MACHINE LEARNING FOR FINANCIAL RISK ASSESSMENT

(Group Coursework)

MSc Fintech with Business Analytics
Module: Artificial Intelligence and Machine Learning (FNCE043W)
Module Leader: Dr. Yumei Yao
Word Count: 3256(Excluding cover page, Index, References, and appendix)

Submitted by: Group-5
Virati Nilesh Mehta (Submitting peer review)
Federica Migliaro (Submitting peer review)
Hajarat Oloko (Submitting peer review)
Sarika Patel (Submitting Report)

TABLE OF CONTENTS

CONTENTS

1.INTRODUCTION

In today's volatile economic landscape, characterised by rapid technological advancements and unpredictable global events, the financial stability of firms is continually at risk. Accurately predicting the creditworthiness of businesses has thus become a cornerstone of risk management and a critical factor in strategic planning. Credit scores, pivotal quantitative measures of credit risk, significantly influence lending decisions, investment opportunities, and broader economic perceptions of a firm's health. Given the complexities of financial environments, traditional methods of credit assessment are increasingly being supplemented with advanced analytical techniques to enhance accuracy and predictive power. This report aims to harness the capabilities of machine learning to effectively predict credit status, thereby deepening the understanding of factors that influence a firm's creditworthiness. Through this analysis, we intend to explore how well machine learning models can predict credit status based on historical financial data, identify which financial metrics are most influential in determining credit scores, and provide actionable insights to help stakeholders, including financial analysts, credit officers, and investors, make more informed decisions regarding credit risk and investment strategies. By integrating machine learning techniques with traditional financial analysis, this study seeks to bridge the gap between historical data interpretation and future credit status prediction, aiming to contribute significantly to the field of financial analytics by providing a model that not only predicts outcomes with high accuracy but also enhances our understanding of the dynamics at play in financial risk assessments.

2. DATASET OVERVIEW

The EM dataset contains 8,176 entries and 72 columns both numerical and categorical data, offering a detailed look into the credit status and characteristics of various firms in the UK, between 2019 and 2020. This dataset provides insights into several key areas like:

Credit Scoring and Risk: Metrics such as 'Credit Score', 'Credit Score Indicator', and 'Likelihood of Failure', the dataset gives a clear indication of the credit risk associated with each firm.

Financial Performance Over Time: The inclusion of annual data for both 2019 and 2020 allows for year-over-year comparisons, offering a window into how each firm's financial stability and performance have evolved. Metrics like 'Return on Total Assets', 'Net Assets Turnover', and various turnover ratios are indicative of operational efficiency and asset management.

Asset Management: Detailed breakdowns of assets, including fixed and intangible assets, provide deeper understanding of each firm's asset base and how effectively it's being utilised to generate revenue.

Liquidity and Solvency: Ratios such as 'Current Ratio' and 'Solvency Ratio' shed light on the firms' ability to meet short-term obligations and their overall financial health, respectively. These are crucial for understanding the immediate financial stability of a firm.

3.EXPLORATORY DATA ANALYSIS (EDA)

I.FEATURES SELECTION

To better understand the dataset and the relation between its variables an EDA was conducted considering the following features:

Credit Score: which reflects the creditworthiness of each firm, as a fundamental indicator of financial health and risk.

Return on Total Assets 2019 (ROTA2019): This ratio measures how effectively a company uses its assets to generate earnings. It is a significant indicator of a firm's operational efficiency and financial health prior to the COVID-19 pandemic, providing a baseline for understanding performance before economic disruptions. Analysing these features will offer insights into how they influence or correlate with the firms' credit status, which is essential for both descriptive analysis and predictive modelling.

II.DESCRIPTIVE STATISTICS AND VISUALISATION

Descriptive statistics were conducted to understand the central tendency and variability of both features. From the results below the average credit score across all firms is approximately 61.42, suggesting a moderate average creditworthiness among the firms while Standard Deviation is about 29.51, indicating that there is a moderate variation in the credit scores among the firms.

Fig 1: Descriptive Statistics

```
Descriptive Statistics for Creditscore and Return on Total Assets 2019:
       Creditscore  ReturnonTotalAssets2019
count  6773.000000              6773.000000
mean     61.423594                22.988502
std      29.509956               105.484711
min      15.000000              -912.765957
25%      33.000000                 0.058547
50%      59.000000                 6.909952
75%      92.000000                24.904707
max      99.000000               977.934426
```

The histogram below displays the bimodal nature of the distribution and helps us visualise the above results:

25th Percentile: 25% of firms have a credit score of 33 or lower.

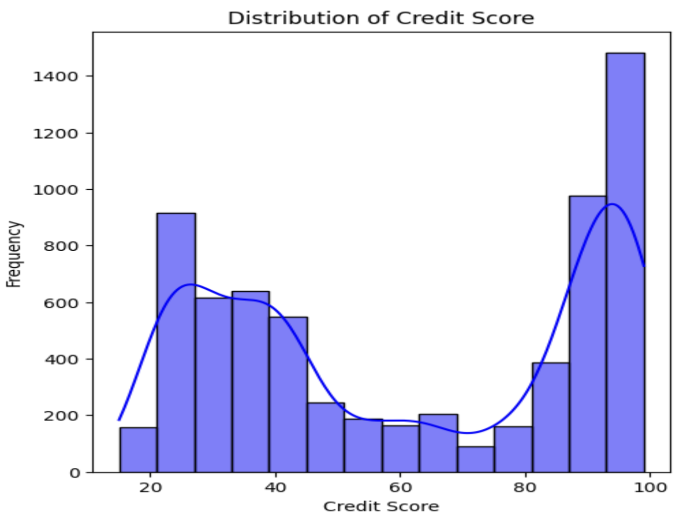
Median: The median score is 59, slightly below the mean, suggesting a skew towards lower scores.

75th Percentile: 75% of firms score 92 or lower, indicating few firms with very high credit scores.

Max and min credit score results: 99 and 33 respectively.

Fig 2: Histogram Credit Score

```
In [59]: #Plotting Histograms for Creditscore Feature
         plt.figure(figsize=(12, 6))
         plt.subplot(1, 2, 1)
         sns.histplot(data['Creditscore'], kde=True, color='blue')
         plt.title('Distribution of Credit Score')
         plt.xlabel('Credit Score')
         plt.ylabel('Frequency')

Out[59]: Text(0, 0.5, 'Frequency')
```



The creation of the credit score indicator helps visualise how firms are distributed within each risk category defined by the 'Credit score indicator'. The height of the bars indicates the count of firms that fall within each bin or interval of credit scores. As can be noticed most of the firms in the provided dataset are considered “secure”.

```
In [52]: #Histogram of 'Credit Score' grouped by 'Creditscoreindicator'
         chart = sns.FacetGrid(data, col='Creditscoreindicator')
         chart.map(sns.histplot, 'Creditscore')

Out[52]: <seaborn.axisgrid.FacetGrid at 0x15f799a10>
```

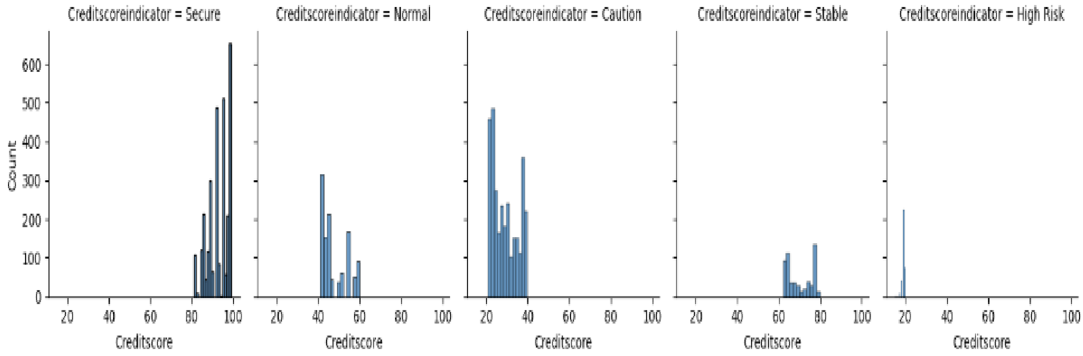


Fig 3: Histogram of 'Credit Score' grouped by 'Credit score indicator’.

The 2019 Return on Total Assets (ROTA) predominantly centers around 0%, indicating that most firms had negligible or negative returns. The median ROTA of 6.90% confirms this trend and is significantly lower than the mean, reflecting a left-skewed distribution influenced by a few firms with exceptionally high ROTAs.

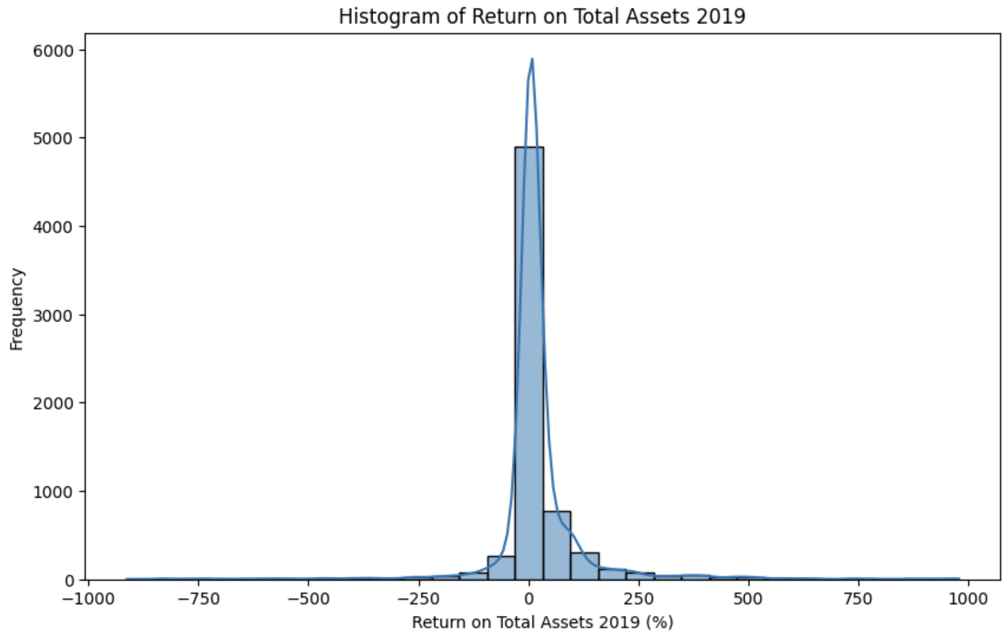
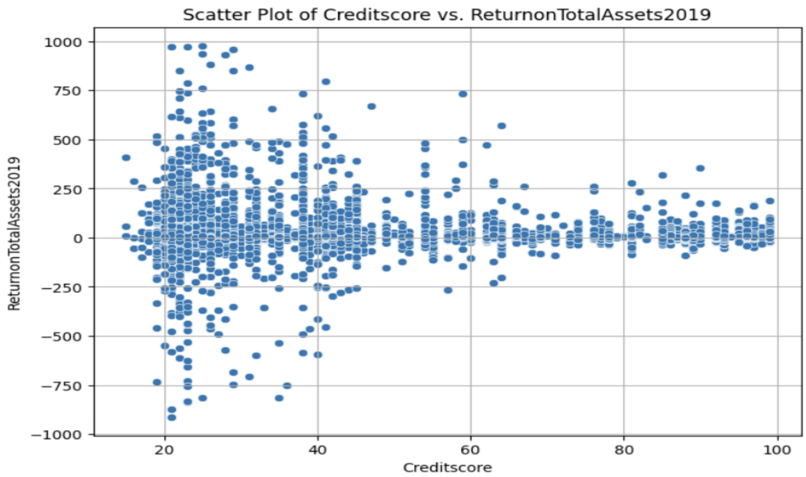


Fig 4: Histogram Return on Total Assets 2019

The scatter plot illustrates that there is no clear linear relationship between Credit Score and ROTA2019, suggesting that Credit Score may not strongly predict ROTA2019. The graph also indicates significant outliers in the ROTA2019 data.

Fig5: Scatter plot and Line plot between 'Credit Score' and 'ReturnonTotalAssets2019'

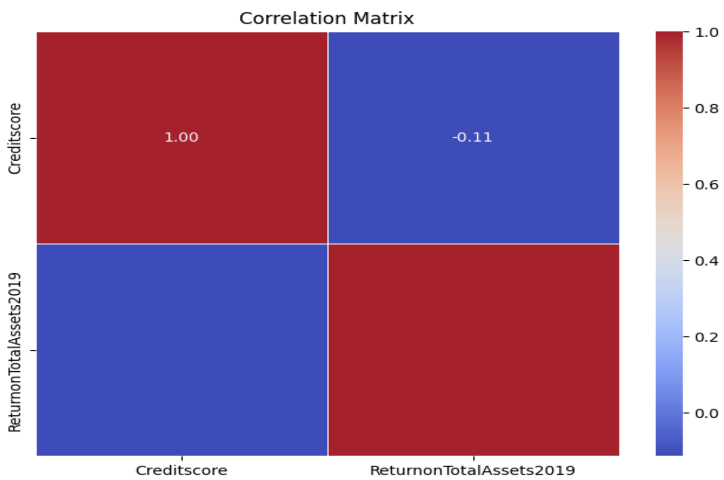
```
In [53]: #Scatter plot and Line plot between 'Creditscore' and 'ReturnonTotalAssets2019'
         plt.figure(figsize=(8, 6))
         sns.scatterplot(data=data, x='Creditscore', y='ReturnonTotalAssets2019')
         plt.title('Scatter Plot of Creditscore vs. ReturnonTotalAssets2019')
         plt.xlabel('Creditscore')
         plt.ylabel('ReturnonTotalAssets2019')
         plt.grid(True)
         plt.show()
```



The correlation matrix confirms the nonlinear relationship between the two features, with a negative correlation of -0.11, meaning that they tend to move in opposite directions and that the relationship is not strong enough to make reliable predictions on both variables.

Fig 6: Correlation Matrix between 'Credit Score' and 'ROTA2019'

```
Correlation matrix between Creditscore and Return on Total Assets 2019:
                         Creditscore  ReturnonTotalAssets2019
Creditscore                 1.000000                -0.113849
ReturnonTotalAssets2019    -0.113849                 1.000000
```



III.INSIGHTS FROM THE EDA

Based on these results we can conclude that the credit score is not influenced by the Return on total assets obtained in 2019 and vice versa. Given these we went on further exploring the correlation between credit score and other variables like the “likelihood of failure”, the EDIBITA and the Credit Limit GBP and find out that:

1. Credit Score and Credit Limit GBP have a linear relationship. indicating the firms with higher credit scores have higher credit limits.
2. The Credit Score and EBITDA graph shows non-linearity between the two features with the 2020 EBITDA being more volatile than the 2019 EBITDA and with some credit score ranges showing high EBITDA values especially in the high credit score range.
3. Credit score and likelihood of failure have an inverse relationship. As the credit score increases, the risk of failure is lowered and vice versa.

Fig 7: Scatter Plot ‘Credit Score’ and ‘Credit Limit GBP’

In [62]:
```
#Plot Creditscore and Credit Limit GBP
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()
```

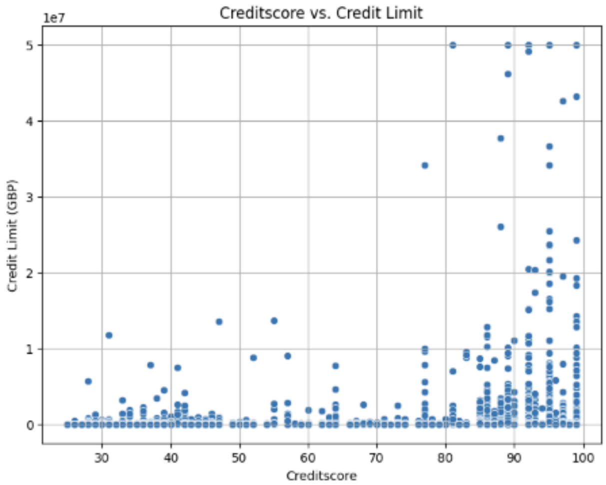


Fig 8: Line Plot ‘Credit Score’ and ‘EBITDA Comparison’

```
In [63]: # 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()
```

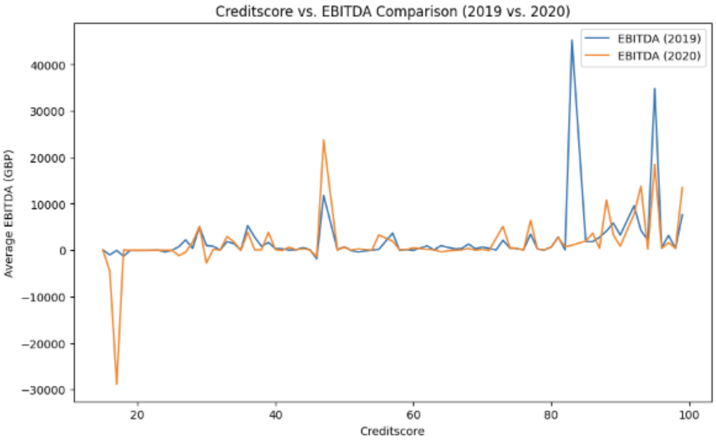
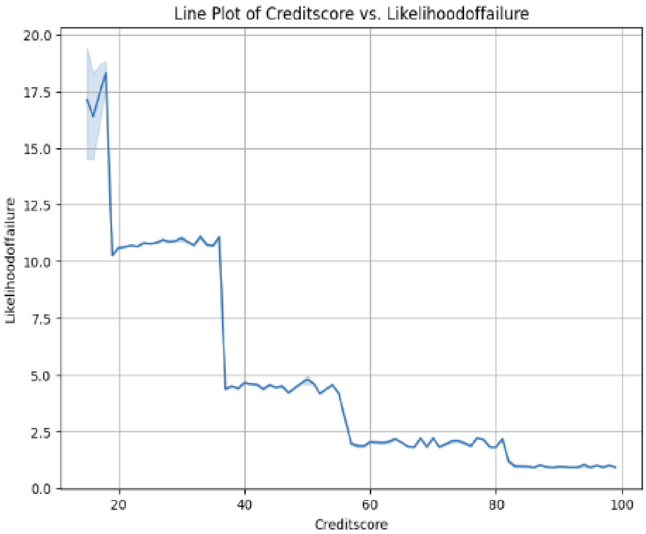


Fig 9: Line Plot 'Credit Score' and 'Likelihood of Failure'

```
In [64]: #Plot Creditscore and Likelihoodoffailure
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()
```



4.DATA PRE-PROCESSING

Data preprocessing is a crucial step in preparing data for machine learning models and ensuring accurate and reliable results. In this analysis, the dataset underwent several transformations to convert it into a more efficient and effective format. The following steps were carried out:

Handling Missing Data:

Missing or incomplete data in real-world datasets can bias models and skew predictions. To mitigate this, missing values in numerical columns were imputed with column means, preserving the dataset's mean and variance and providing a balanced solution to data incompleteness. *(Lawton, 2022).*

Removing Duplicates:

Duplicate records can skew analysis and conclusions. To avoid this, algorithms were used to detect and eliminate duplicates, ensuring the dataset contains only unique and relevant data.

Creating the Target Variable:

A binary target variable, 'CreditLevel,' was created from the 'Creditscore' column in the "EM.csv" dataset. Firms with a 'Creditscore' above the column mean were assigned a 'CreditLevel' of 1, and those below, 0. This transformation enables binary classification analysis.

Train-Test Split:

Before training the machine learning models, the preprocessed data was split into training and test sets. The training set, comprising 70% of the data, was used to fit the models, while the remaining 30% was allocated to the test set for evaluating the models' performance. This splitting ensures that the models are trained on a representative subset of the data and tested on an unseen portion, providing an unbiased assessment of their predictive capabilities. To ensure reproducibility, the `random_state` parameter was set to 123.

Feature Scaling:

Machine learning algorithms often require scaled input features to avoid bias and training issues. To address this, the dataset's numerical features were scaled using scikit-learn's StandardScaler, which standardizes features to have equal contributions to model predictions by subtracting the mean and dividing by the standard deviation.

5.METHODOLOGY

In this analysis, we employed two machine learning classification methods to predict the credit status of firms: the Extra Trees Classifier and Logistic Regression.

I.LOGISTIC REGRESSION

Logistic Regression is a statistical method for binary classification problems. It models the probability of an instance belonging to a particular class based on a linear combination of input features. The core concept is the logistic function (sigmoid function) which maps the output of the linear equation to a probability value between 0 and 1, representing the likelihood of the positive class thus interpreting it as a probability. (David G.kleinbaum,Mitchel Klein, 2010)The `Logistic Regression` class from the `sklearn. linear_model` module in scikit-learn was used for implementation, with a `random_state` parameter set for reproducibility.

II.EXTRA TREES

Extra Trees, short for Extremely Randomised Trees, is an ensemble machine learning algorithm that aggregates the results of multiple de-correlated decision trees collected in a "forest" to output its final prediction. It is a variant of the classic Random Forest algorithm. *(Thankachan, 2022)* where multiple decision trees are constructed, and their predictions are combined to improve accuracy and reduce overfitting. The key difference from Random Forest lies in the random selection of split points at each node, leading to better generalisation and capturing complex patterns. The `Extra Trees Classifier` class from the `sklearn.ensemble` module in scikit-learn was used for implementation, also with a `random_state` parameter set.

Both models were trained on the scaled training data (`X_train_scaled` and `y_train`) using the `fit` method. After training, the models can make predictions on new data using the `predict` method. (Dr.Yumei Yao, 2024)

The selection of these two methods, Logistic Regression and Extra Trees Classifier, represents a balance between a statistical approach and an ensemble learning approach. While Logistic Regression provides a straightforward and interpretable model, the Extra Trees Classifier can capture more complex patterns in the data and can also provide feature importance information. Using these two methods gave us a clear understanding of the detailed reasoning behind credit ratings of firms to a clear-cut prediction system.

III.MODEL EVALUATION

Evaluating the performance of our Extra Trees and Logistic Regression models is crucial to assess their effectiveness and identify potential areas for improvement. The following methods were used for model evaluation:

Cross-Validation: We have utilised cross-validation techniques to estimate the generalisation ability of our models on unseen data. This involves splitting the dataset into 10 folds. The model was trained on a subset of 9 folds and tested on the remaining 1-fold. This process was repeated for all 10 folds.

Classification Report: For both the Extra Trees and Logistic Regression models, we generated a classification report which summarised performance of both models on a classification task by providing metrics like precision, recall, F1-score, and support for each class.

Confusion Matrix: A confusion matrix plot was generated for both models. This table visually depicts the number of correct and incorrect predictions by the models. By examining the confusion matrix, we were able to identify the weaknesses in the models.

ROC Curve and AUC: We have generated ROC curves and calculated the corresponding AUC (Area Under the Curve) values for both models. Analysing the ROC curves and AUC values will allow us to compare the performance of the Extra Trees and Logistic Regression models in distinguishing between the classes.

6. RESULTS

In this analysis, the two machine learning classification methods were implemented in predicting the "Credit Level" target variable. Both models were trained on the scaled training data (X_train_scaled and y_train), and their performance was evaluated on the scaled test data (X_test_scaled and y_test).

I. CROSS-VALIDATION PERFORMANCE

The cross-validation results demonstrate the robustness and stability of the Extra Trees Classifier, with a mean accuracy of 0.9948 and a low standard deviation of 0.0035. The cross-validation results demonstrate the robustness and stability of the Logistics Regression, with a mean accuracy of 0.9937 and a low standard deviation of 0.0041.

II. CLASSIFICATION REPORT ON TEST SET

The Extra Trees Classifier demonstrated excellent performance, achieving a very high accuracy of 0.9951 on the test set. The classification report shows near-perfect precision and recall for both the high and low credit score classes. For the high credit score class, the model had a precision of 1.00 and a recall of 0.99, indicating it accurately identified nearly all the firms with high credit scores. Similarly, for the low credit score class, the model had a precision of 0.99 and a recall of 1.00, meaning it correctly classified almost all the firms with low credit scores. This exceptional precision and recall values across both classes demonstrate the Extra Trees Classifier's ability to make highly accurate predictions with very few false positives or false negatives.

```
Extra Trees Classifier Accuracy: 0.9951080309824705
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      1356
           1       0.99      1.00      0.99      1097

    accuracy                           1.00      2453
   macro avg       0.99      1.00      1.00      2453
weighted avg       1.00      1.00      1.00      2453
```

Fig (10): Classification Report (Extra Trees Classifier)

The Logistic Regression model also performed very well, achieving a high overall accuracy of 0.99. The classification report shows the model had a precision and recall of over 0.99 for both the high and low credit score classes. This means the model had a low rate of false positives and was able to effectively capture most true positives in each class. The high F1-scores of 0.99 for both classes further confirm the model's balanced performance between precision and recall. These results indicate the Logistic Regression model is highly effective at differentiating between the credit score categories.

```
Logistic Regression Accuracy: 0.9914390542193233
              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
```

Fig (11): Classification Report (Logistic Regression)

III.CONFUSION MATRIX PLOT

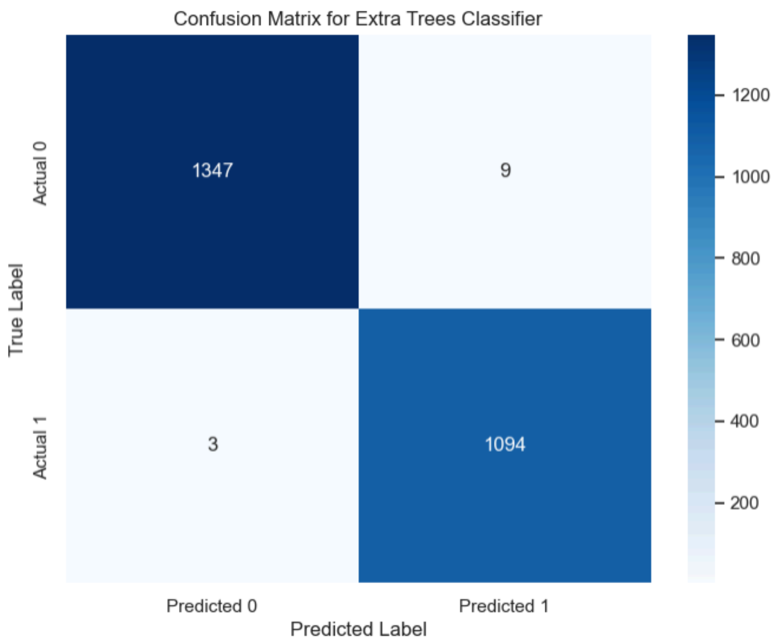


Fig (12): Confusion Matrix Plot for Extra Trees Classifier

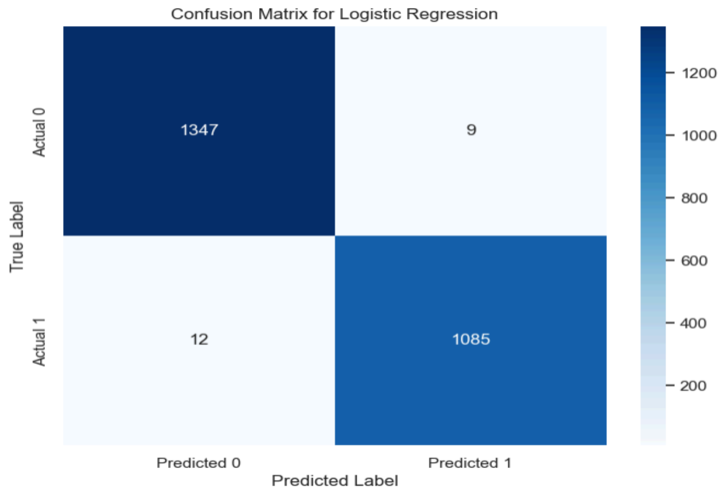


Fig (13): Confusion Matrix Plot for Logistic Regression

The results from the classification report are further supported by the confusion matrix, which shows only a small number of firms were misclassified between the two credit score categories.

IV.ROC CURVE PLOT

The ROC curve plot for the Extra Trees Classifier demonstrates its excellent performance in classifying firms based on credit scores. The model achieved an AUC of 0.99, indicating it has an outstanding ability to differentiate between high and low credit scores, far exceeding the performance of a random classifier (AUC of 0.5). The shape of the ROC curve, rising sharply towards the top-left corner, shows the model can accurately identify true positives while maintaining a low false positive rate. This makes the Extra Trees Classifier a highly effective model for predicting credit status.

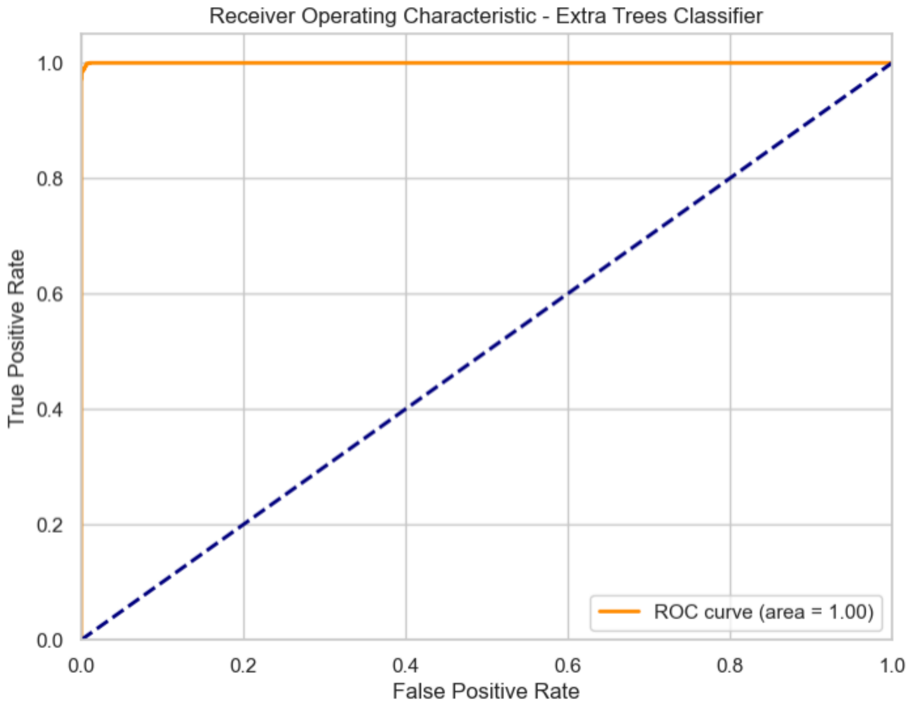


Fig (14): ROC curve plot for Extra Trees Classifier

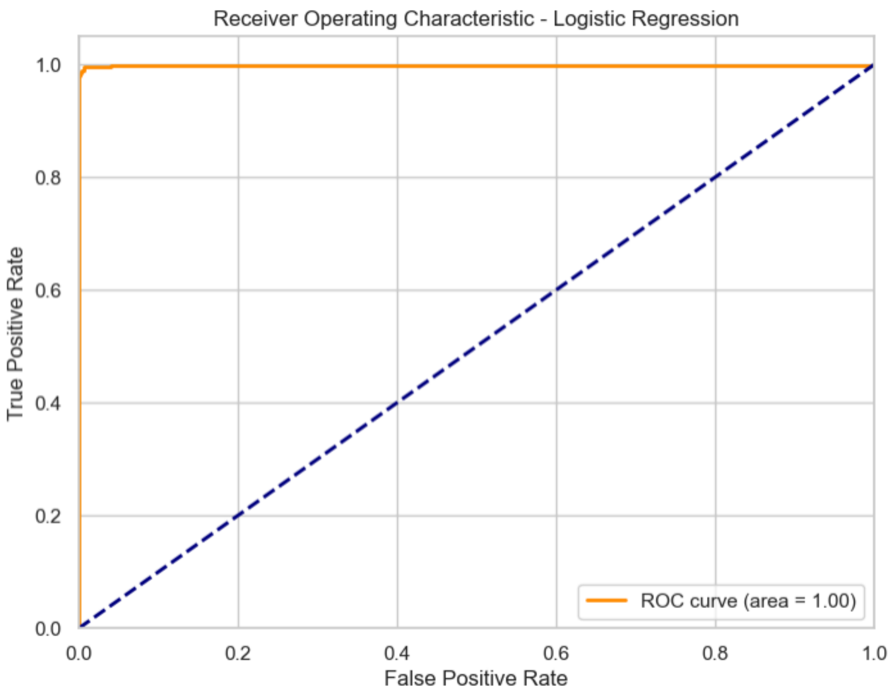


Fig (15): ROC curve plot for Logistic Regression Method

7. ANALYSIS

Both the Extra Trees Classifier and Logistic Regression models performed exceptionally well in predicting the "Credit Level" target variable, achieving accuracy scores above 99%. The Extra Trees Classifier performed better than the Logistic Regression model, in terms of overall accuracy and cross-validation results. Specifically, the Extra Trees Classifier achieved an impressive accuracy of 99.51% on the test set, while its cross-validation mean accuracy stood at an exceptional 99.48% and standard deviation of 0.0035.

In comparison, the Logistic Regression model, also highly accurate, with a test set accuracy of 99.14% and a cross-validation mean accuracy of 99.37% and standard deviation of 0.0041, which was higher than that of the Extra Trees Classifier.

Based on these results, it is evident that machine learning techniques can predict credit status of firms with a high degree of accuracy, so the assumption that "Machine learning can predict the credit status of firms" is supported. Overall, both models demonstrated strong predictive capabilities, but the Extra Trees Classifier emerged as the more robust and accurate method for classifying firms based on their credit level in this analysis.

Both the Extra Trees Classifier and Logistic Regression models exhibited remarkable performance in predicting the "Credit Level" target variable, with accuracy scores surpassing the 99% mark. While both models exhibited excellent predictive capabilities, the Extra Trees Classifier emerged as the more robust and accurate method for classifying firms based on their credit level in this analysis.

8.FEATURE IMPORTANCE

The feature importance analysis for the Extra Trees Classifier showed that "Credit score" was the most important feature in predicting the "CreditLevel" target variable. This was closely followed by "Likelihood of failure" and "Previous credit score". This aligns with the notion that credit score is a key factor in determining a firm's credit status. Credit score is widely recognized as one of the primary indicators used by lenders and financial institutions to assess creditworthiness and risk.

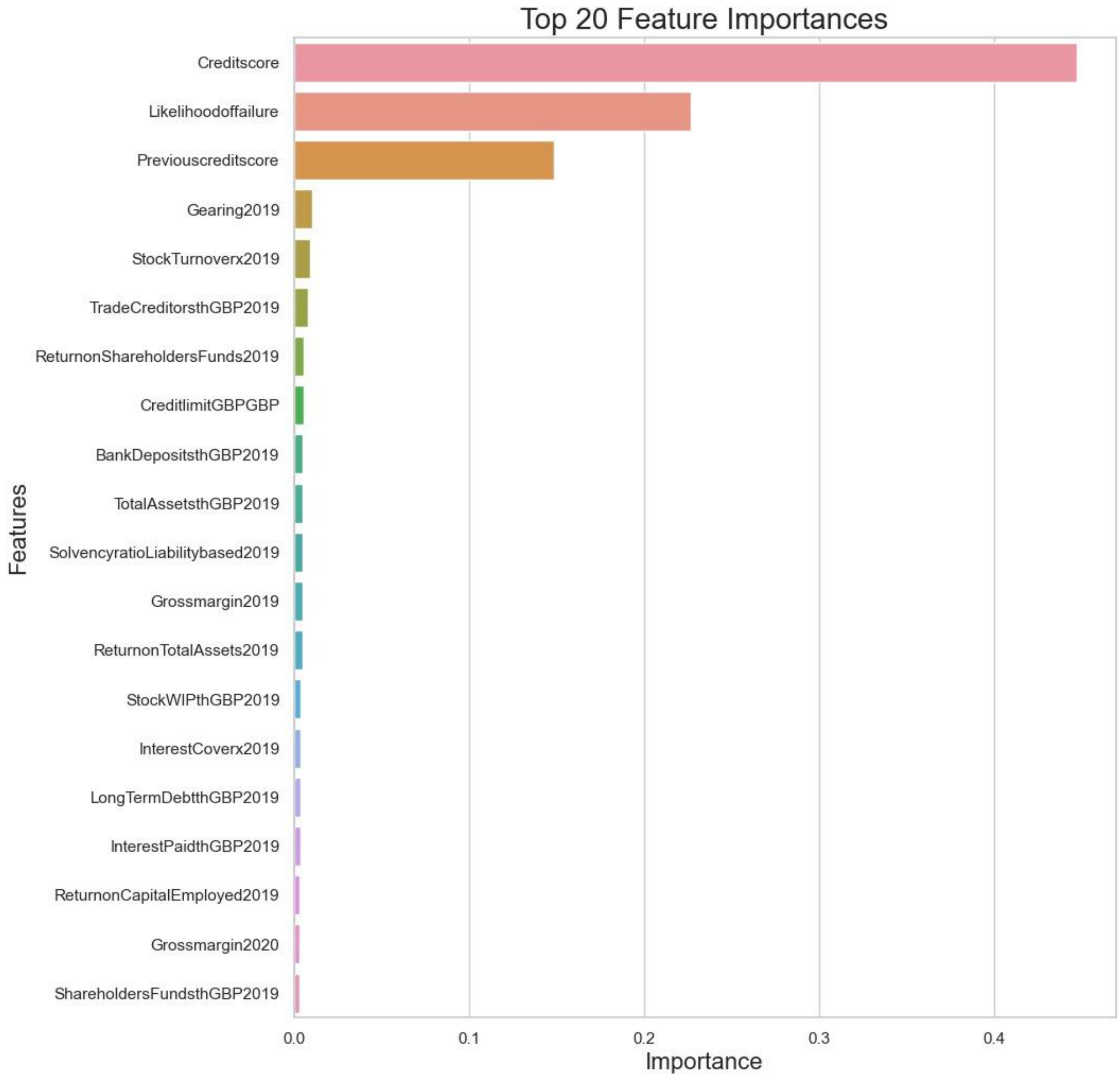


Fig (16): Feature Importance Analysis.

9. DISCUSSION

It is evident that machine learning can predict credit status effectively. The high accuracy and precision demonstrated by the models, particularly the Extra Trees Classifier, bolster the confidence in machine learning as a powerful tool for credit evaluation.

The empirical results form a strong foundation for this assertion. With an accuracy exceeding 99%, machine learning models, specifically the Extra Trees Classifier, have proven capable of discerning the nuanced patterns that characterise a firm's creditworthiness. The significant feature importance of Credit score aligns with traditional credit assessment's reliance on this metric, validating the models' ability to replicate and potentially enhance human expertise in credit rating. Additionally, literature in the field of financial risk assessment supports the use of machine learning, citing its ability to handle large datasets and uncover complex interactions between variables, often outperforming traditional statistical methods.)

However, the effectiveness of machine learning is not without potential limitations and biases. One such limitation is the quality and representativeness of the data; if the training data is not comprehensive or current, model predictions may not accurately reflect the actual risk.

Furthermore, machine learning models can unintentionally amplify biases present in the training data, leading to unfair or unethical credit assessments. Literature on Artificial Intelligence ethics emphasises the need for transparency and fairness in machine learning models, especially in high-stakes applications like credit scoring, where biased decisions can have significant repercussions.

Overfitting is another risk associated with classification methods especially in cases where the model has many predictor variables. This often leads to perfect performance on the training data (reflected in the ROC curves) but poor performance on unseen data with

10. CONCLUSION

The analysis conducted using machine learning to predict credit status has yielded definitive insights. The Extra Trees Classifier and Logistic Regression models both demonstrated exceptional accuracy, with the former providing key insights through feature importance, particularly highlighting the predominance of Credit score in determining credit levels. The results have reinforced the premise that machine learning can effectively predict credit status, supported by both the empirical outcomes of the analysis and existing financial risk assessment literature.

In terms of practical implications, these findings suggest that stakeholders in the financial sector can leverage machine learning to enhance credit evaluation processes. The high predictive accuracy of the models indicates that machine learning can serve as a robust tool for credit decision-making, potentially leading to more informed and data-driven financial judgments. For future research, the exploration into machine learning applications in finance can expand into assessing how these models handle evolving economic conditions and integrating alternative data sources to mitigate biases.

The debate on the effectiveness of machine learning, while leaning towards its efficacy, also brings to light the importance of model transparency and ethical considerations. As machine learning becomes increasingly integrated into financial practices, it is crucial to ensure that these systems are not only accurate but also fair and equitable. Therefore, while the analytical prowess of machine learning is clear, it must be harnessed with caution, keeping in mind the responsibility towards ethical implications and the need for continuous oversight.

11. REFERENCES

Lawton, G. (2022) Data pre-processing: Definition, key steps and concepts, Data Management. Available at: https://www.techtarget.com/searchdatamanagement/definition/data-preprocessing (Accessed: 16 April 2024).

Thankachan, K. (2022) What? when? how? Extra Trees Classifier, Medium. Available at: https://towardsdatascience.com/what-when-how-extratrees-classifier-c939f905851c (Accessed: 12 April 2024).

David G.kleinbaum,Mitchel Klein. (2010). Logistic Regression. USA: springer New York,NY.

Dr.Yumei Yao. (2024, April 18). Seminar 7. AI&ML. London, UK, UK: University of Westminster.

12. APPENDICES

Contribution (Sarika Patel)

I detail my contributions as follows

My contribution to the report includes looking the entire **coding process**, from developing the codebase for data preprocessing to implementing and evaluating machine learning models using scikit-learn libraries. I did bit of methodology section, specifically in selecting and justifying the use of Logistic Regression and Extra Trees Classifier and designed the model evaluation strategy.

I led the analysis of model outputs, **interpreting results** from cross-validation and test set evaluations to assess model performance. I analysed and highlighted the importance of 'Credit Score' in **feature importance**, drafted the **discussion** to emphasize the practical applications of machine learning in credit scoring, addressed potential biases, and incorporated relevant literature. I also composed the **conclusion**, summarizing key findings and suggesting future research directions, and was responsible for the **final review and editing** of the document to ensure clarity, coherence, and academic integrity.

We would like to thank Professor Dr. Yumei Yao for placing us in this group and assigning us this collaborative task, which has greatly enhanced our learning experience.

EXPLORATORY DATA ANALYSIS

In [50]: `data.info()`

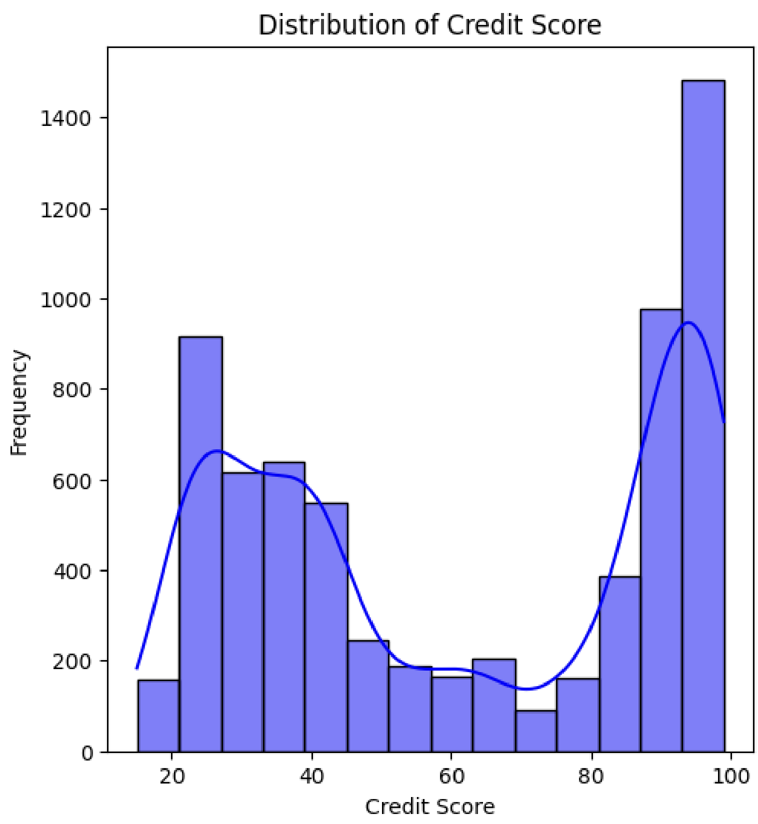
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8176 entries, 0 to 8175
Data columns (total 72 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   Creditscore                        8176 non-null   int64  
 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
 24  FixedAssetsthGBP2020               2783 non-null   float64
 25  FixedAssetsthGBP2019               6127 non-null   float64
 26  IntangibleAssetsthGBP2020          501 non-null    float64
 27  IntangibleAssetsthGBP2019          1697 non-null   float64
 28  TotalAssetsthGBP2020               3877 non-null   float64
 29  TotalAssetsthGBP2019               7838 non-null   float64
 30  OrdinarySharesthGBP2020            1970 non-null   float64
 31  OrdinarySharesthGBP2019            5016 non-null   float64
 32  RetainedProfitLossGBP2020          3145 non-null   float64
 33  RetainedProfitLossthGBP2019        6985 non-null   float64
 34  CurrentAssetsthGBP2020             3690 non-null   float64
 35  CurrentAssetsthGBP2019             7598 non-null   float64
 36  CurrentLiabilitiesthGBP2020        3105 non-null   float64
 37  CurrentLiabilitiesthGBP2019        6866 non-null   float64
 38  LiquidityRatiox2020                3014 non-null   float64
 39  LiquidityRatiox2019                6718 non-null   float64
 40  LongTermDebtthGBP2020              789 non-null    float64
 41  LongTermDebtthGBP2019              2077 non-null   float64
 42  ShortTermLoansOverdrafts2020       1357 non-null   float64
 43  ShortTermLoansOverdrafts2019       3996 non-null   float64
 44  CapitalExpenditure2020             219 non-null    float64
 45  CapitalExpenditure2019             1370 non-null   float64
 46  InterestPaidthGBP2020              994 non-null    float64
 47  InterestPaidthGBP2019              3016 non-null   float64
 48  InterestCoverx2020                 927 non-null    float64
 49  InterestCoverx2019                 2746 non-null   float64
 50  Numberofemployees2020              2858 non-null   float64
 51  Numberofemployees2019              5494 non-null   float64
 52  IssuedCapitalthGBP2020             2027 non-null   float64
 53  IssuedCapitalthGBP2019             5110 non-null   float64
 54  AdministrationExpensesthGBP2020    2987 non-null   float64
 55  AdministrationExpensesthGBP2019    6868 non-null   float64
 56  EBITDAthGBP2020                    3170 non-null   float64
 57  EBITDAthGBP2019                    7090 non-null   float64
 58  BankDepositsthGBP2020              2093 non-null   float64
 59  BankDepositsthGBP2019              5197 non-null   float64
 60  Gearing2020                        1586 non-null   float64
 61  Gearing2019                        4012 non-null   float64
 62  ShareholdersFundsthGBP2020         3854 non-null   float64
 63  ShareholdersFundsthGBP2019         7707 non-null   float64
 64  TaxationthGBP2020                  2020 non-null   float64
 65  TaxationthGBP2019                  5075 non-null   float64
 66  Grossmargin2020                    1676 non-null   float64
 67  Grossmargin2019                    4552 non-null   float64
 68  Currentratiox2020                  3018 non-null   float64
 69  Currentratiox2019                  6724 non-null   float64
 70  SolvencyratioLiabilitybased2020    1385 non-null   float64
 71  SolvencyratioLiabilitybased2019    3075 non-null   float64
dtypes: float64(69), int64(1), object(2)
memory usage: 4.5+ MB
```

In [58]:
```
import pandas as pd

# Load the dataset
data = pd.read_csv('EM.csv')  # Make sure to update the path to where your dataset is stored

# Dropping rows with missing values to ensure accurate statistics
data.dropna(subset=['Creditscore', 'ReturnonTotalAssets2019'], inplace=True)

# Generate descriptive statistics
descriptive_stats = data[['Creditscore', 'ReturnonTotalAssets2019']].describe()

# Print the descriptive statistics
print("Descriptive Statistics for Creditscore and Return on Total Assets 2019:")
print(descriptive_stats)
```

```
Descriptive Statistics for Creditscore and Return on Total Assets 2019:
       Creditscore  ReturnonTotalAssets2019
count  6773.000000              6773.000000
mean     61.423594                22.988502
std      29.509956               105.484711
min      15.000000              -912.765957
25%      33.000000                 0.058547
50%      59.000000                 6.909952
75%      92.000000                24.904707
max      99.000000               977.934426
```

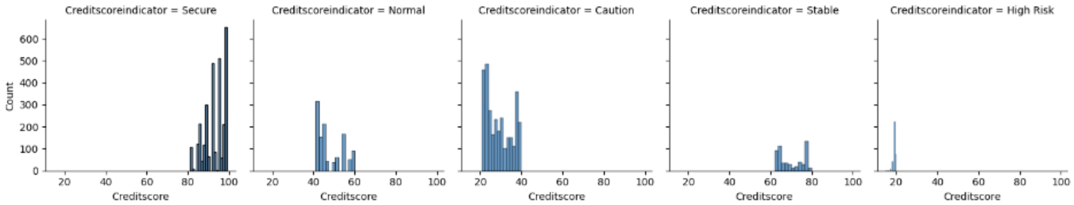
In [33]:
```
#Plotting Histograms for Creditscore Feature
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(data['Creditscore'], kde=True, color='blue')
plt.title('Distribution of Credit Score')
plt.xlabel('Credit Score')
plt.ylabel('Frequency')
```

Out[33]: Text(0, 0.5, 'Frequency')

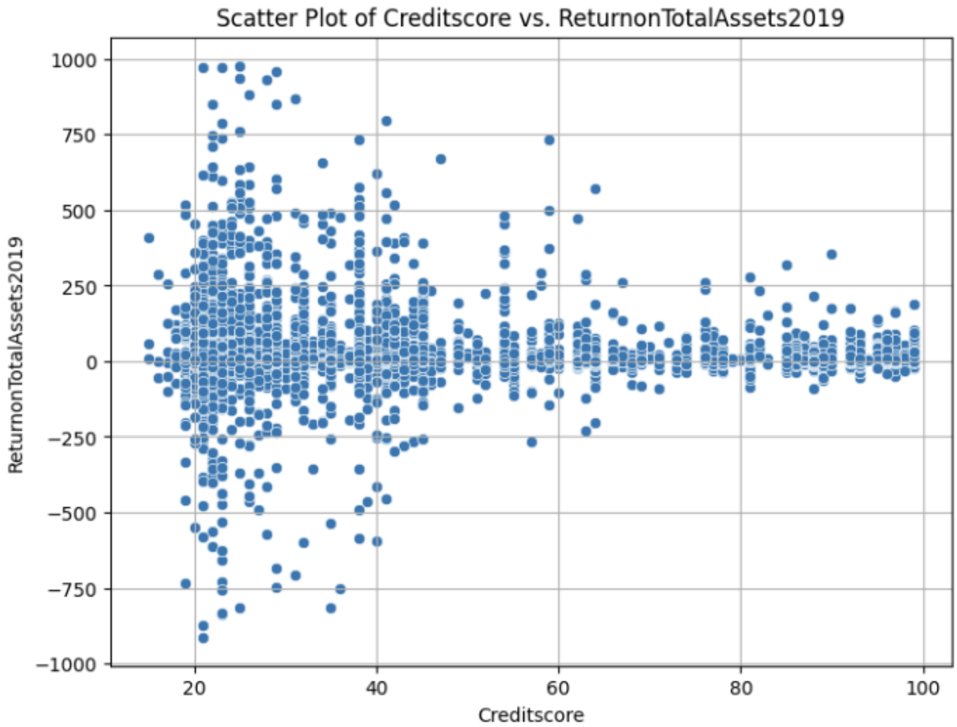


In [52]:
```
#Histogram of 'Credit Score' grouped by 'Creditscoreindicator'
chart = sns.FacetGrid(data, col='Creditscoreindicator')
chart.map(sns.histplot, 'Creditscore')
```

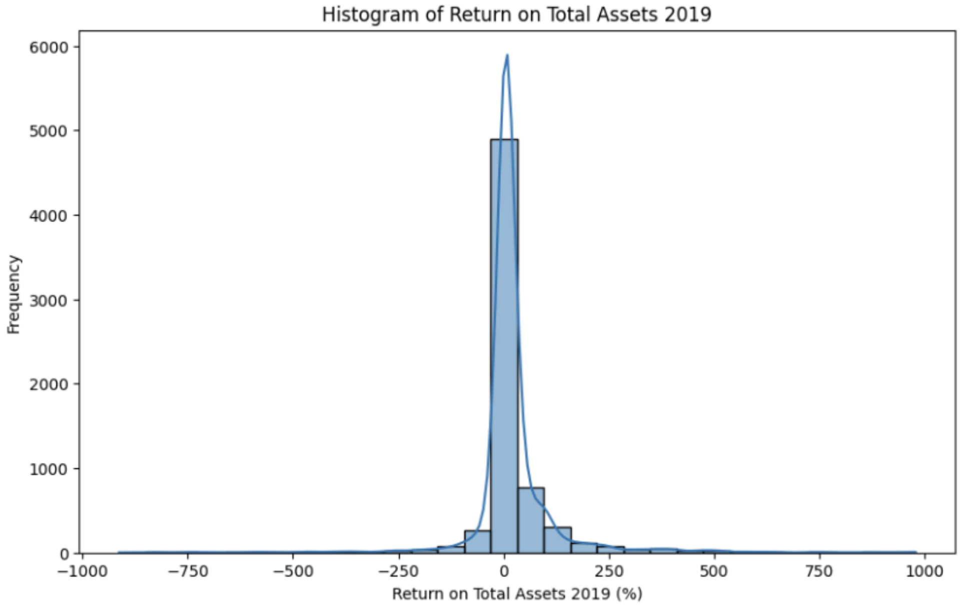
Out[52]: <seaborn.axisgrid.FacetGrid at 0x15f799a10>



In [53]:
```
#Scatter plot and Line plot between 'Creditscore' and 'ReturnonTotalAssets2019'
plt.figure(figsize=(8, 6))
sns.scatterplot(data=data, x='Creditscore', y='ReturnonTotalAssets2019')
plt.title('Scatter Plot of Creditscore vs. ReturnonTotalAssets2019')
plt.xlabel('Creditscore')
plt.ylabel('ReturnonTotalAssets2019')
plt.grid(True)
plt.show()
```

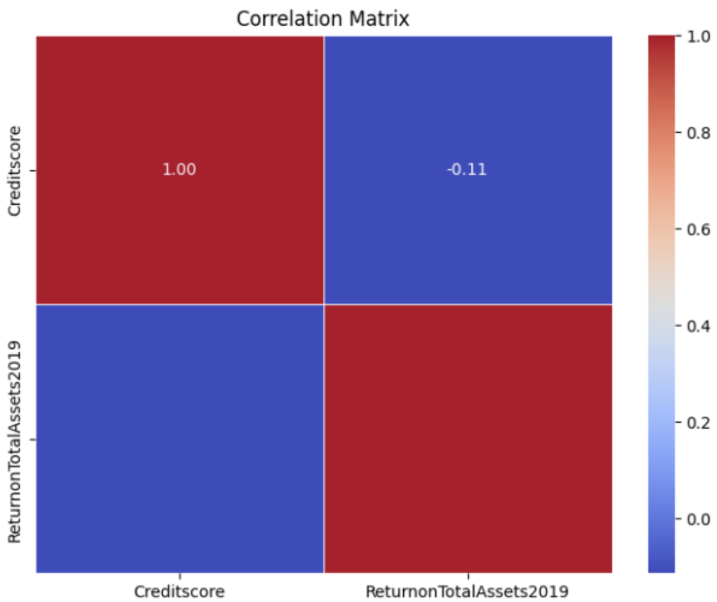


In [61]:
```python
# Histogram for ROTA2019
plt.figure(figsize=(10, 6))
sns.histplot(data['ReturnonTotalAssets2019'], kde=True, bins=30)
plt.title('Histogram of Return on Total Assets 2019')
plt.xlabel('Return on Total Assets 2019 (%)')
plt.ylabel('Frequency')
plt.show()
```

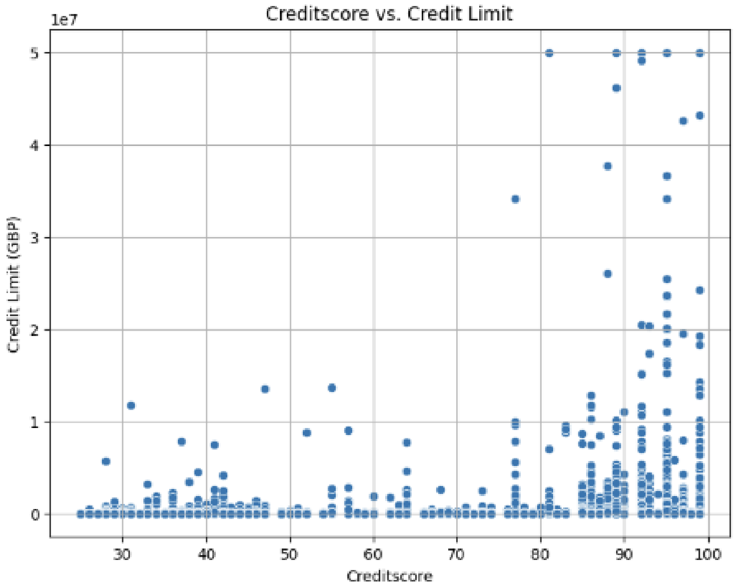


In [60]:
```python
# Fig 7: Correlation matrix of data variables

correlation_matrix = data[['Creditscore', 'ReturnonTotalAssets2019']].corr()
print("\nCorrelation matrix between Creditscore and Return on Total Assets 2019:")
print(correlation_matrix)

# Visualize the correlation matrix
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Matrix')
plt.show()
```

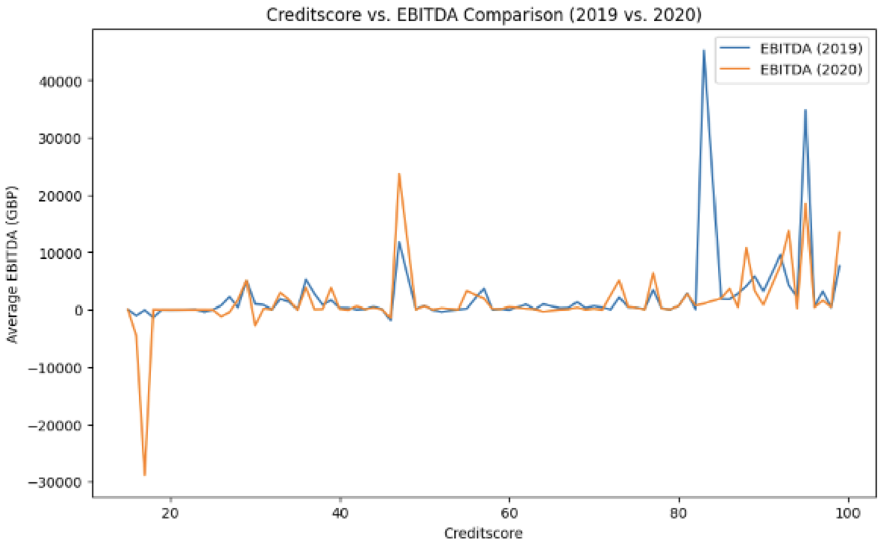
```
Correlation matrix between Creditscore and Return on Total Assets 2019:
                         Creditscore  ReturnonTotalAssets2019
Creditscore                 1.000000                -0.113849
ReturnonTotalAssets2019    -0.113849                 1.000000
```



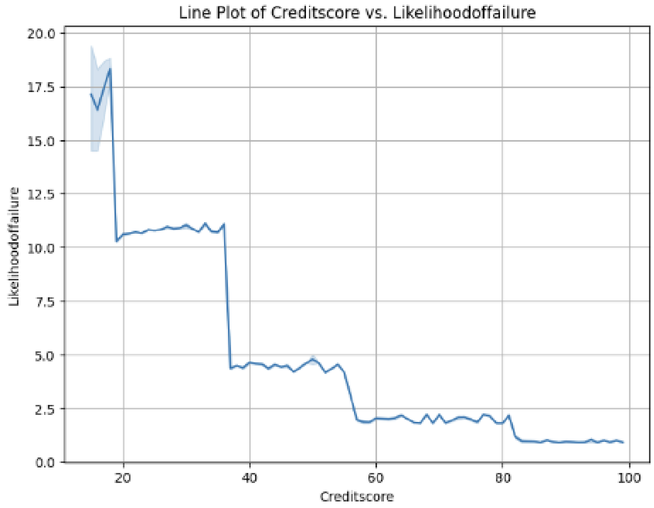
In [62]:
```
#Plot Creditscore and Credit Limit GBP
plt.figure(figsize=(8, 6))
sns.scatterplot(data=data_all, x='Creditscore', y='CreditlimitGBPGBP')
plt.title('Creditscore vs. Credit Limit')
plt.xlabel('Creditscore')
plt.ylabel('Credit Limit (GBP)')
plt.grid(True)
plt.show()
```



In [63]:
```
# 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 [64]:
```python
#Plot Creditscore and Likelihoodoffailure
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()
```



CREATING 'CREDIT LEVEL' TARGET VARIABLE

In [1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
data = pd.read_csv('EM.csv')  # Make sure the file name is correct

# Display the first few rows to check the data
data.head()
```

Out[1]:

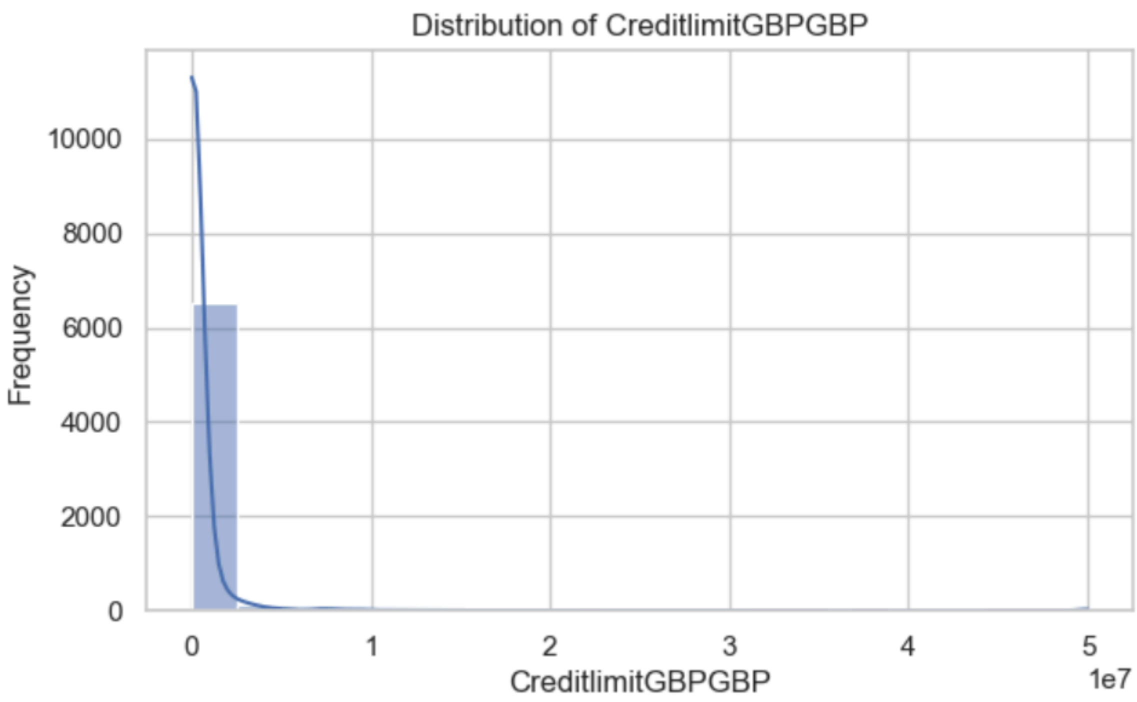
	Creditscore	Creditscoreindicator	Likelihoodoffailure	CreditlimitGBPGBP	Previouscreditscore	SMEindicator	ReturnonTotalAssets2020	Retu
0	92	Secure	0.9	50000000.0	95.0	No	NaN	
1	92	Secure	0.9	50000000.0	99.0	No	-0.130484	
2	95	Secure	0.9	16574000.0	99.0	No	NaN	
3	89	Secure	0.9	5380000.0	92.0	No	NaN	
4	99	Secure	0.9	50000000.0	99.0	No	21.144665	

5 rows × 72 columns

In [2]:
```
# Calculate descriptive statistics for the Creditscore and CreditlimitGBPGBP

stats = data[['Creditscore', 'CreditlimitGBPGBP']].describe()
print(stats)
```

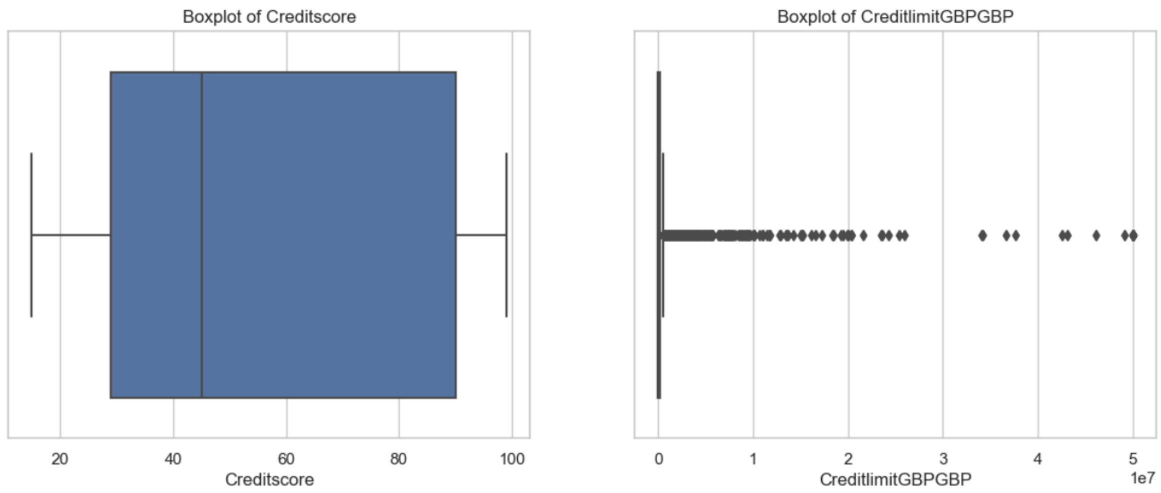
```
       Creditscore  CreditlimitGBPGBP
count  8176.000000       6.771000e+03
mean     57.300514       5.292997e+05
std      29.730987       2.983264e+06
min      15.000000       5.000000e+02
25%      29.000000       5.000000e+02
50%      45.000000       1.543000e+04
75%      90.000000       2.342970e+05
max      99.000000       5.000000e+07
```

In [3]:
```
# Set the aesthetic style of the plots
sns.set(style="whitegrid")

# Plotting the distribution of Creditscore
plt.figure(figsize=(7, 4))
sns.histplot(data['Creditscore'], bins=20, kde=True)
plt.title('Distribution of Creditscore')
plt.xlabel('Creditscore')
plt.ylabel('Frequency')
plt.show()

# Plotting the distribution of CreditlimitGBPGBP
plt.figure(figsize=(7, 4))
sns.histplot(data['CreditlimitGBPGBP'], bins=20, kde=True)
plt.title('Distribution of CreditlimitGBPGBP')
plt.xlabel('CreditlimitGBPGBP')
plt.ylabel('Frequency')
plt.show()
```



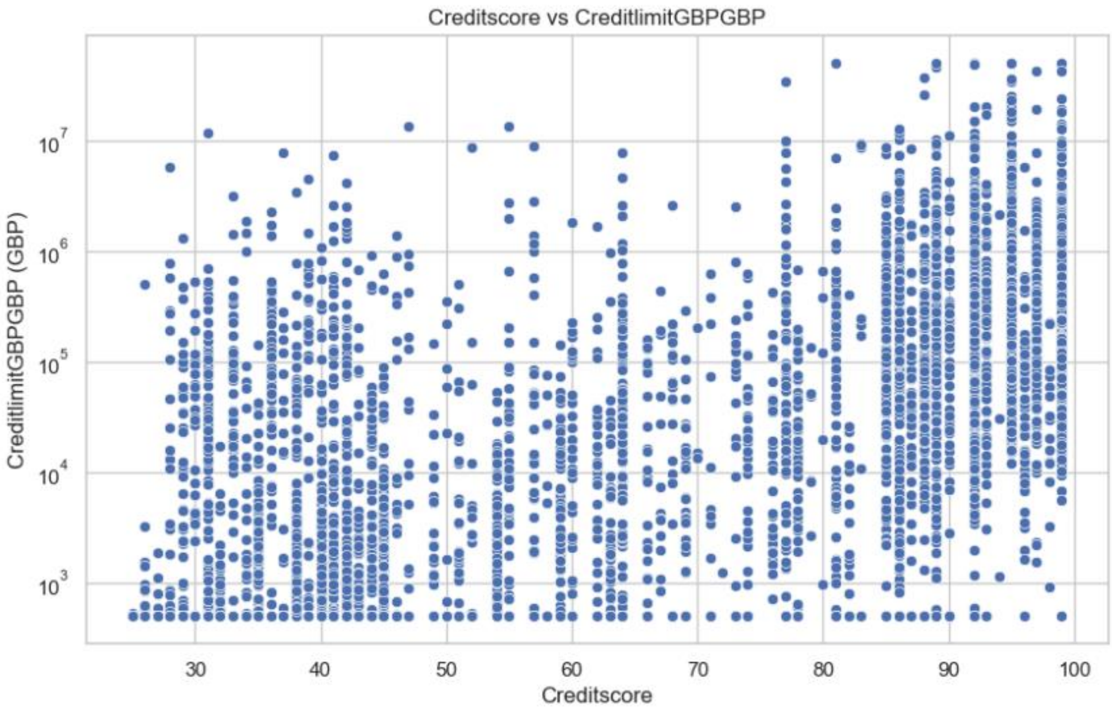
In [4]:

```python
# Plotting boxplots for visualizing outliers
plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 1)
sns.boxplot(x=data['Creditscore'])
plt.title('Boxplot of Creditscore')

plt.subplot(1, 2, 2)
sns.boxplot(x=data['CreditlimitGBPGBP'])
plt.title('Boxplot of CreditlimitGBPGBP')
plt.show()
```



In [5]:

```python
# Scatter plot to explore the relationship between Creditscore and CreditlimitGBPGBP
plt.figure(figsize=(10, 6))
sns.scatterplot(x=data['Creditscore'], y=data['CreditlimitGBPGBP'])
plt.title('Creditscore vs CreditlimitGBPGBP')
plt.xlabel('Creditscore')
plt.ylabel('CreditlimitGBPGBP (GBP)')
plt.yscale('log')  # Use logarithmic scale for y-axis
plt.grid(True)
plt.show()
```



2.DATA CLEANING AND PROCESSING FOR 70&30% TEST AND TRAINIGN DATA

In [6]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load the dataset
data = pd.read_csv('EM.csv')

# Remove non-numerical columns
numerical_data = data.select_dtypes(include=['number'])

# Handle missing values by filling with the mean of each column
numerical_data_filled = numerical_data.fillna(numerical_data.mean())

# Display the cleaned data
print(numerical_data_filled.head())
```

```
   Creditscore  Likelihoodoffailure  CreditlimitGBPGBP  Previouscreditscore  \
0           92                  0.9         50000000.0                 95.0   
1           92                  0.9         50000000.0                 99.0   
2           95                  0.9         16574000.0                 99.0   
3           89                  0.9          5380000.0                 92.0   
4           99                  0.9         50000000.0                 99.0   

   ReturnonTotalAssets2020  ReturnonTotalAssets2019  \
0                27.903380                 5.614489   
1                -0.130484                 2.832746   
2                27.903380                 3.817802   
3                27.903380                -5.702719   
4                21.144665                26.910621   

   ReturnonShareholdersFunds2020  ReturnonShareholdersFunds2019  \
0                      65.895390                      10.284360   
1                      -0.283735                       7.000426   
2                      65.895390                      18.074145   
3                      65.895390                     -67.554766   
4                     169.535674                     130.534489   

   ReturnonCapitalEmployed2020  ReturnonCapitalEmployed2019  ...  \
0                    51.135009                     8.326938  ...   
1                    -0.175859                     4.070147  ...   
2                    51.135009                    15.152711  ...   
3                    51.135009                   -15.251972  ...   
4                    28.898498                    45.933410  ...   

   ShareholdersFundsthGBP2020  ShareholdersFundsthGBP2019  TaxationthGBP2020  \
0                8.513789e+03                   2110000.0         -536.39174   
1                2.713800e+06                   2582700.0       -14200.00000   
2                8.513789e+03                    397247.0         -536.39174   
3                8.513789e+03                    168900.0         -536.39174   
4                4.415000e+05                    553800.0      -138300.00000   

   TaxationthGBP2019  Grossmargin2020  Grossmargin2019  Currentratiox2020  \
0           -50000.0        44.991604        37.888106           4.137871   
1           -58000.0        29.740966        29.250830           1.474780   
2           -19589.0        44.991604        13.304340           4.137871   
3            -3300.0        44.991604        10.490224           4.137871   
4          -132500.0        38.453425        35.374574           2.058749   
```

```
   Currentratiox2019  SolvencyratioLiabilitybased2020  \
0           1.224782                        35.770585   
1           1.300454                        85.144166   
2           0.892335                        35.770585   
3           0.921290                        35.770585   
4           1.826697                        14.249290   

   SolvencyratioLiabilitybased2019  
0                        36.916356  
1                        67.969367  
2                        26.779674  
3                         9.219936  
4                        25.969519  

[5 rows x 70 columns]
```

```
Create credit level feature
```

TRAINING SET AND TEST SET DATA

In [9]:
```python
# Calculate the mean of Creditscore
creditscore_mean = numerical_data_filled['Creditscore'].mean()

# Create 'CreditLevel' where 1 if Creditscore > mean, else 0
numerical_data_filled['CreditLevel'] = (numerical_data_filled['Creditscore'] > creditscore_mean).astype(int)

# Set 'CreditLevel' as the target variable Y and the rest as X
X = numerical_data_filled.drop(columns=['CreditLevel'])
Y = numerical_data_filled['CreditLevel']
```

In [10]:
```
# Split the data into 70% training and 30% testing set with a random state of 123
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30, random_state=123)

# Normalize the features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Display the shape of the datasets to verify
print("Training set (X_train_scaled):", X_train_scaled.shape)
print("Test set (X_test_scaled):", X_test_scaled.shape)
```

```
Training set (X_train_scaled): (5723, 70)
Test set (X_test_scaled): (2453, 70)
```

EXTRA TREES CLASSIFIER

In [11]:
```
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import accuracy_score, classification_report

# Instantiate and train the Extra Trees model
et_classifier = ExtraTreesClassifier(random_state=123)
et_classifier.fit(X_train_scaled, Y_train)

# Predict on the test data
et_predictions = et_classifier.predict(X_test_scaled)

# Evaluate the model
et_accuracy = accuracy_score(Y_test, et_predictions)
print("Extra Trees Classifier Accuracy:", et_accuracy)
print(classification_report(Y_test, et_predictions))

# Feature Importance from Extra Trees
et_feature_importances = et_classifier.feature_importances_
features = X.columns
et_feature_importance = pd.DataFrame({'Feature': features, 'Importance': et_feature_importances}).sort_values(by='
print(et_feature_importance)
```

```
Extra Trees Classifier Accuracy: 0.9951080309824705
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      1356
           1       0.99      1.00      0.99      1097

    accuracy                           1.00      2453
   macro avg       0.99      1.00      1.00      2453
weighted avg       1.00      1.00      1.00      2453

                  Feature  Importance
0             Creditscore    0.446949
1    Likelihoodoffailure    0.226676
3     Previouscreditscore    0.148910
59            Gearing2019    0.010693
17      StockTurnoverx2019    0.009203
..                    ...         ...
46    InterestCoverx2020    0.000984
62      TaxationthGBP2020    0.000923
54         EBITDAthGBP2020    0.000883
24 IntangibleAssetsthGBP2020    0.000823
42   CapitalExpenditure2020    0.000404

[70 rows x 2 columns]
```

LOGISTIC REGRESSION

In [12]:
```
from sklearn.linear_model import LogisticRegression

# Instantiate and train the Logistic Regression model
log_reg = LogisticRegression(random_state=123)
log_reg.fit(X_train_scaled, Y_train)

# Predict on the test data
log_reg_predictions = log_reg.predict(X_test_scaled)

# Evaluate the model
log_reg_accuracy = accuracy_score(Y_test, log_reg_predictions)
print("Logistic Regression Accuracy:", log_reg_accuracy)
print(classification_report(Y_test, log_reg_predictions))
```

```
Logistic Regression Accuracy: 0.9914390542193233
              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
```

In [13]:
```
et_classifier.fit(X_train_scaled, Y_train)
log_reg.fit(X_train_scaled, Y_train)
```

Out[13]:
```
LogisticRegression
LogisticRegression(random_state=123)
```

In [14]:
```
et_predictions = et_classifier.predict(X_test_scaled)
log_reg_predictions = log_reg.predict(X_test_scaled)
```

In [15]:
```
et_accuracy = accuracy_score(Y_test, et_predictions)
log_reg_accuracy = accuracy_score(Y_test, log_reg_predictions)
```

CROSS VALIDATION

In [16]:
```
from sklearn.model_selection import cross_val_score
```

In [17]:
```
# Settings for cross-validation
cv_folds = 10

# Extra Trees Classifier cross-validation
et_cv_scores = cross_val_score(ExtraTreesClassifier(random_state=123), X_train_scaled, Y_train, cv=cv_folds, scori
et_mean_accuracy = et_cv_scores.mean()
et_std_accuracy = et_cv_scores.std()
print(f'Extra Trees Classifier - Mean Accuracy: {et_mean_accuracy:.4f}, Standard Deviation: {et_std_accuracy:.4f}'

# Logistic Regression cross-validation
log_reg_cv_scores = cross_val_score(LogisticRegression(random_state=123), X_train_scaled, Y_train, cv=cv_folds, sc
log_reg_mean_accuracy = log_reg_cv_scores.mean()
log_reg_std_accuracy = log_reg_cv_scores.std()
print(f'Logistic Regression - Mean Accuracy: {log_reg_mean_accuracy:.4f}, Standard Deviation: {log_reg_std_accurac
```

```
Extra Trees Classifier - Mean Accuracy: 0.9948, Standard Deviation: 0.0035
Logistic Regression - Mean Accuracy: 0.9937, Standard Deviation: 0.0041
```

```
In [18]:  from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
           import matplotlib.pyplot as plt
           import numpy as np

           # Make predictions with the Extra Trees Classifier
           et_predictions = et_classifier.predict(X_test_scaled)

           # Generate the classification report
           et_classification_report = classification_report(Y_test, et_predictions)
           print("Extra Trees Classifier Classification Report:\n", et_classification_report)
```

```
Extra Trees Classifier Classification Report:
               precision    recall  f1-score   support

           0       1.00      0.99      1.00      1356
           1       0.99      1.00      0.99      1097

    accuracy                           1.00      2453
   macro avg       0.99      1.00      1.00      2453
weighted avg       1.00      1.00      1.00      2453
```

CONFUSION MATRIX AND ROC CURVE FOR EXTRA TREES CLASSIFIER

```
In [19]:  # Calculate confusion matrix
           et_conf_matrix = confusion_matrix(Y_test, et_predictions)

           # Plot confusion matrix
           plt.figure(figsize=(8, 6))
           sns.heatmap(et_conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Predicted 0', 'Predicted 1'], ytickla
           plt.title('Confusion Matrix for Extra Trees Classifier')
           plt.ylabel('True Label')
           plt.xlabel('Predicted Label')
           plt.show()
```

```
In [20]:  # Calculate the probabilities of predictions
           et_probabilities = et_classifier.predict_proba(X_test_scaled)
           et_prob_positive = et_probabilities[:, 1]

           # Calculate ROC curve and AUC
           fpr, tpr, _ = roc_curve(Y_test, et_prob_positive)
           roc_auc = auc(fpr, tpr)

           # Plot ROC curve
           plt.figure(figsize=(8, 6))
           plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
           plt.plot([0, 1], [0, 1], color='navy', 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('Receiver Operating Characteristic - Extra Trees Classifier')
           plt.legend(loc="lower right")
           plt.show()
```

CONFUSION MATRIX AND ROC CURVE FOR LOGISTIC REGRESSION

In [22]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate the confusion matrix
log_reg_conf_matrix = confusion_matrix(Y_test, log_reg_predictions)

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(log_reg_conf_matrix, annot=True, fmt="d", cmap='Blues', xticklabels=['Predicted 0', 'Predicted 1'], yt
plt.title('Confusion Matrix for Logistic Regression')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()
```

TOP 20 FEATURE IMPORTANCE

In [29]:

```
# Define N, the number of top features you want to display
N = 20

# Create a subset of the DataFrame for top N features
top_features = feature_importance_df.head(N)

plt.figure(figsize=(10, 12))  # Adjust size appropriately for N features

# Creating a barplot to visualize the feature importances of top N features
sns.barplot(x='Importance', y='Feature', data=top_features)

plt.title('Top {} Feature Importances'.format(N), fontsize=20)
plt.xlabel('Importance', fontsize=16)
plt.ylabel('Features', fontsize=16)
plt.show()
```