-score: 0.99 for both classes (0 and 1)

These scores indicate that both models are capable of accurately classifying instances into the two classes ('CreditLevel' 0 or 1) with high precision and recall.

Cross-Validation

- AdaBoost classifier:

  Mean accuracy: 0.9906 (99.06%)

  Standard deviation of accuracy: 0.0026

- Random Forest classifier:

  Mean accuracy: 0.9921 (99.21%)

  Standard deviation of accuracy: 0.0023

The cross-validation results show that the Random Forest classifier achieved a slightly higher mean accuracy (99.21%) compared to the AdaBoost classifier (99.06%) on the training data. Additionally, the Random Forest classifier had a slightly lower standard deviation of accuracy, suggesting more consistent performance across the different folds.

Mean Absolute Error

- AdaBoost classifier: 0.0077
- Random Forest classifier: 0.0069

The Random Forest classifier had a lower MAE compared to the AdaBoost classifier, indicating a slightly lower average misclassification rate on the test set.

Overall, the Random Forest classifier consistently showed slightly better performance across all evaluation metrics, demonstrating a slight advantage over the AdaBoost classifier in terms of overall predictive power and generalization ability on this dataset and problem. Therefore, Random Forest has been opted for the prediction of target variable.

8. Predicting the target variable with X_Test data: Random Forest Classifier

First, the Random Forest classifier is trained, where 'X_train_scaled' variable contains the features (input data) for training, and 'Y_train' contains the corresponding target labels. Once trained, the classifier is used to make predictions on the test data using the predict () method. The predicted labels are stored in the 'random_forest_preds' variable.

8.1 Classification Report

As shown in Section 3.2, a classification report is generated to evaluate the performance of Random Forest classifier on the test data using various metrics. The accuracy score (99.31%) of the classifier is determined by comparing the predicted labels with true labels. Finally, a detailed classification report is generated, including metrics such as precision (0.99), recall (0.99), F1-score (0.99), and support for each class.

Classification Report of Random Forest classifier:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	1356
1	0.99	0.99	0.99	1097
accuracy			0.99	2453
macro avg	0.99	0.99	0.99	2453
weighted avg	0.99	0.99	0.99	2453

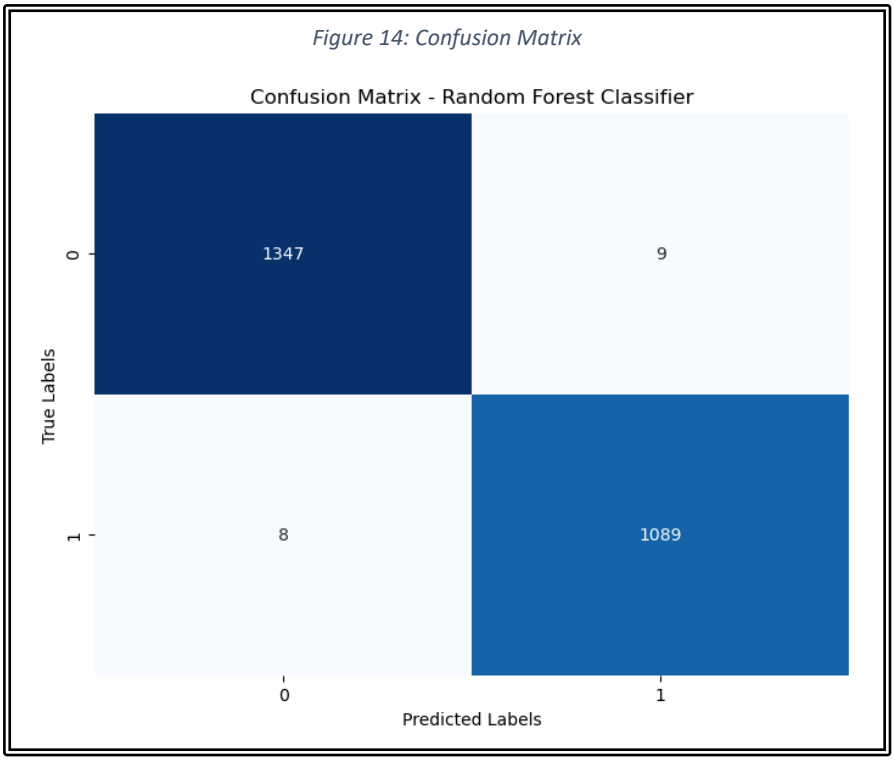
8.2 Confusion Matrix

Confusion matrix is a tool that evaluates how well a classification algorithm performs. It provides a visual summary of the algorithm's performance by comparing predicted and actual classifications (Bhandari, 2024).

The image below shows the confusion matrix for the Random Forest Classifier on the binary classification task of predicting the 'CreditLevel' target variable, where the rows represent the actual classes (True Labels), and the columns represent the predicted classes (Predicted Labels).

In this case, the Random Forest Classifier performed well, correctly classifying 1347 out of 1356 instances of class 0 (True Negatives) and 1089 out of 1097 instances of class 1 (True Positives). However, it misclassified 9 instances of class 0 as class 1 (False Positives), and 8 instances of class 1 as class 0 (False Negatives).

Figure 14: Confusion Matrix



Actual \ Predicted	0	1
0	TN	FP
1	FN	TP

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

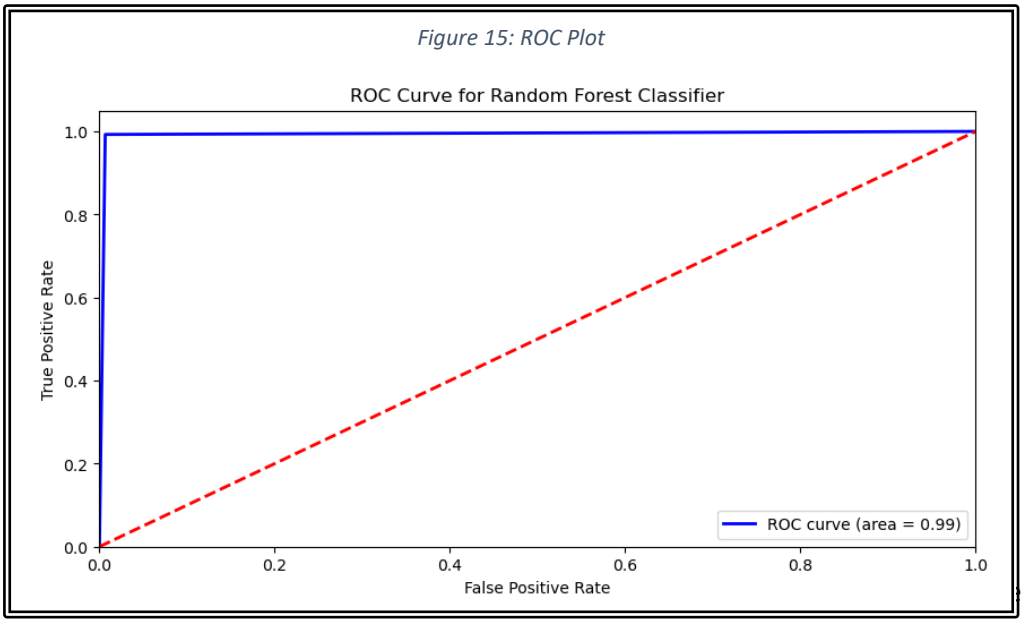
Source: (Selvaraj, 2022)

8.3 ROC Plot

The Receiver Operating Characteristic (ROC) curve is a performance evaluation metric that plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various classification thresholds. It provides a comprehensive view of the trade-off between the model's ability to correctly classify positive instances and negative instances (Fawcett, 2005).

The area under the ROC curve (AUC) is a summary metric that represents the overall performance of the classifier. An AUC of 1.0 indicates a perfect classifier, while an AUC of 0.5 represents a random classifier.

Figure 15: ROC Plot



The ROC curve for the Random Forest Classifier is very close to the top-left corner, indicating excellent performance. The AUC value is reported as 0.99, which is very close to 1.0 and suggests that the classifier has a high true positive rate while maintaining a low false positive rate across different classification thresholds.

9. Is Machine Learning Capable of Accurate Prediction?

Our analysis of the 'EM' dataset highlights the strengths and limitations of various machine learning algorithms for credit scoring. Logistic regression, while simple and interpretable, assumes linear relationships and pre-determined distributions, constraining its flexibility (Dastile et al, 2020). AdaBoost builds a strong classifier by iteratively training weak learners, but can be sensitive to noisy data, potentially leading to overfitting. Conversely, Random Forest's ensemble of decision trees offers robustness against noisy data and overfitting, making it our choice for predicting 'CreditLevel'.

Based on our analysis, **machine learning can predict the credit status of the firms accurately with Random Forest accuracy score of 99.31%**. However, the very high accuracy rate can be attributed to the overfitting of the training data on the test data.

Machine learning's advancements in credit scoring offer greater flexibility and effectiveness than traditional statistical methods (Bacham and Zhao, 2017), but faces limitations such as lack of explainability and interpretability. According to Dumitrescu et al (2021), most machine learning algorithms are considered as “black box” due to the scoring rules and credit approval being complicated to explain to customers and regulators. Furthermore, Amato et al (2022) discuss the quality of input data, inaccuracies, incompleteness, or biases in the data can lead to incorrect assessment. Another problem they mention is overfitting and unintentional bias.

Dastile et al (2020) discusses future research in credit scoring set to enhance model accuracy and efficiency through several innovative approaches. These areas include balancing dataset classes using techniques like linear dependence and wavelet transformations. Incorporating macro-economic variables such as interest rates, unemployment rate, and inflation rate are also recommended to enhance the assessment of default risks. In our opinion, possible eventualities such as financial crisis, epidemic and natural disasters also need to be factored in.

Furthermore, understanding the time to default could enable lenders to adopt a more proactive approach in risk management. The author advocate for a deeper focus to research on EDAs to better understand the distribution within a credit dataset as it would be fundamental for selecting the most appropriate modelling techniques and for accurately capturing complexities. Further, incorporating heterogeneous base classifiers in ensemble methods and leveraging Explainable Artificial Intelligence (XAI) can enhance model interpretability, address overfitting, and balance accuracy with transparency. These strategies aim to refine credit scoring models, making them more dynamic and reflective of real-world financial risks, thereby advancing the reliability and applicability of credit assessment in evolving economic landscapes (Moscato and Sperli, 2023).

Addressing these limitations, we believe, will refine credit scoring models to be more dynamic and reflective of real-world financial risks.

10. CONCLUSION

This report explores firm creditworthiness using EDA and machine learning on UK firms' 2019-2020 data. EDA focused on credit scores and numerical attributes, laying the groundwork for modeling. AdaBoost and Random Forest classifiers excelled in predicting 'CreditLevel', demonstrating high accuracy, precision, recall, and F1-scores. Random Forest achieved 99.30% accuracy, slightly outperforming AdaBoost. While showcasing effective prediction, it's crucial to acknowledge potential limitations such as dataset dependency, overfitting, and real-world applicability. High accuracy alone doesn't guarantee real-world performance; factors like dataset quality and overfitting must be considered. Careful interpretation and ongoing scrutiny are essential in leveraging machine learning effectively for credit assessment.

11. REFERENCE

almabetter, n.d. *AdaBoost Algorithm in Machine Learning.* [Online]
Available at: https://www.almabetter.com/bytes/tutorials/data-science/adaboost-algorithm
[Accessed 16 April 2024].

Amato, F. Ferraro, A. Galli, A. Moscato, V. and Sperli (2022) Credit Score Prediction Relying on Machine Learning. Available on https://ceur-ws.org/Vol-3194/paper64.pdf [Accessed on 17 April 2024]

AWS, n.d. *Cross-Validation.* [Online]
Available at: https://docs.aws.amazon.com/machine-learning/latest/dg/cross-validation.html
[Accessed 17 April 2024].

Bacham, D. and Zhao, J. (2017) Machine Learning: Challenges, Lessons, and Opportunities in Credit Risk Modelling. Available on https://www.moodysanalytics.com/risk-perspectives-magazine/managing-disruption/spotlight/machine-learning-challenges-lessons-and-opportunities-in-credit-risk-modeling [Accessed on 17 April 2024]

Bhandari, A., 2024. *Understanding & Interpreting Confusion Matrix in Machine Learning (Updated 2024).* [Online]
Available at: https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/
[Accessed 16 April 2024].

Christiansen, S. D., 2021. *Schematic Diagram of the Random Forest Classifier.* [Online]
Available at: https://www.researchgate.net/figure/11-Schematic-diagram-of-the-random-forest-classifier_fig9_353045870
[Accessed 16 April 2024].

Dastile, X., Celik, T. and Potsane, M. (2020) Statistical and machin learning models in credit scoring: A systematic literature survey. *Applied Soft Computing Journal.* Vol 91 pp. 106263.

Dumitrescu, E. Hue, S., Hurlin, C., Tokpavi, S. (2021) Machine learning for credit scoring: Improving logistic regression with non-linear decision-tree effects. *European Journal of Operational Research*. No. 297 pp. 1178 -1192.

Fawcett, T., 2005. An introduction to ROC analysis. *Pattern Recognition Letters,* 27(8).

Filho, M., 2023. *How To Get Feature Importance in Random Forests.* [Online]
Available at: https://forecastegy.com/posts/feature-importance-in-random-forests/
[Accessed 16 April 2024].

Kanade, V., 2022. *What Is a Support Vector Machine? Working, Types, and Examples.* [Online]
Available at: https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/
[Accessed 16 April 2024].

Kanade, V., 2022. *What Is Logistic Regression? Equation, Assumptions, Types, and Best Practices.* [Online]
Available at: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/
[Accessed 16 April 2024].

Lejafar, 2018. *Feature Value Importance - AdaBoost Classifier.* [Online] Available at: https://stats.stackexchange.com/questions/324383/feature-value-importance-adaboost-classifier#:~:text=AdaBoost's%20feature%20importance%20is%20derived,provided%20by%20each%20Decision%20Tree.
[Accessed 16 April 2024].

Moscato, V, Sperli, G. () A benchmark of credit score prediction using Machin Learning. Available on https://ceur-ws.org/Vol-3486/86.pdf [Accessed on 17 April 2024].

Mukhiya, S. K., & Ahmed, U. (2020). Hands-on exploratory data analysis with python : perform EDA techniques to understand, summarize, and investigate your data. (1st edition). Packt. Publishing, Birmingham-Mumbai

Schneider, P. & Xhafa, F., 2022. Mean Absolute Error (MAE) - Chapter 3 - Anomaly Detection: Concepts and Methods. In: *Anomaly Detection and Complex Event Processing over IoT Data Streams.* s.l.:Academic Press, pp. 49-66.

scikit-learn.org, n.d. *sklearn.ensemble.AdaBoostClassifier.* [Online]
Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#:~:text=An%20AdaBoost%20%5B1%5D%20classifier%20is,focus%20more%20on%20difficult%20cases.
[Accessed 16 April 2024].

scikit-learn.org, n.d. *sklearn.ensemble.RandomForestClassifier.* [Online]
Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[Accessed 16 April 2024].

Selvaraj, N., 2022. *Confusion Matrix, Precision, and Recall Explained.* [Online]
Available at: https://www.kdnuggets.com/2022/11/confusion-matrix-precision-recall-explained.html
[Accessed 01 April 2024].

Shin, T., 2023. *Understanding Feature Importance in Machine Learning.* [Online]
Available at: https://builtin.com/data-science/feature-importance
[Accessed 17 April 2024].

Yao, Y., 2024. *Lecture Notes - Artificial Intelligence and Machine Learning.* s.l.:University of Westminster.

Appendix 1. Coding

The jupyter notebook file for all relevant coding can be found on the below link:

https://github.com/vershasandesh/AI-ML/blob/main/AI%20and%20ML%20-%20Group%20CW%20-%20Final.ipynb

Note: The screenshots of codes are provided below in same sequence as the report headings.

Refer Section 1: Loading EM dataset for machine learning process:

Credit status and characteristics of some firms in the UK from 2019 and 2020

Load necessary libraries

```
In [45]: import warnings
         warnings.simplefilter(action='ignore', category=FutureWarning)
         warnings.filterwarnings('ignore')
```

```
In [46]: import numpy as np
         import seaborn as sns
         import matplotlib.pyplot as plt
         %matplotlib inline
```

Loading dataset

```
In [47]: import pandas as pd
         data_all= pd.read_csv("EM.csv")
```

```
In [48]: data_all
```

Out[48]:

	Creditscore	Creditscoreindicator	Likelihoodoffailure	CreditlimitGBPGBP	Previouscreditscore	SMEindicator	ReturnonTotalAssets2020	ReturnonTotalAssets
0	92	Secure	0.9	50000000.0	95.0	No	NaN	5.61
1	92	Secure	0.9	50000000.0	99.0	No	-0.130484	2.83
2	95	Secure	0.9	16574000.0	99.0	No	NaN	3.81
3	89	Secure	0.9	5380000.0	92.0	No	NaN	-5.70
4	99	Secure	0.9	50000000.0	99.0	No	21.144665	26.91
...	...	...	...	...	...	...	...	
8171	31	Caution	10.8	846.0	38.0	Yes	NaN	51.34
8172	19	High Risk	10.6	NaN	21.0	Unconfirmed	NaN	-73.02
8173	21	Caution	10.6	NaN	29.0	Yes	NaN	94.23
8174	21	Caution	10.6	NaN	21.0	Yes	NaN	-100.00
8175	64	Stable	2.2	1017000.0	52.0	Yes	17.365920	1.82

8176 rows × 72 columns

Refer Section 2: Exploratory Data Analysis

Section 2.1: Summarizing the data

Exploratory Data Analysis (EDA)

Showing the summary statistics

In [50]:
```
data_all.describe()
```

Out[50]:

	Creditscore	Likelihoodoffailure	CreditlimitGBPGBP	Previouscreditscore	ReturnonTotalAssets2020	ReturnonTotalAssets2019	ReturnonShareholdersFunds2
count	8176.000000	8107.000000	6.771000e+03	7740.000000	2981.000000	6773.000000	2443.00(
mean	57.300514	5.222030	5.292997e+05	60.431912	27.903380	22.988502	65.89(
std	29.730987	4.402007	2.983264e+06	29.744458	122.359699	105.484711	160.95(
min	15.000000	0.900000	5.000000e+02	12.000000	-986.769231	-912.765957	-986.76(
25%	29.000000	0.900000	5.000000e+02	31.000000	0.258019	0.058547	5.02
50%	45.000000	4.500000	1.543000e+04	57.000000	8.895044	6.909952	25.34(
75%	90.000000	10.600000	2.342970e+05	92.000000	44.012601	24.904707	95.71(
max	99.000000	19.400000	5.000000e+07	99.000000	815.362798	977.934426	991.53(

8 rows × 70 columns

In [51]:
```
data_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8176 entries, 0 to 8175
Data columns (total 72 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Creditscore                     8176 non-null   int64
 1   Creditscoreindicator            8176 non-null   object
 2   Likelihoodoffailure             8107 non-null   float64
 3   CreditlimitGBPGBP               6771 non-null   float64
 4   Previouscreditscore             7740 non-null   float64
 5   SMEindicator                    8176 non-null   object
 6   ReturnonTotalAssets2020         2981 non-null   float64
 7   ReturnonTotalAssets2019         6773 non-null   float64
 8   ReturnonShareholdersFunds2020   2443 non-null   float64
 9   ReturnonShareholdersFunds2019   5456 non-null   float64
 10  ReturnonCapitalEmployed2020     2913 non-null   float64
 11  ReturnonCapitalEmployed2019     6601 non-null   float64
 12  TurnoverthGBP2020               3172 non-null   float64
 13  TurnoverthGBP2019               7163 non-null   float64
 14  NetAssetsTurnoverx2020          2802 non-null   float64
 15  NetAssetsTurnoverx2019          6260 non-null   float64
 16  TradeCreditorsthGBP2020         1472 non-null   float64
 17  TradeCreditorsthGBP2019         4338 non-null   float64
 18  StockTurnoverx2020              929 non-null    float64
 19  StockTurnoverx2019              3014 non-null   float64
 20  StockWIPthGBP2020               1074 non-null   float64
 21  StockWIPthGBP2019               3209 non-null   float64
 22  NetCurrentAssets2020            3743 non-null   float64
 23  NetCurrentAssets2019            7680 non-null   float64
```

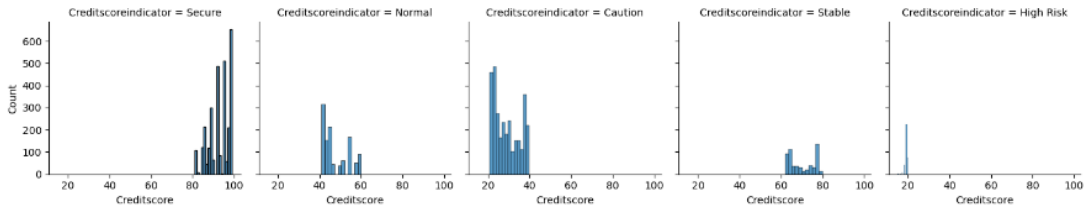
Section 2.2: Analysis of Visual Outputs

EDA Analysis of the two features

Histogram of 'Credit Score' grouped by 'Creditscoreindicator'

In [52]:
```
chart = sns.FacetGrid(data_all, col='Creditscoreindicator')
chart.map(sns.histplot, 'Creditscore')
```

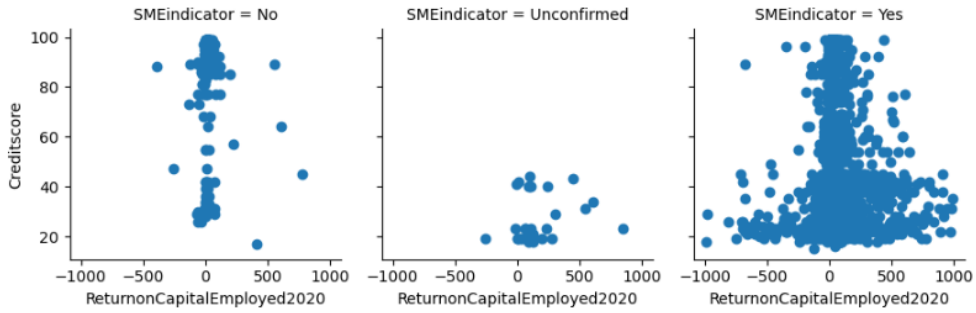
Out[52]:
```
<seaborn.axisgrid.FacetGrid at 0x1a84bd34750>
```



Scatter plot between 'Creditscore' and 'ReturnonCapitalEmployed2020' grouped by 'SMEindicator'

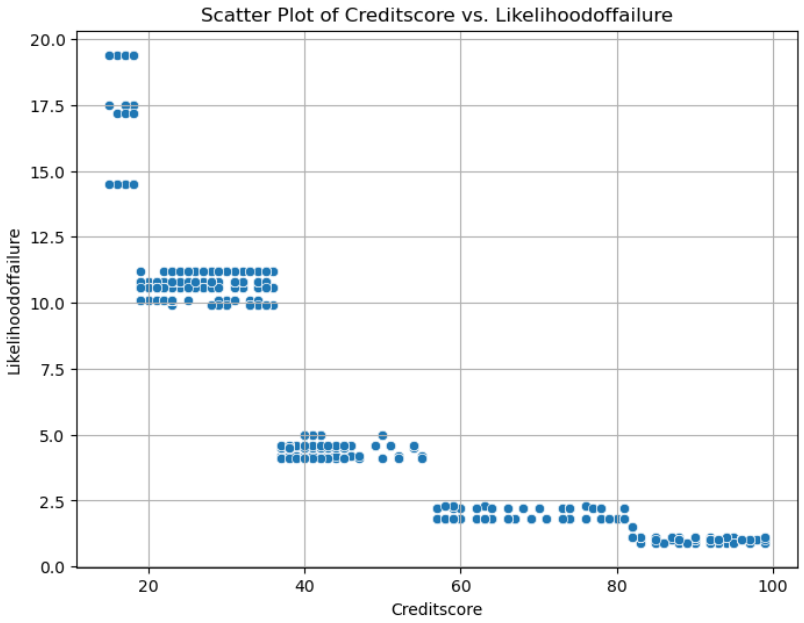
In [53]:
```
chart = sns.FacetGrid(data_all, col='SMEindicator')
chart.map(plt.scatter, 'ReturnonCapitalEmployed2020', 'Creditscore')
```

Out[53]: <seaborn.axisgrid.FacetGrid at 0x1a84c1c2f10>

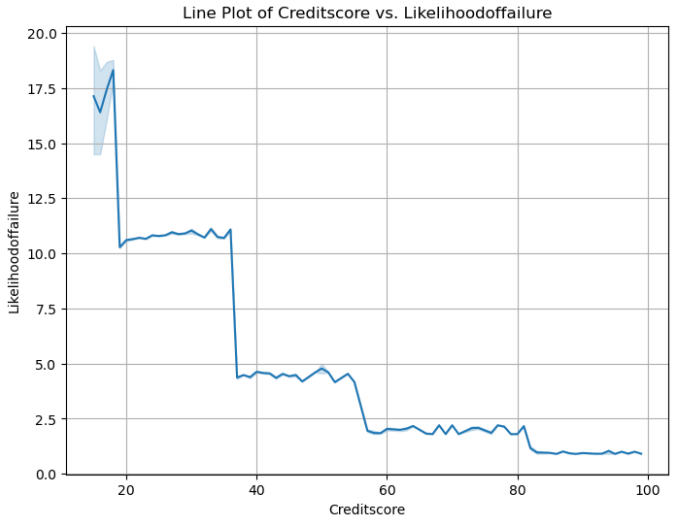


Scatter plot and Line plot between 'Creditscore' and 'Likelihoodoffailure'

In [54]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=data_all, x='Creditscore', y='Likelihoodoffailure')
plt.title('Scatter Plot of Creditscore vs. Likelihoodoffailure')
plt.xlabel('Creditscore')
plt.ylabel('Likelihoodoffailure')
plt.grid(True)
plt.show()
```

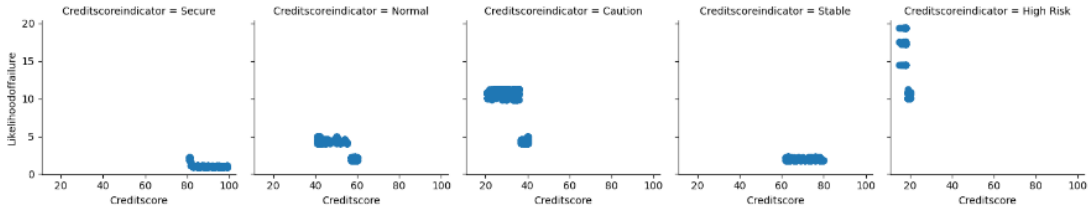


In [55]:
```
plt.figure(figsize=(8, 6))
sns.lineplot(data=data_all, x='Creditscore', y='Likelihoodoffailure')
plt.title('Line Plot of Creditscore vs. Likelihoodoffailure')
plt.xlabel('Creditscore')
plt.ylabel('Likelihoodoffailure')
plt.grid(True)
plt.show()
```



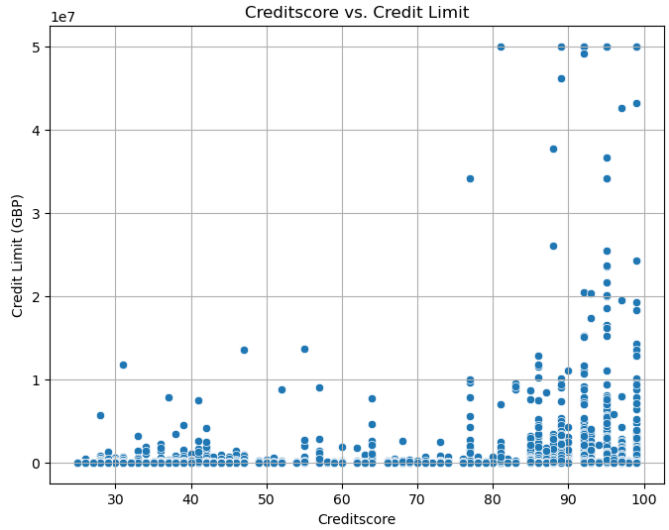
Scatter plot between 'Creditscore' and 'Likelihoodoffailure' grouped by 'Creditscoreindicator'

In [56]:
```
chart = sns.FacetGrid(data_all, col='Creditscoreindicator')
chart.map(plt.scatter, 'Creditscore', 'Likelihoodoffailure')
plt.show()
```



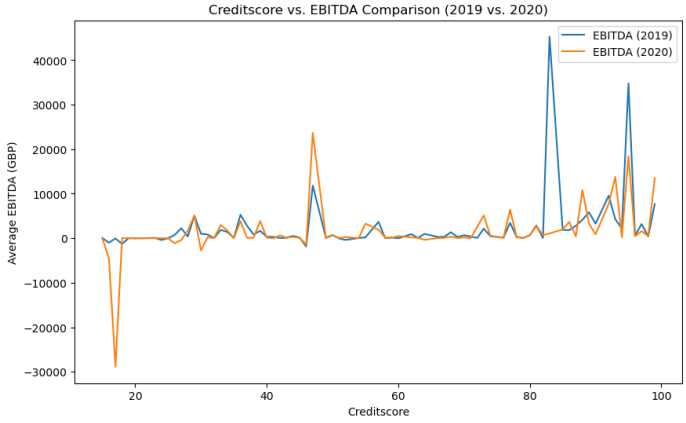
Scatter plot between 'Creditscore' and 'CreditlimitGBP'

In [57]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=data_all, x='Creditscore', y='CreditlimitGBPGBP')
plt.title('Creditscore vs. Credit Limit')
plt.xlabel('Creditscore')
plt.ylabel('Credit Limit (GBP)')
plt.grid(True)
plt.show()
```

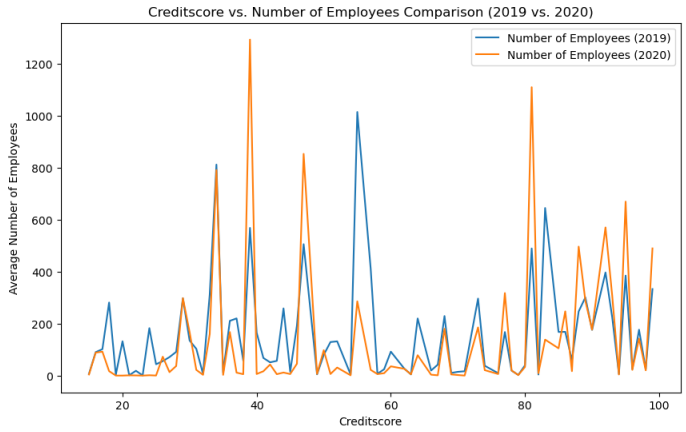


'Creditscore' with 'EBITDA' and 'Numberofemployees'

In [58]:
```
# Calculate the average EBITDA for each Creditscore for 2019
avg_ebitda_2019 = data_all.groupby('Creditscore')['EBITDAthGBP2019'].mean().reset_index()

# Calculate the average EBITDA for each Creditscore for 2020
avg_ebitda_2020 = data_all.groupby('Creditscore')['EBITDAthGBP2020'].mean().reset_index()

# Plot Creditscore vs. EBITDA for 2019
plt.figure(figsize=(10, 6))
sns.lineplot(data=avg_ebitda_2019, x='Creditscore', y='EBITDAthGBP2019', label='EBITDA (2019)')

# Plot Creditscore vs. EBITDA for 2020
sns.lineplot(data=avg_ebitda_2020, x='Creditscore', y='EBITDAthGBP2020', label='EBITDA (2020)')

# Add labels and title
plt.title('Creditscore vs. EBITDA Comparison (2019 vs. 2020)')
plt.xlabel('Creditscore')
plt.ylabel('Average EBITDA (GBP)')
plt.legend()
plt.show()
```



In [59]:
```
# Calculate the average Numberofemployees for each Creditscore for 2019
avg_employees_2019 = data_all.groupby('Creditscore')['Numberofemployees2019'].mean().reset_index()

# Calculate the average Numberofemployees for each Creditscore for 2020
avg_employees_2020 = data_all.groupby('Creditscore')['Numberofemployees2020'].mean().reset_index()

# Plot Creditscore vs. Numberofemployees for 2019
plt.figure(figsize=(10, 6))
sns.lineplot(data=avg_employees_2019, x='Creditscore', y='Numberofemployees2019', label='Number of Employees (2019)')

# Plot Creditscore vs. Numberofemployees for 2020
sns.lineplot(data=avg_employees_2020, x='Creditscore', y='Numberofemployees2020', label='Number of Employees (2020)')

# Add labels and title
plt.title('Creditscore vs. Number of Employees Comparison (2019 vs. 2020)')
plt.xlabel('Creditscore')
plt.ylabel('Average Number of Employees')
plt.legend()
plt.show()
```



Section 2.3: Creating New Feature

Creating new feature: 'CreditLevel'

In [62]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd

# Create 'CreditLevel' feature
mean_creditscore = data_all['Creditscore'].mean()
data_all['CreditLevel'] = (data_all['Creditscore'] > mean_creditscore).astype(int)
```

Dropping 'Creditscore' feature, as a binary variable 'CreditLevel' has been created for it

In [63]:
```
# Drop 'CreditScore' feature
data_all.drop(columns=['Creditscore'], inplace=True)
```

Setting the 'CreditLevel' as the target variable 'Y' and the remaining numerical data are 'X'

In [64]:
```
# Separate features (X) and target variable (Y)
X = data_all.drop(['CreditLevel'], axis=1)  # Drop 'CreditLevel' from features
Y = data_all['CreditLevel']                 # Assigning 'CreditLevel' as the target variable Y
```

In [65]:
```
X
```

Out[65]:

rsFunds2019	ReturnonCapitalEmployed2020	...	ShareholdersFundsthGBP2020	ShareholdersFundsthGBP2019	TaxationthGBP2020	TaxationthGBP2019	Grossmargir
10.284360	NaN	...	NaN	2110000.000	NaN	-50000.0	
7.000426	-0.175859	...	2713800.000	2582700.000	-14200.0	-58000.0	29.7
18.074145	NaN	...	NaN	397247.000	NaN	-19589.0	
-67.554766	NaN	...	NaN	168900.000	NaN	-3300.0	
130.534489	28.898498	...	441500.000	553800.000	-138300.0	-132500.0	38.4
...	...	...	...	...	...	...	
92.855191	NaN	...	31.868	36.600	NaN	NaN	
NaN	NaN	...	NaN	-0.823	NaN	NaN	
NaN	NaN	...	92.061	NaN	NaN	NaN	
NaN	NaN	...	-13.702	NaN	NaN	NaN	
100.000000	95.171137	...	3921.047	189.342	NaN	NaN	

In [66]:
```
Y
```

Out[66]:
```
0       1
1       1
2       1
3       1
4       1
       ..
8171    0
8172    0
8173    0
8174    0
8175    1
Name: CreditLevel, Length: 8176, dtype: int32
```

Section 2.4: Data Preprocessing

Data preprocessing

Dropping non-numeric columns

In [67]:
```
# Identify non-numeric columns
non_numeric_cols = X.select_dtypes(exclude=['float64', 'int64']).columns

# Print non-numeric columns
print("Non-numeric columns:")
print(non_numeric_cols)
```

```
Non-numeric columns:
Index(['Creditscoreindicator', 'SMEindicator'], dtype='object')
```

In [68]:
```
X = X.drop(columns=['Creditscoreindicator', 'SMEindicator'])
```

In [69]:
```
X
```

Out[69]:

	Likelihoodoffailure	CreditlimitGBPGBP	Previouscreditscore	ReturnonTotalAssets2020	ReturnonTotalAssets2019	ReturnonShareholdersFunds2020	Returnon
0	0.9	50000000.0	95.0	NaN	5.614489	NaN	
1	0.9	50000000.0	99.0	-0.130484	2.832746	-0.283735	
2	0.9	16574000.0	99.0	NaN	3.817802	NaN	
3	0.9	5380000.0	92.0	NaN	-5.702719	NaN	
4	0.9	50000000.0	99.0	21.144665	26.910621	169.535674	
...	...	...	...	...	...	...	
8171	10.8	846.0	38.0	NaN	51.344614	NaN	
8172	10.6	NaN	21.0	NaN	-73.025732	NaN	
8173	10.6	NaN	29.0	NaN	94.231524	NaN	
8174	10.6	NaN	21.0	NaN	-100.000000	NaN	
8175	2.2	1017000.0	52.0	17.365920	1.825321	95.171137	

8176 rows × 69 columns

Filling missing values with feature means

In [72]:
```
# Fill missing values with feature means
X = X.fillna(X.mean())
```

using 30% data as test set, and random state as '123'

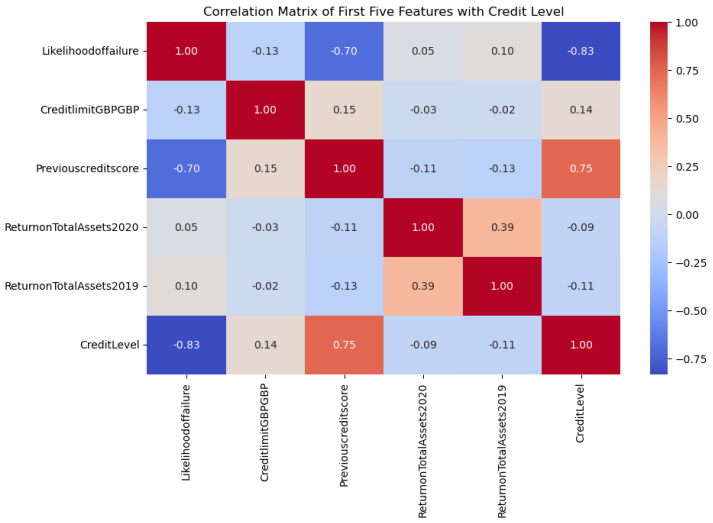
In [73]:
```
# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=123)
```

Using StandardScaler to X

In [74]:
```
# Fit the StandardScaler on the training data
scaler = StandardScaler()
scaler.fit(X_train)

# Transform the training and testing sets using the fitted StandardScaler
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Section 2.5: Correlation Matrix

In [71]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Concatenate X and Y for correlation analysis
df_corr = pd.concat([X.iloc[:, :5], Y], axis=1)

# Generate correlation matrix
corr_matrix = df_corr.corr()

# Plot correlation matrix as a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", annot_kws={"size": 10})
plt.title('Correlation Matrix of First Five Features with Credit Level')
plt.show()
```



Refer Section 3: Machine Learning Classification Methods

1. AdaBoostClassifier

In [75]:
```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score, classification_report


# Initialize AdaBoost classifier
adaboost_clf = AdaBoostClassifier(n_estimators=50, random_state=123)  # Adjust n_estimators as needed

# Train AdaBoost classifier
adaboost_clf.fit(X_train_scaled, Y_train)

# Predict using AdaBoost classifier
adaboost_preds = adaboost_clf.predict(X_test_scaled)

# Evaluate AdaBoost classifier
print("Accuracy of AdaBoost classifier:", accuracy_score(Y_test, adaboost_preds))
print("Classification Report of AdaBoost classifier:")
print(classification_report(Y_test, adaboost_preds))
```

```
Accuracy of AdaBoost classifier: 0.9922543823889115
Classification Report of AdaBoost classifier:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1356
           1       0.99      0.99      0.99      1097

    accuracy                           0.99      2453
   macro avg       0.99      0.99      0.99      2453
weighted avg       0.99      0.99      0.99      2453
```

2. RandomForest Classifier

In [77]:
```
from sklearn.ensemble import RandomForestClassifier

# Initialize Random Forest classifier
random_forest_clf = RandomForestClassifier(n_estimators=50, random_state=123)  # Adjust n_estimators as needed

# Train Random Forest classifier
random_forest_clf.fit(X_train_scaled, Y_train)

# Predict using Random Forest classifier
random_forest_preds = random_forest_clf.predict(X_test_scaled)

# Evaluate Random Forest classifier
print("Accuracy of Random Forest classifier:", accuracy_score(Y_test, random_forest_preds))
print("Classification Report of Random Forest classifier:")
print(classification_report(Y_test, random_forest_preds))
```

```
Accuracy of Random Forest classifier: 0.9930697105584998
Classification Report of Random Forest classifier:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1356
           1       0.99      0.99      0.99      1097

    accuracy                           0.99      2453
   macro avg       0.99      0.99      0.99      2453
weighted avg       0.99      0.99      0.99      2453
```

Support Vector Machine (SVM) Classification

In [79]:
```
from sklearn.svm import SVC

# Initialize SVM classifier
svm_clf = SVC(kernel='linear', random_state=123)

# Train SVM classifier
svm_clf.fit(X_train_scaled, Y_train)

# Predict using SVM classifier
svm_preds = svm_clf.predict(X_test_scaled)

# Evaluate SVM classifier
print("Accuracy of SVM classifier:", accuracy_score(Y_test, svm_preds))
print("Classification Report of SVM classifier:")
print(classification_report(Y_test, svm_preds))
```

```
Accuracy of SVM classifier: 0.9889930697105584
Classification Report of SVM classifier:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1356
           1       0.99      0.99      0.99      1097

    accuracy                           0.99      2453
   macro avg       0.99      0.99      0.99      2453
weighted avg       0.99      0.99      0.99      2453
```

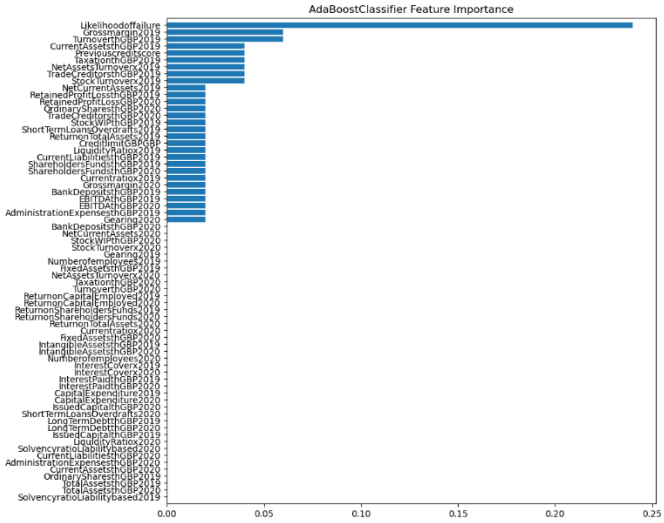
4. Logistic Regression

In [80]:
```
from sklearn.linear_model import LogisticRegression

# Initialize Logistic Regression classifier
logreg_clf = LogisticRegression(random_state=123)

# Train Logistic Regression classifier
logreg_clf.fit(X_train_scaled, Y_train)

# Predict using Logistic Regression classifier
logreg_preds = logreg_clf.predict(X_test_scaled)

# Evaluate Logistic Regression classifier
print("Accuracy of Logistic Regression classifier:", accuracy_score(Y_test, logreg_preds))
print("Classification Report of Logistic Regression classifier:")
print(classification_report(Y_test, logreg_preds))
```

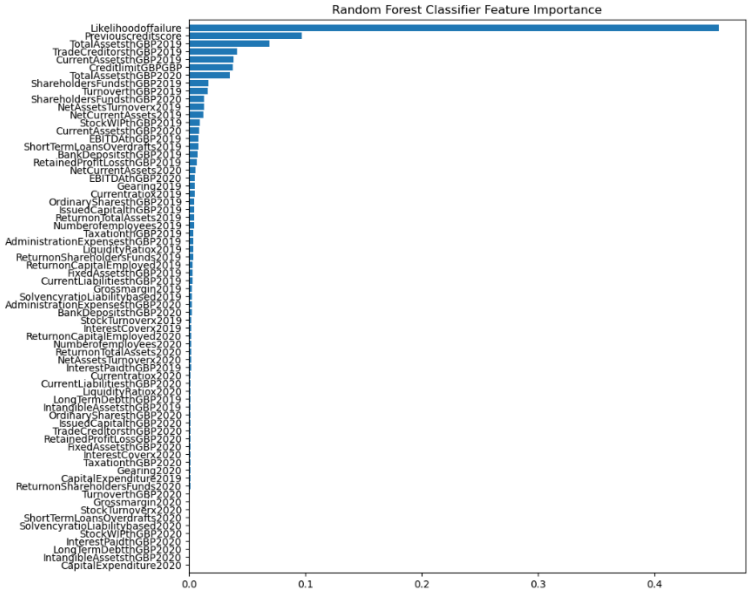
```
Accuracy of Logistic Regression classifier: 0.9894007337953526
Classification Report of Logistic Regression classifier:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1356
           1       0.99      0.99      0.99      1097

    accuracy                           0.99      2453
   macro avg       0.99      0.99      0.99      2453
weighted avg       0.99      0.99      0.99      2453
```

Refer Section 4: Feature Importance

In [76]:
```
# Extract feature importance
feature_names = X_train.columns
importance = adaboost_clf.feature_importances_
indices = np.argsort(importance)
range1 = range(len(importance))

# Plot feature importance
plt.figure(figsize=(10, 10))
plt.title("AdaBoostClassifier Feature Importance")
plt.barh(range1, importance[indices])
plt.yticks(range1, feature_names[indices])
plt.ylim([-1, len(range1)])
plt.show()
```



In [78]:
```
# Extract feature importance
importance_rf = random_forest_clf.feature_importances_
indices_rf = np.argsort(importance_rf)

# Plot feature importance
plt.figure(figsize=(10, 10))
plt.title("Random Forest Classifier Feature Importance")
plt.barh(range1, importance_rf[indices_rf])
plt.yticks(range1, feature_names[indices_rf])
plt.ylim([-1, len(range1)])
plt.show()
```



Refer Section 5: Cross-Validation

Cross-Validation of AdaBoost and RandomForest Classifiers

In [81]:
```
from sklearn.model_selection import cross_val_score

# Perform cross-validation for AdaBoost classifier
adaboost_scores = cross_val_score(adaboost_clf, X_train_scaled, Y_train, cv=5)
adaboost_mean_accuracy = adaboost_scores.mean()
adaboost_std_accuracy = adaboost_scores.std()

print("Cross-validation scores for AdaBoost classifier:", adaboost_scores)
print("Mean accuracy of AdaBoost classifier:", adaboost_mean_accuracy)
print("Standard deviation of accuracy of AdaBoost classifier:", adaboost_std_accuracy)
```

```
Cross-validation scores for AdaBoost classifier: [0.98951965 0.99126638 0.9930131  0.98601399 0.99300699]
Mean accuracy of AdaBoost classifier: 0.9905640211317067
Standard deviation of accuracy of AdaBoost classifier: 0.0026173624897451817
```

In [82]:
```
# Perform cross-validation for Random Forest classifier
random_forest_scores = cross_val_score(random_forest_clf, X_train_scaled, Y_train, cv=5)
random_forest_mean_accuracy = random_forest_scores.mean()
random_forest_std_accuracy = random_forest_scores.std()

print("Cross-validation scores for Random Forest classifier:", random_forest_scores)
print("Mean accuracy of Random Forest classifier:", random_forest_mean_accuracy)
print("Standard deviation of accuracy of Random Forest classifier:", random_forest_std_accuracy)
```

```
Cross-validation scores for Random Forest classifier: [0.99126638 0.99475983 0.9930131  0.98863636 0.99388112]
Mean accuracy of Random Forest classifier: 0.9923113567655053
Standard deviation of accuracy of Random Forest classifier: 0.002170214543654544
```

Refer Section 6: Mean Absolute Error

Mean Absolute Error

In [83]:
```
from sklearn.metrics import mean_absolute_error

# Calculate mean absolute error for AdaBoost classifier
adaboost_mae = mean_absolute_error(Y_test, adaboost_preds)
print("Mean Absolute Error of AdaBoost classifier:", adaboost_mae)

# Calculate mean absolute error for Random Forest classifier
random_forest_mae = mean_absolute_error(Y_test, random_forest_preds)
print("Mean Absolute Error of Random Forest classifier:", random_forest_mae)
```

```
Mean Absolute Error of AdaBoost classifier: 0.0077456176110884635
Mean Absolute Error of Random Forest classifier: 0.006930289441500204
```

Refer Section 8: Predicting the target variable

Predicting the target variable using RandomForestClassifier Method

In [84]:
```
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc

# Predict using Random Forest classifier
random_forest_preds = random_forest_clf.predict(X_test_scaled)

# Evaluate Random Forest classifier
print("Accuracy of Random Forest classifier:", accuracy_score(Y_test, random_forest_preds))
print("Classification Report of Random Forest classifier:")
print(classification_report(Y_test, random_forest_preds))
```

```
Accuracy of Random Forest classifier: 0.9930697105584998
Classification Report of Random Forest classifier:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1356
           1       0.99      0.99      0.99      1097

    accuracy                           0.99      2453
   macro avg       0.99      0.99      0.99      2453
weighted avg       0.99      0.99      0.99      2453
```

Confusion Matrix

In [85]:
```
# Confusion Matrix
conf_matrix_rf = confusion_matrix(Y_test, random_forest_preds)

# Plot Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_rf, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title('Confusion Matrix - Random Forest Classifier')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

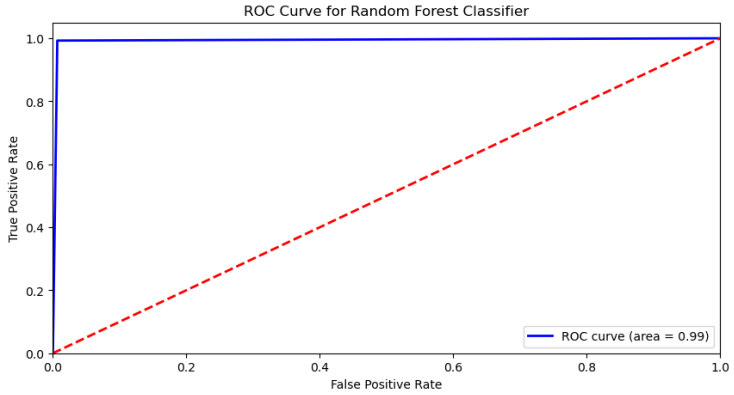
		Predicted 0	Predicted 1
Actual	0	TN	FP
	1	FN	TP

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

ROC Plot

In [86]:
```
# Plot ROC curve
fpr_rf, tpr_rf, thresholds_rf = roc_curve(Y_test, random_forest_preds)
roc_auc_rf = auc(fpr_rf, tpr_rf)

plt.figure(figsize=(10, 5))
plt.plot(fpr_rf, tpr_rf, color='blue', lw=2, label='ROC curve (area = %0.2f)' % roc_auc_rf)
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Random Forest Classifier')
plt.legend(loc="lower right")
```



APPENDIX 2. PEER REVIEW

Westminster Business School

Group Assessment
Team Evaluation Form
Machine Learning and Artificial Intelligence Group Report

This form is intended to assess personal contribution as well as group members' contribution towards the preparation and submission of the Machine Learning and Artificial Intelligence Module [7FNCE043W.2] group assignment.

In our initial meeting, we established a frame of expectations essential for the successful completion of the assignment. This framework included clear and effective communication, timely completion of tasks, collaborative problem-solving, and the ability to quickly incorporate feedback into the ongoing tasks assigned to each group member. The form includes a self-assessment section as well as an assessment of each team member, ensuring a thorough evaluation of contributions based on these established criteria.

Self-Assessment:
Name: Versha Sandesh
Student ID: w2039582

For this group project, I worked on the following sections:

1. Whole python coding on the given dataset,
2. Did wider research on the topics,
3. Description of AdaBoost Classification Method and all the relevant test,
4. Wrote short description for SVM and Logistic Regression methods,
5. Evaluated Machine Learning Classification methods,
6. Combined all the individual work from other group members and compiled the report in its final format.

Peer Review:

Group Member 1: Wubedel Asfaw Shuba

Student ID: w1957950

Wubedel contributed at her best for this group work, her specific contributions are as below:

1. Described whole EDA Analysis, including summary statistics, visualisations, data pre-processing and feature creation,
2. She did the wider research on the topic,
3. She wrote introduction, literature and analysis write up on machine learning, limitations and the Abstract,
4. She created template for this report, and for the peer review form,
5. She accompanied me in the final report compilation.

Group Member 2: Surbhiben Rameshbhai Rabadiya

Student ID: w2026095

Surbhiben contributed satisfactorily to the group project and completed her tasks on time.

1. She described the Random Forest classification method and its relevant tests,
2. She wrote the description about predicting the target variable,
3. She also described the Confusion matrix and ROC plot,
4. She did research relevant to these topics.

Group Member 3: Harshit Suleghai Ravi

Student ID: w1951638

Harshit contributed to group project and completed all his tasks satisfactorily, as below:

1. He helped me in python coding for the group work,
2. He did some part of Introduction,
3. He wrote the description for correlation matrix and conclusion,
4. He did research relevant to these topics and provided feedback on codes based on his prior coding experience.

Overall, every group member contributed to the best of our knowledge and effort to compose this report. This assignment has been both challenging and rewarding, providing us with valuable lessons in time management, organization, and the value of constructive feedback to improve each other's input. These experiences have enhanced our practical skills as well as strengthened our ability to effectively work in a group setting on a subject, we are fairly new in. In addition, it has taught us the importance of each member's contributions and the necessity of clear communication and mutual support in achieving our collective goals.