Mid-progress Report

Predicting Bankruptcy

Prepared by:

- Nourhan Deif 202201959
- Rahma Mourad 202201407
- Radwa Badran 202200875
- Rihana Nasr 202201092

GitHub Link: Link

Preprocessing steps

1. Data Loading and Initial Exploration

- Dataset was loaded using `pandas.read_csv`.
- Displayed the **first 10 records** to understand column names and data types.
- Used `data.describe()` to check basic **statistics** (mean, std, min, max).
- `data.info()` was used to check **data types** and **null values**.
- Checked the **number of missing values** in each column using `data.isnull().sum()`

2. Data Preprocessing

- **Handled missing values**: Used the **mean** to fill numeric columns.
- Checked for **duplicate records** using `data.duplicated()`.
- Applied **Label Encoding** for all categorical (string) columns using `LabelEncoder`.
- Performed **Feature Scaling** on all numeric features (except target) using `StandardScaler`

3. Outlier Handling

- **Boxplots** were created for each numeric feature to visually detect outliers
- **IQR technique** was used to clip values outside the lower and upper bounds

EDA Techniques:

Univariate Analysis: Statistical Methods

- mean(): Calculates the average (mean) value for each column.
- median(): Computes the middle value (median) for each column when the data is sorted.
- std(): Measures the spread or dispersion of the data by calculating the standard deviation for each column.
- List item
- var(): Computes the variance, which measures the degree of spread in the data.
- skew(): Calculates the skewness to measure the asymmetry of the distribution of values in each column.
- Kurtosis (): Measures the kurtosis to evaluate the "tailedness" or sharpness of the distribution of values in each column.
- corr() between specific columns: Calculates the correlation coefficient between two specified columns to examine their linear relationship.

Interpretation:

- A low standard deviation means the values are close to the mean.
- A high standard deviation means the values are spread out more.
- A low variance means the data points are closer to the mean, while a high variance means the data points are spread out more.
- Skewness near 0 ("Bankrupt?" and "Net Income Flag") means the data is symmetrical, or close to a normal distribution.
- Negative skew means the data has a longer tail on the left.
- Positive skew means the data has a longer tail on the right.

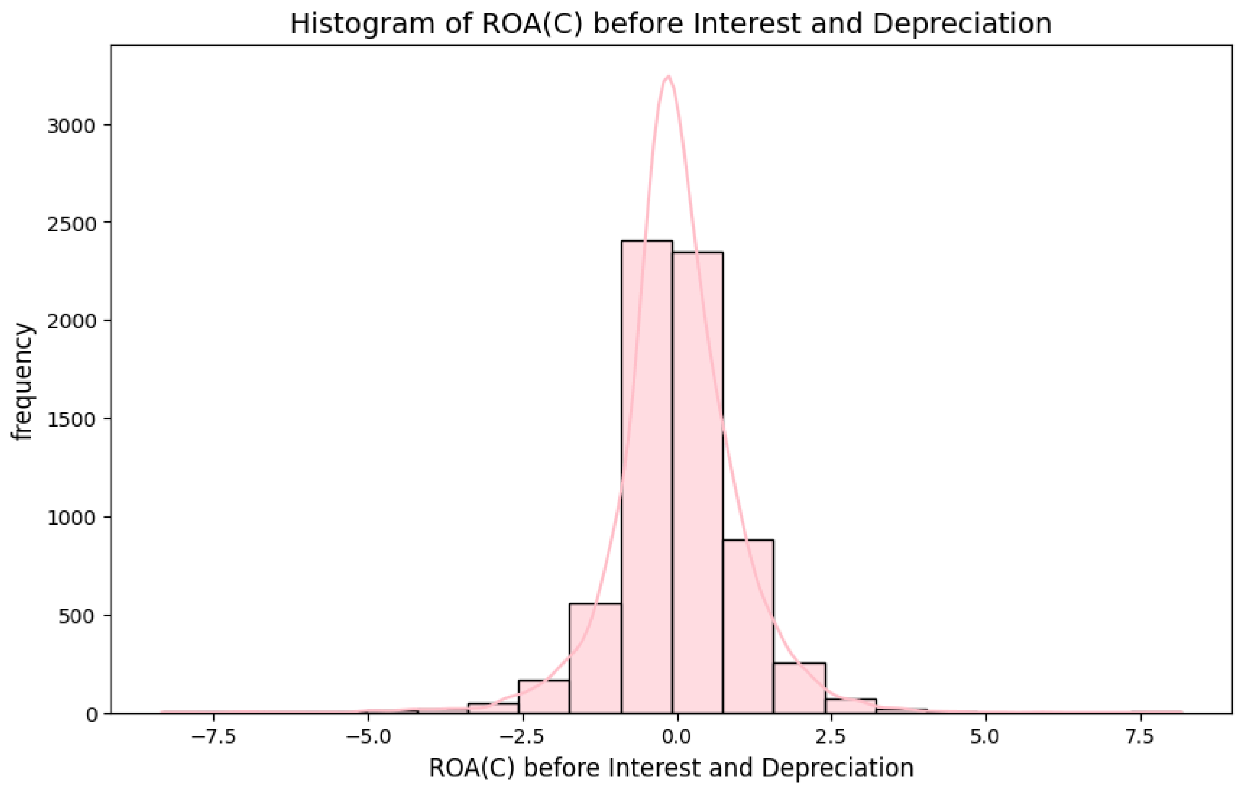
- Most of the Kurtosis values are near 0, which means that the data for these variables is normal.
- Some variables, like Liability to Equity and ROA(B), have positive Kurtosis, meaning their data has more extreme values or outliers than normal data.
- Variables like DFL and Interest Coverage Ratio have negative Kurtosis, which means their data is flatter and has fewer extreme values than normal data.
- Strong correlations between values like 0.92 between ROA(A) and ROA(C).
- A correlation of 0.53 between Operating Gross Margin and ROA(C) indicates that these two variables have a positive relationship.
- There is a positive correlation (0.53) between ROA(C) and Operating Gross Margin, meaning that companies with higher operating efficiency tend to have better returns on assets.

Graphical Methods

Univariate Graphical Methods

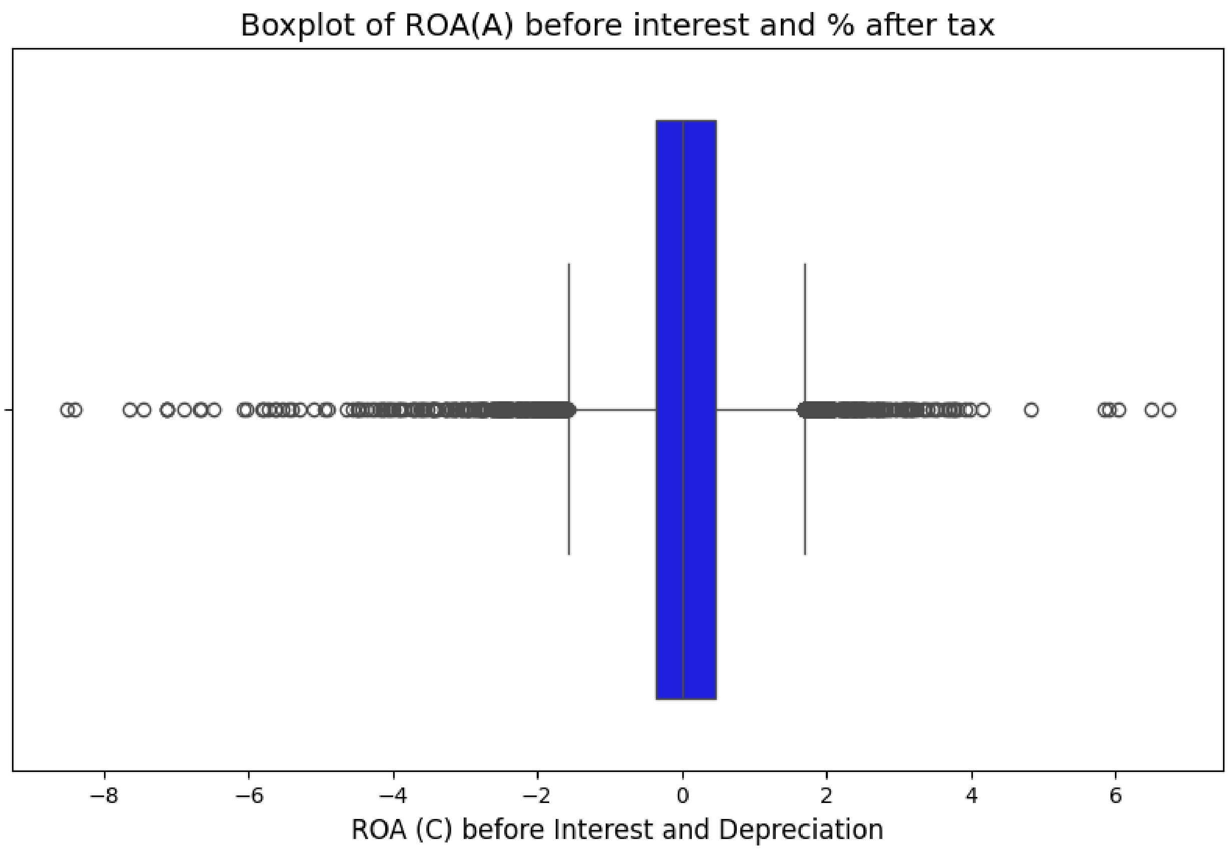
1. **Histogram Interpretation**

   A histogram shows how data is distributed across different ranges. It displays the frequency of data points in intervals, helping us see patterns, trends, and outliers easily. The x-axis shows the data values, and the y-axis shows how often those values occur.



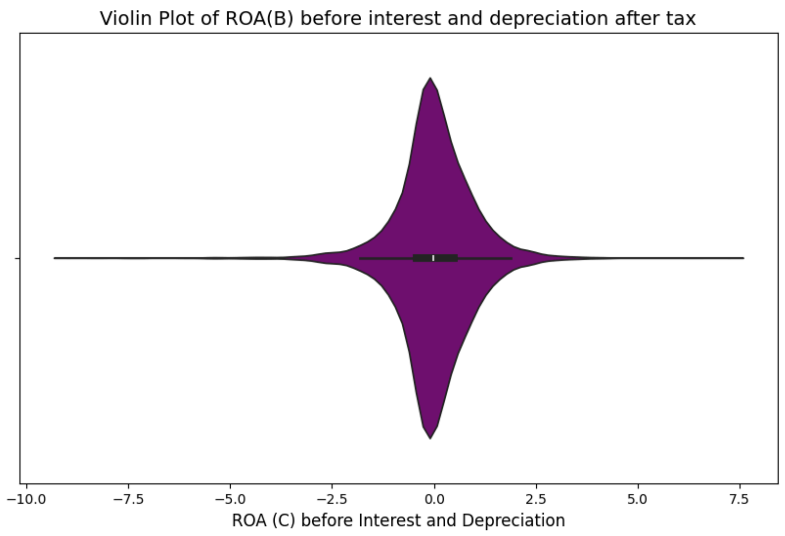
2. **Boxplot Interpretation**

   A boxplot shows the distribution of data, highlighting the median, quartiles, and any outliers. It uses a box to represent the interquartile range, showing the range of the data. Points outside the box are considered outliers.



3. **Violin Plot Interpretation**

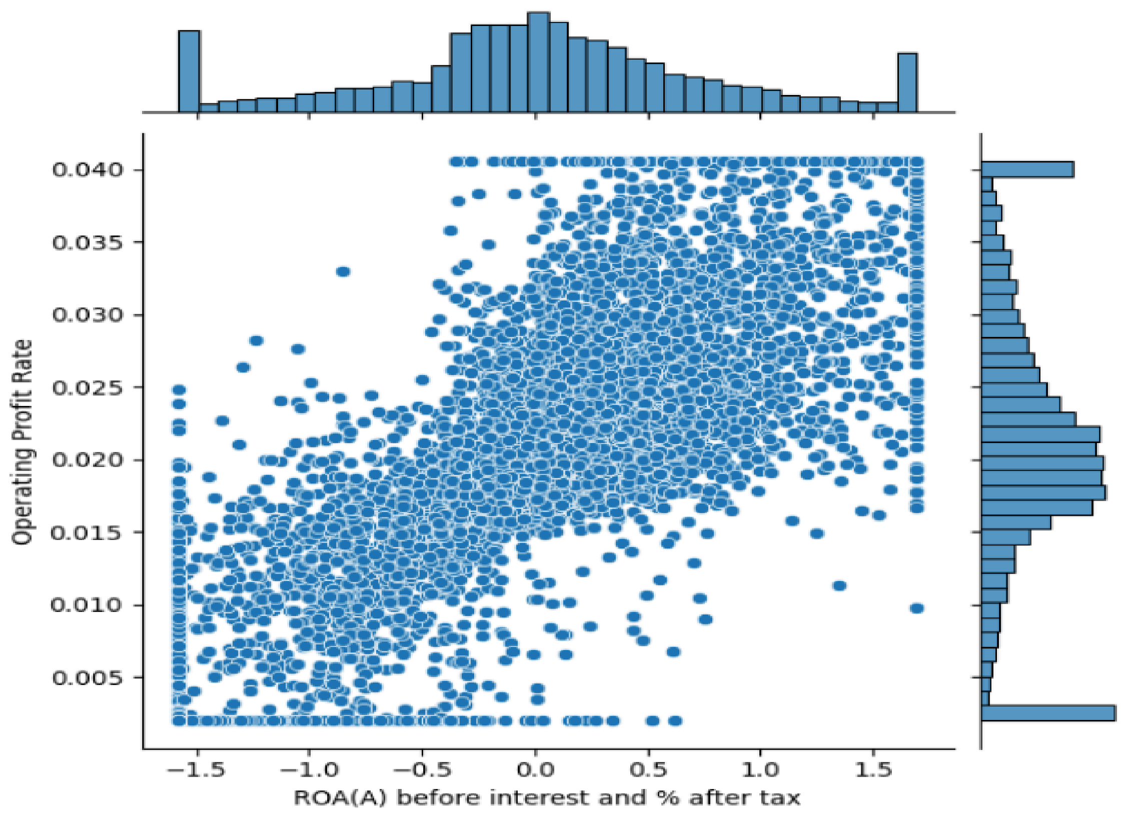
   A violin plot combines a boxplot and a density plot, showing the distribution and density of data at different values. The wider sections of the "violin" represent areas with more data, while the line in the center indicates the median.



Multivariate Graphical Methods

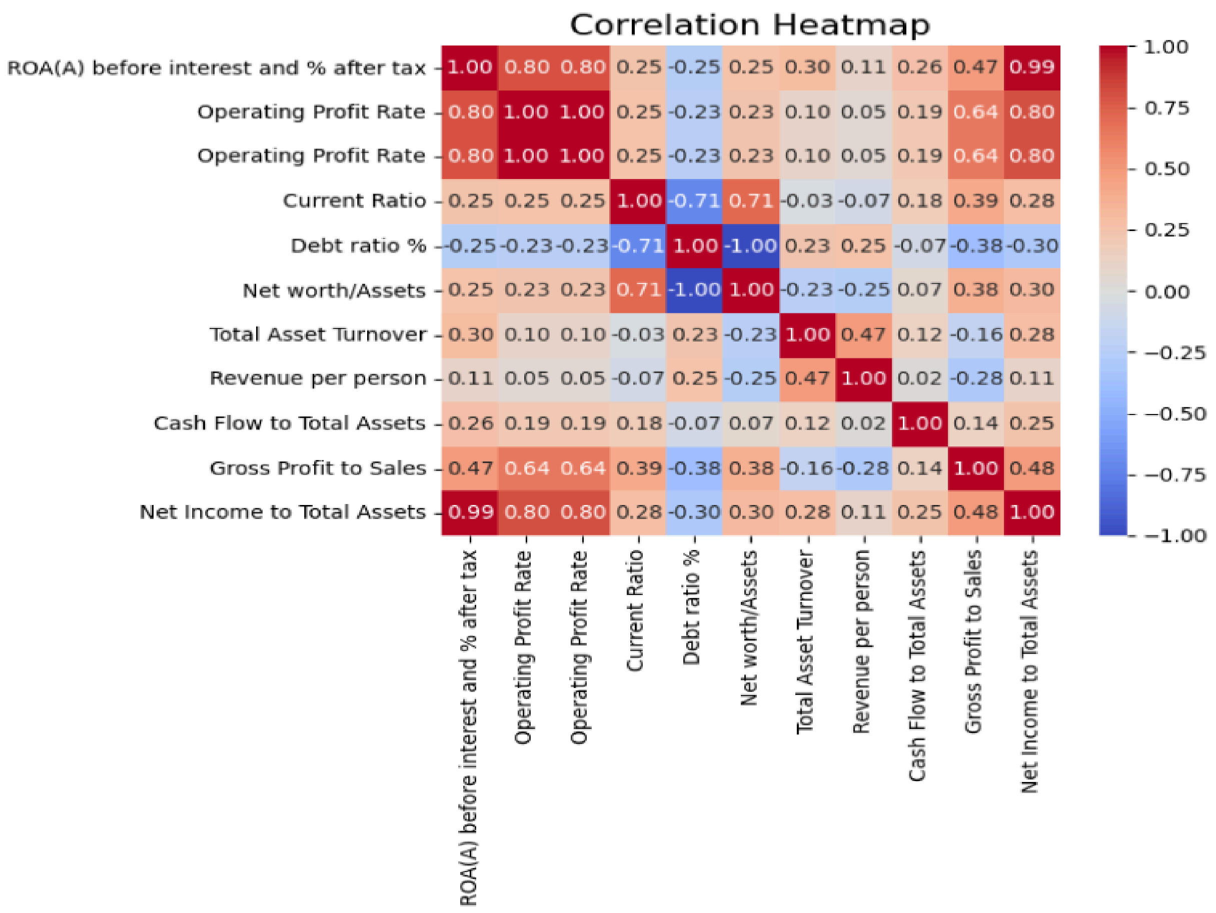
1- Joint Plot

- It shows the relationship between two variables with a scatter plot in the center and histograms or KDE plots on the sides.
- The scatter plot shows how two variables are related. If the points form a line or curve, it suggests a strong relationship. The histograms on the sides show the distribution of each variable individually.
- The choosed columns are explain how well a company turns its operations into profit, so we can expect a potential linear or non-linear relationship.



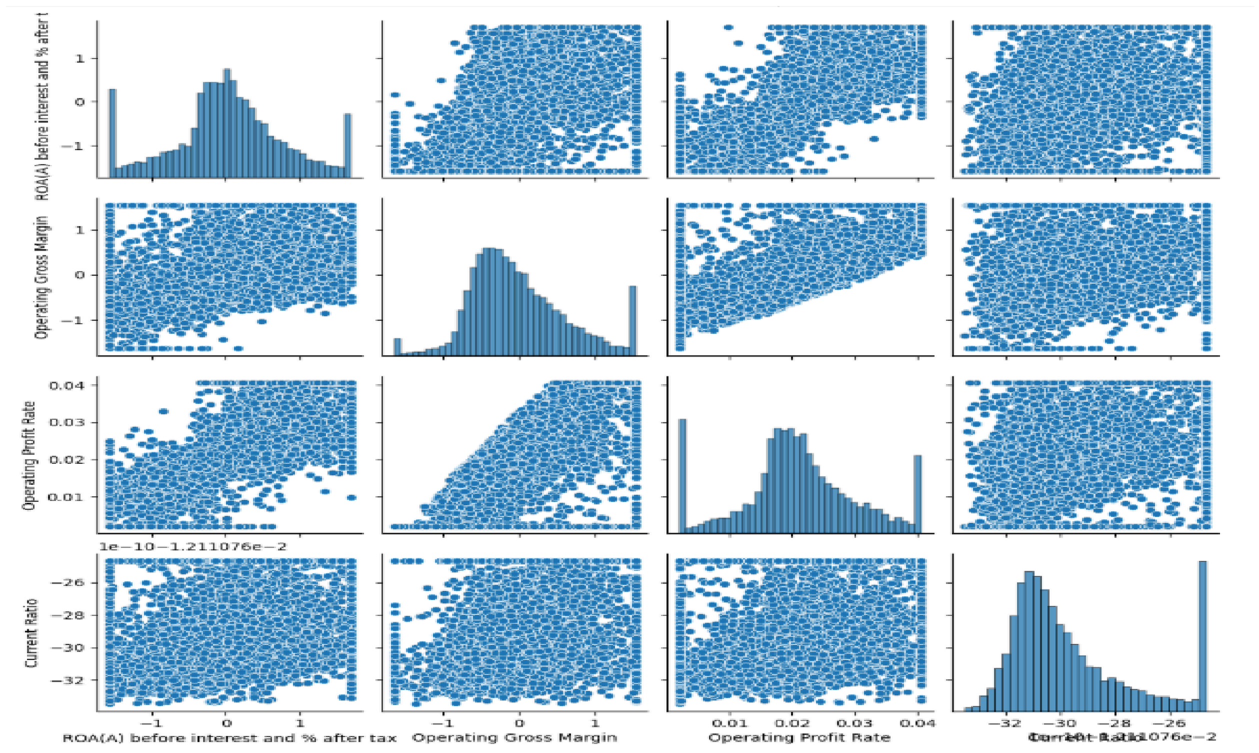
2- Heatmap

- A heatmap displays a matrix of values with colors, making it easy to see patterns. It'used to show correlations between variables.
- High values (close to 1 or -1) indicating a strong positive or negative relationship.
- Low values (close to 0) showing weak or no correlation.



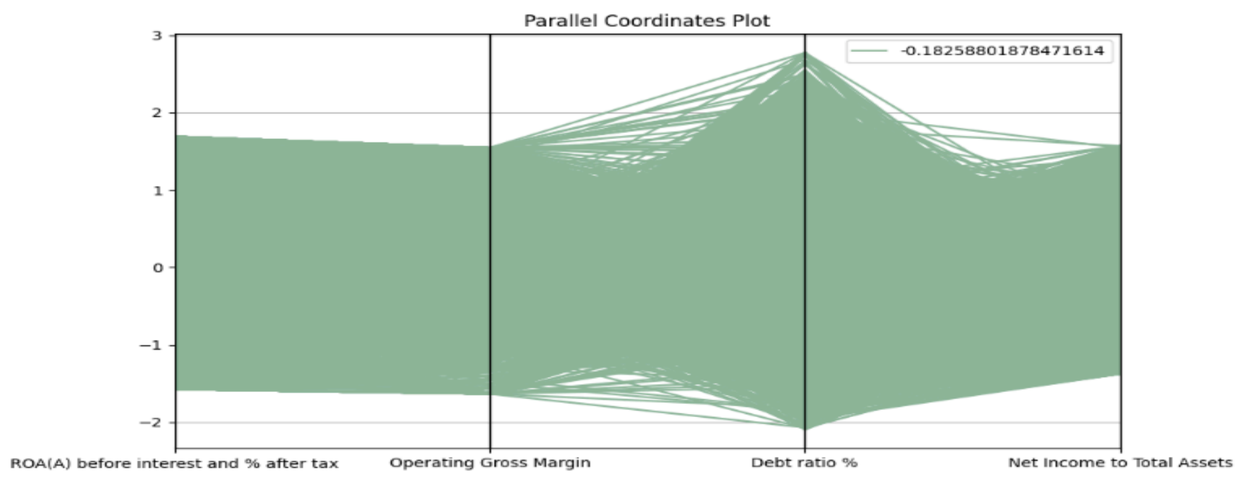
3- Pair Plot

- The scatter plots show the relationships between different variable pairs.
- The diagonal histograms help we see the distribution of each variable.



4- Parallel Coordinates Plot

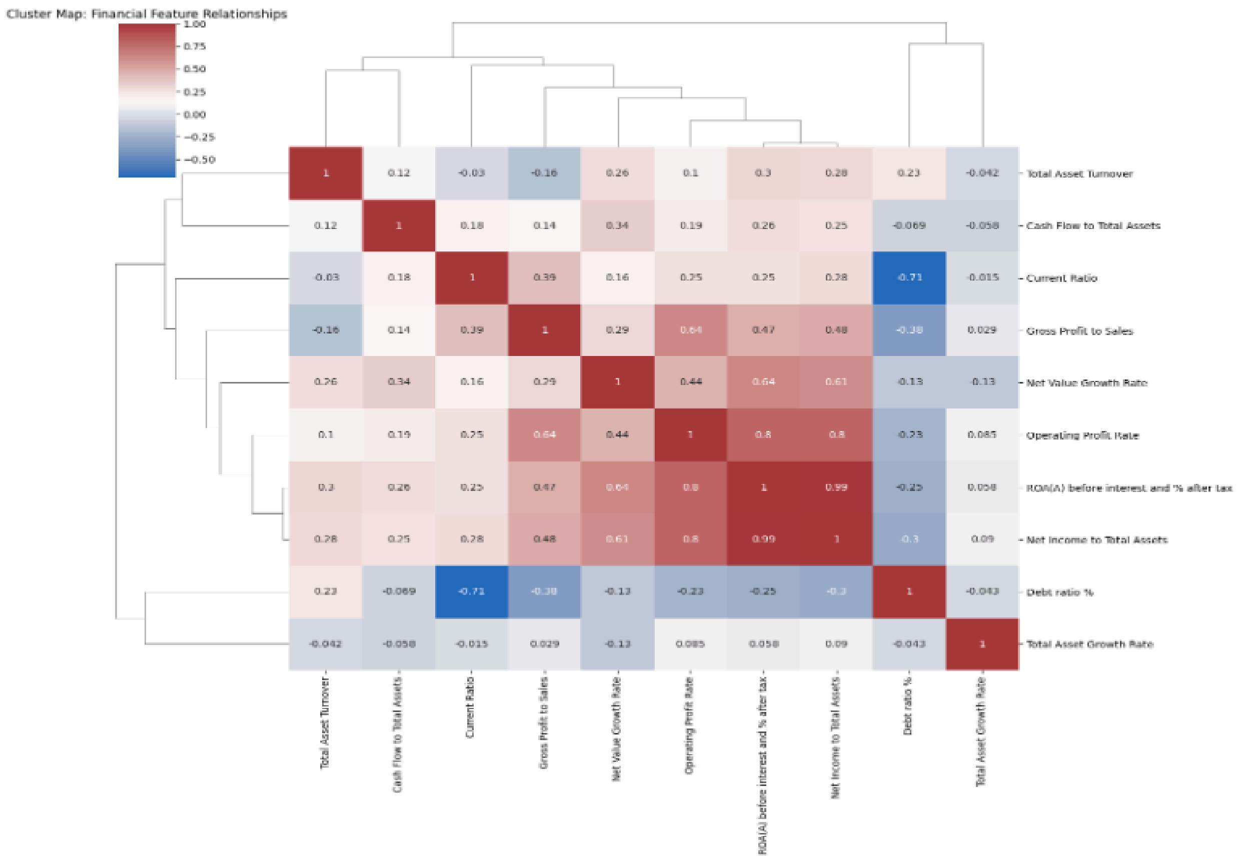
- It displays multiple variables in parallel axes. Each line represents an observation, and we can see how the values change across different variables.
- close lines to each other show that the variables have a relationship.
- Lines that spread apart show that the variables are behaving differently.



5-Cluster Map

- It displays a heatmap with dendrograms added to the side, showing how features or observations group based on similarity.

- Each cell shows the correlation or other distance metric between a pair of variables.
- Hierarchical clustering reorders the rows/columns so that similar features are grouped together visually.



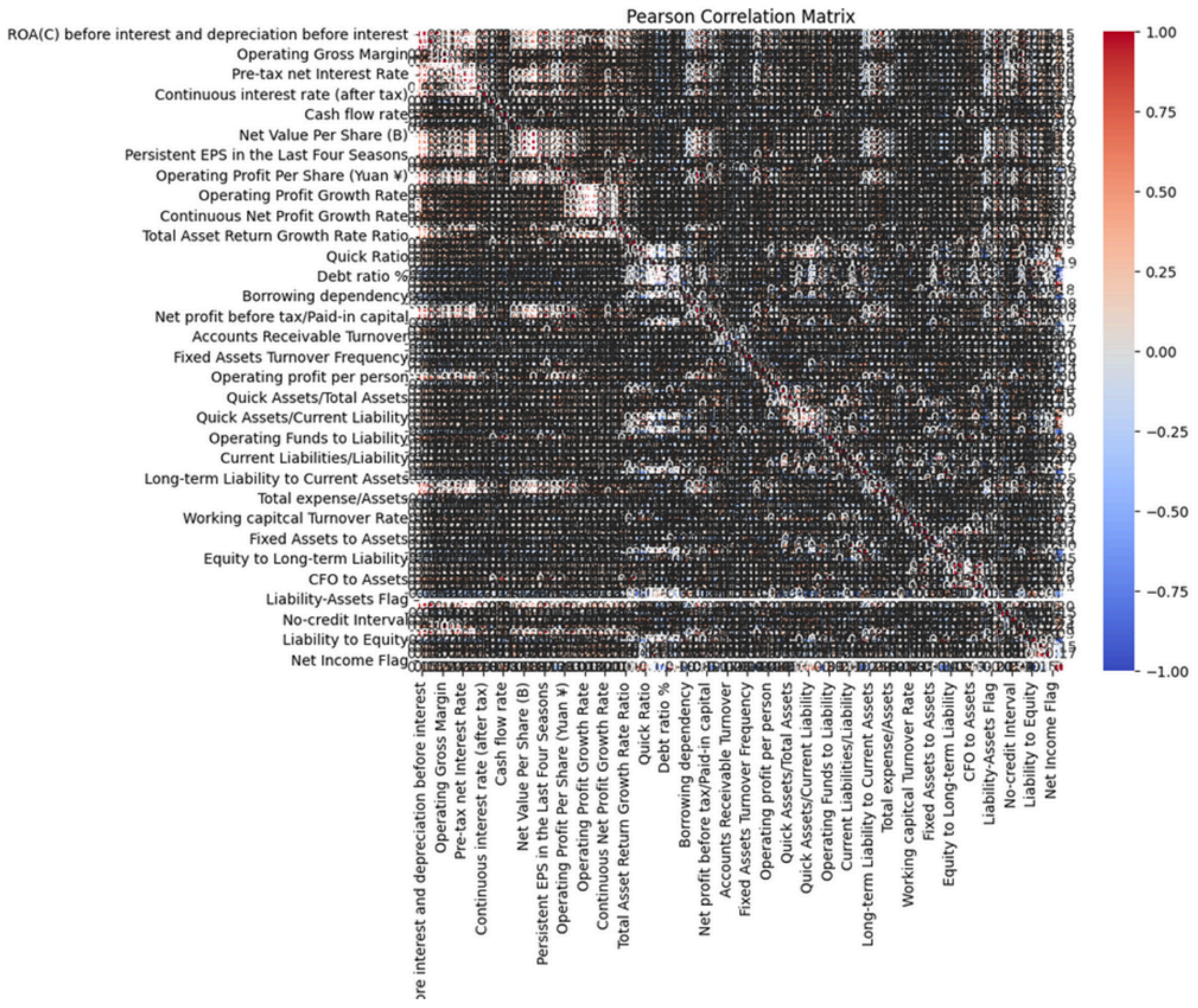
Multivariate Correlation Analysis for Explainability

Overview

Understanding the relationships between multiple variables is essential for building explainable and robust machine learning models. In this section, we analyze the correlations between features using a variety of statistical methods to identify dependencies, potential multicollinearity, and informative predictors. These insights guide feature selection and deepen interpretability.

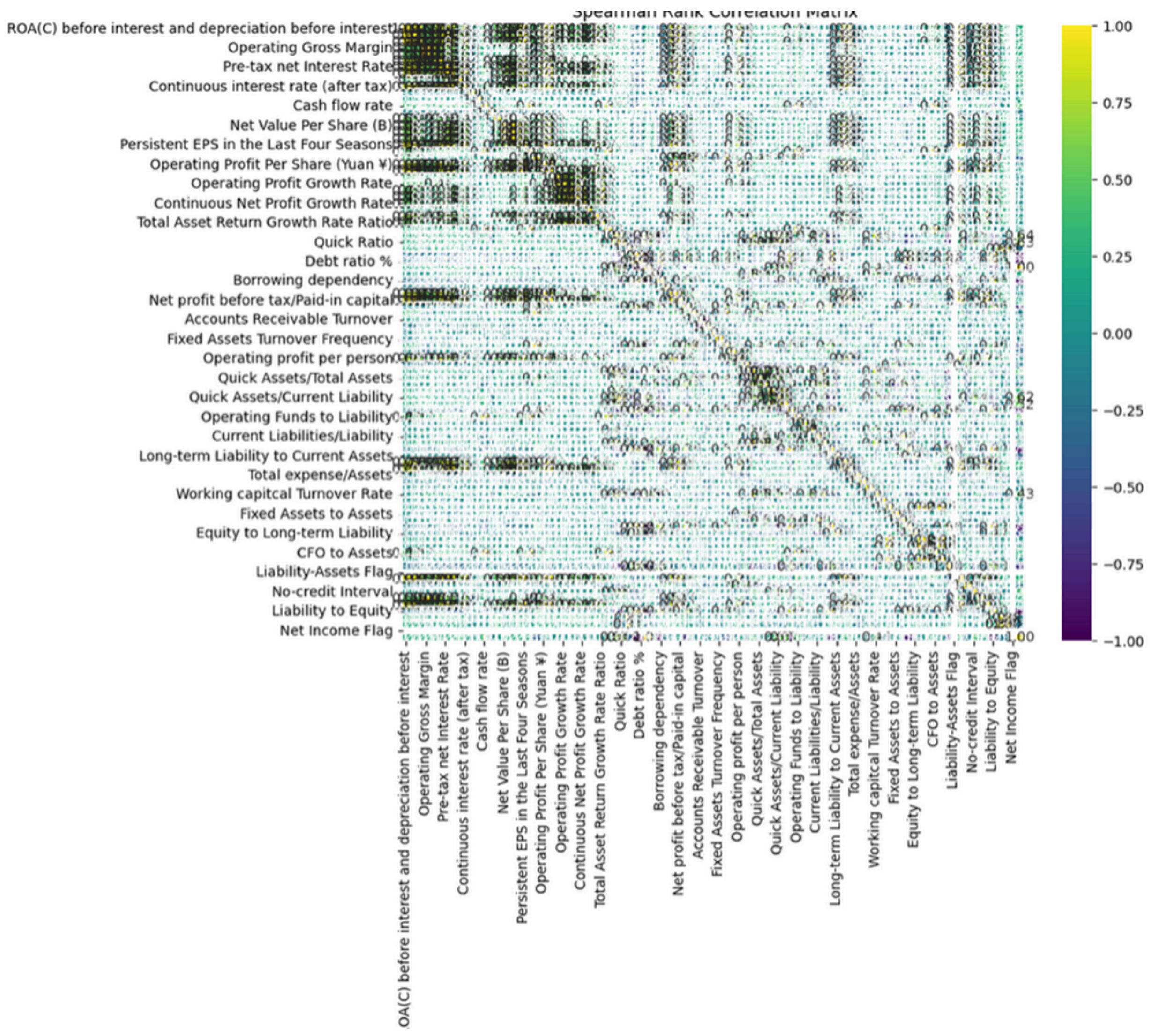
1. Pearson Correlation Coefficient

Pearson's correlation measures the linear relationship between continuous numerical variables. Coefficients close to +1 or –1 suggest strong linear associations, whereas values near 0 indicate no linear dependency.



2. Spearman Rank Correlation

Spearman’s correlation assesses monotonic relationships by comparing ranked variables. It is robust to outliers and effective for skewed or non-normally distributed data.



3. Correlation Ratio (η^2) — Categorical to Numeric

The correlation ratio quantifies how well a categorical variable explains the variance in a numeric variable. This metric is useful for detecting non-linear and non-monotonic effects, especially in classification tasks.

Table 1: Sample η^2 Values for Categorical-Numeric Dependencies

... *all features showed minimal explanatory power* …

Interpretation: The η^2 results indicate that none of the individual categorical vari- ables substantially explain variance in any numeric variables in the dataset, implying that numerical predictors may operate independently of categorical groupings or that effects are non-linear and subtle.

4. **Uncertainty Coefficient (Theil's U) — Categorical to Categorical**

Theil's U is an asymmetric measure that quantifies how much knowledge of one categorical variable reduces the uncertainty of another. It is especially helpful for understanding the redundancy or dependency between categorical features.

Table 1: Sample η^2 Values for Categorical-Numeric Dependencies

Categorical Variable	**Numeric Variable**	η^2 Score
Bankrupt?	ROA(C) before interest	0.000
Bankrupt?	Operating Gross Margin	0.000
Bankrupt?	Net Value Per Share (A)	0.000

... All features showed minimal explanatory power ...

(Only one categorical column present)

Conclusion: Since the dataset contains only one categorical variable, no inter-variable uncertainty could be assessed, but this confirms full certainty when predicting a variable from itself (U = 1.0).

Summary

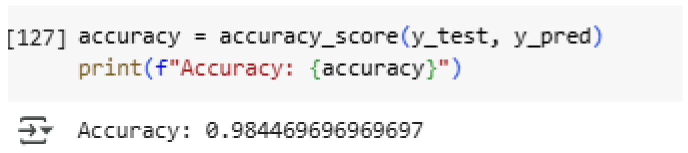
Through correlation and dependency analysis:

- Linear and monotonic relationships between numerical features were visualized via Pearson and Spearman correlations.
- Categorical-to-numeric variance contributions were found to be negligible.
- Categorical inter-dependencies were limited due to the dataset composition.

These insights support feature selection, model explainability, and diagnostic interpretations within the broader XGBoost modeling pipeline.

Model 1: Random Forest Classifier

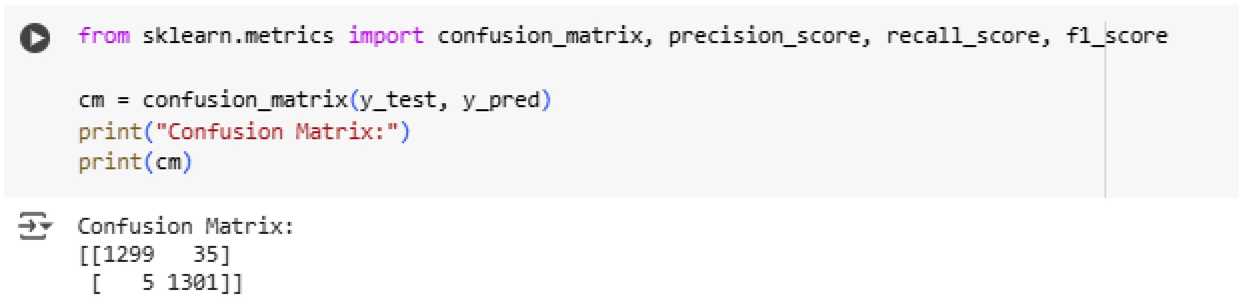
- Used Random Forest Classifier with balanced class weights.
- Trained the model on 80% of the data, tested on 20%
- Made predictions on the test set



```
[127] accuracy = accuracy_score(y_test, y_pred)
      print(f"Accuracy: {accuracy}")

      Accuracy: 0.9844696969696969
```

Accuracy: 98.4% → The model made correct predictions in almost all cases – excellent performance.

Evaluate the model by the confusion matrix



```
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)

Confusion Matrix:
[[1299   35]
 [   5 1301]]
```

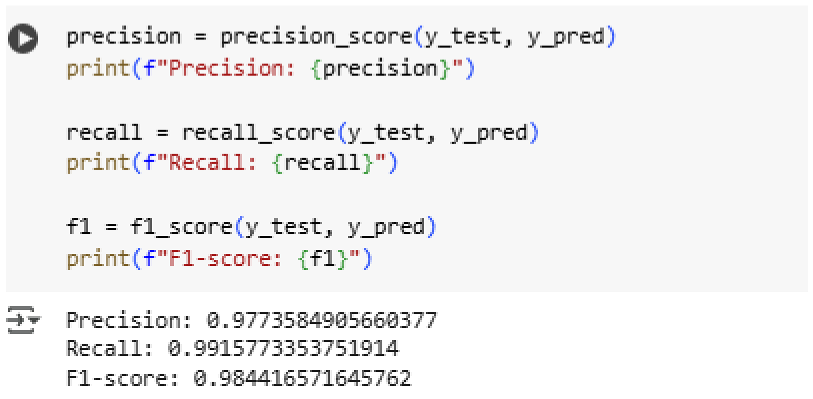
True Negatives (TN) = 1299 The model correctly predicted 1304 instances as non-bankrupt.

False Positives (FP) = 35 The model incorrectly predicted 30 companies as bankrupt when they were actually not.

False Negatives (FN) = 5 The model failed to identify 11 bankrupt companies, labeling them as non-bankrupt.

True Positives (TP) = 1301 The model correctly identified 1295 companies as bankrupt.

Evaluate model by using Precision , Recall , F1_score



```
precision = precision_score(y_test, y_pred)
print(f"Precision: {precision}")

recall = recall_score(y_test, y_pred)
print(f"Recall: {recall}")

f1 = f1_score(y_test, y_pred)
print(f"F1-score: {f1}")
```

```
Precision: 0.9773584905660377
Recall: 0.9915773353751914
F1-score: 0.984416571645762
```

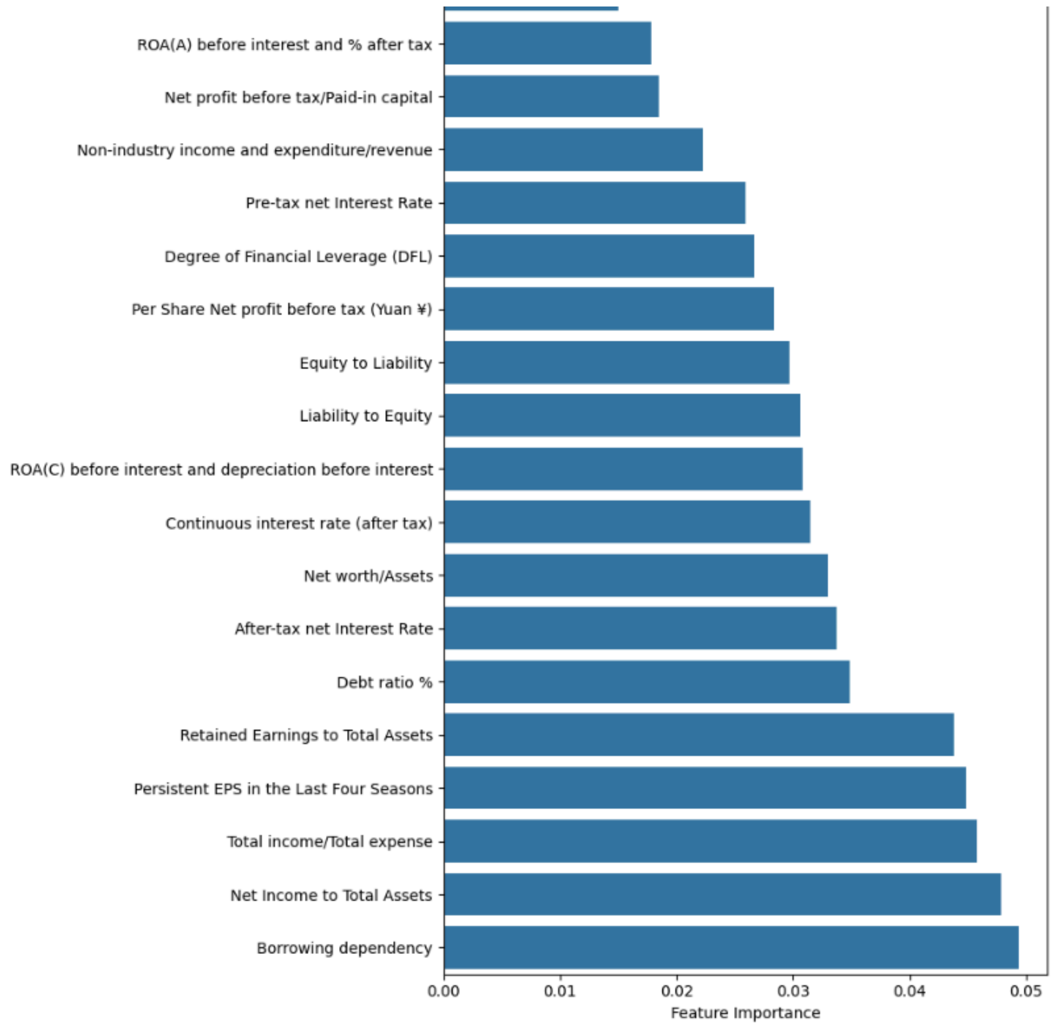
Precision: 97.7% → Most of the predicted bankruptcies were correct, meaning few false positives

Recall: 99.2% → The model detected nearly all actual bankrupt companies – very reliable

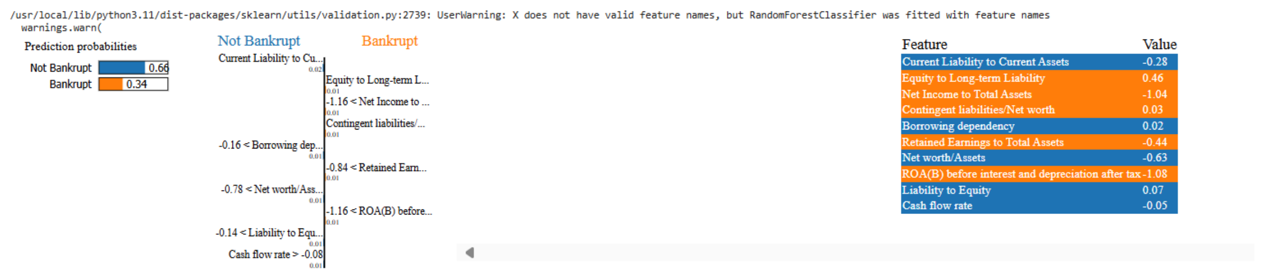
F1-Score: 98.4% → Strong balance between precision and recall, showing overall consistency

Model Interpretation

- Plotted the **feature importances** using bar charts



- Used **LIME (Local Interpretable Model-agnostic Explanations)** to explain model predictions on individual instances.

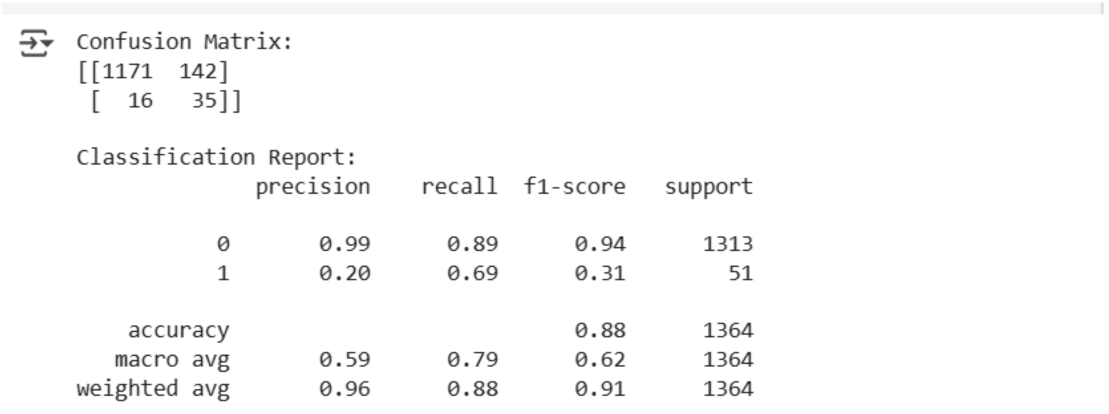


```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

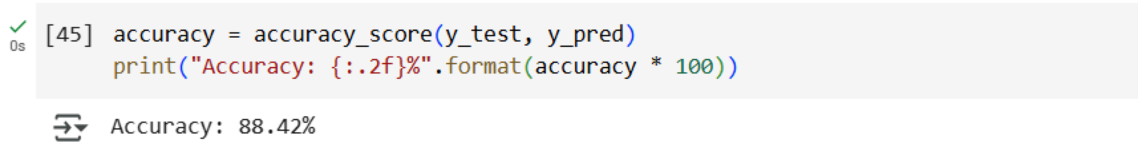
Feature	Value
Current Liability to Current Assets	-0.28
Equity to Long-term Liability	0.46
Net Income to Total Assets	-1.04
Contingent liabilities/Net worth	0.03
Borrowing dependency	0.02
Retained Earnings to Total Assets	-0.44
Net worth/Assets	-0.63
ROA(B) before interest and depreciation after tax	-1.08
Liability to Equity	0.07
Cash flow rate	-0.05

Model 2: SVM model for classifiatcion:
Bankruptcy Prediction Using Machine Learning Techniques

1. First, train the SVM model after performing preprocessing, and provide results with an accuracy of approximately 88.42%.

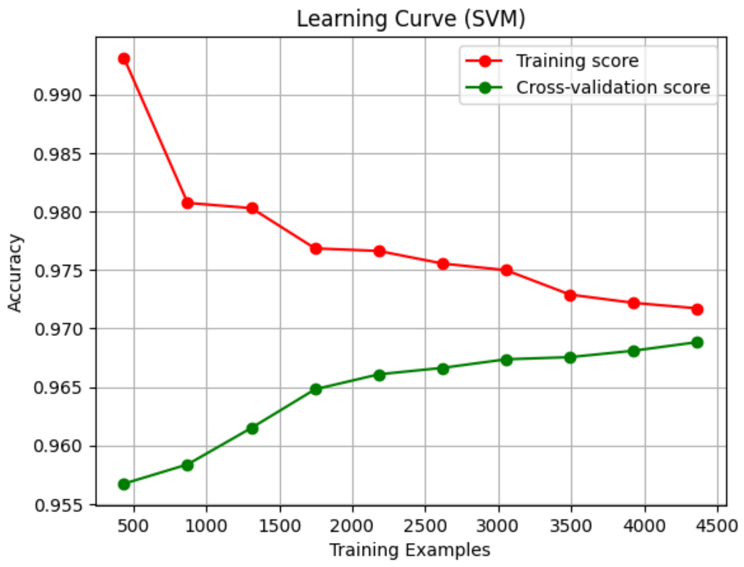


```
Confusion Matrix:
[[1171  142]
 [  16   35]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.89      0.94      1313
           1       0.20      0.69      0.31        51

    accuracy                           0.88      1364
   macro avg       0.59      0.79      0.62      1364
weighted avg       0.96      0.88      0.91      1364
```

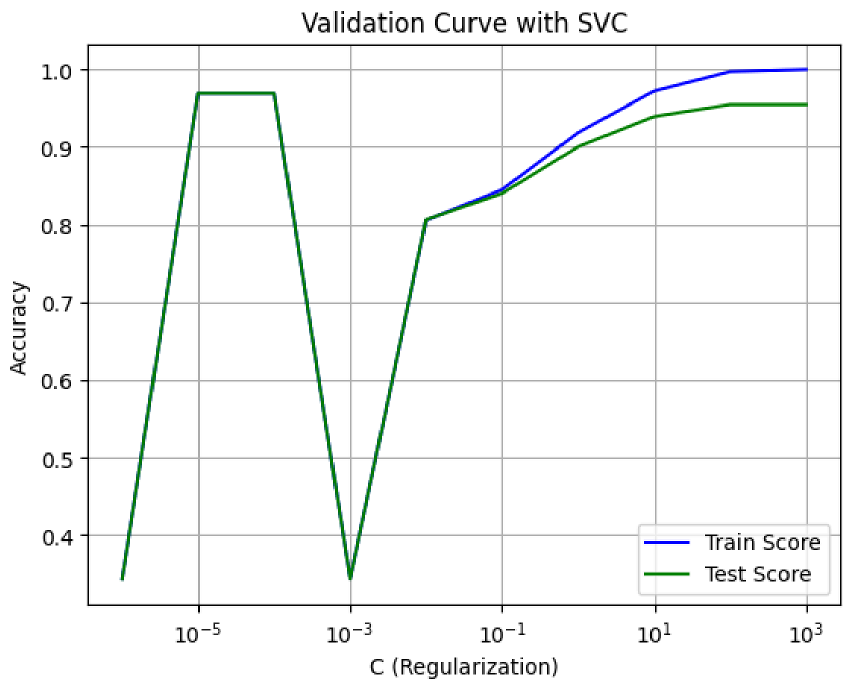


```
[45] accuracy = accuracy_score(y_test, y_pred)
     print("Accuracy: {:.2f}%".format(accuracy * 100))

Accuracy: 88.42%
```

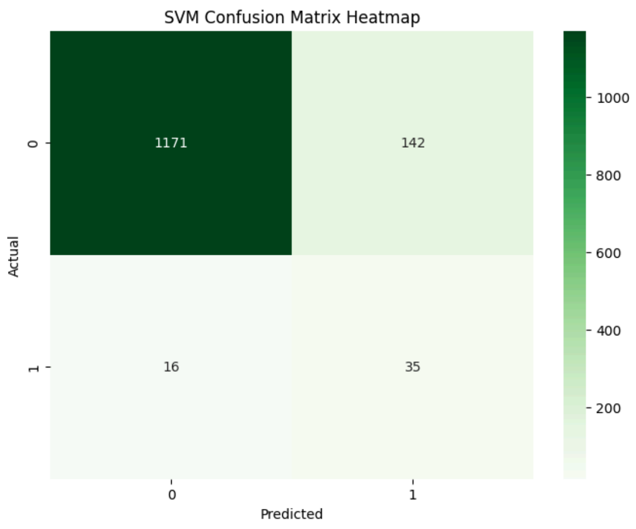
2. Plotting a learning curve, the plot showed that the model learns well without overfitting or underfitting, because the accuracy and cross validation curve were close.



3. Plotting the validation curve to know the performance of the model with the change of parameter “C”, and the test score was in some cases fixed first, it decreases, and in some cases it increases.
   Values of C: -6,3,10



4. A confusion matrix is used to compare the predicted label with the true label:
   - TP: 1171 --> Model correctly predicts the positive class
   - TN: 35 --> Model correctly predicts the negative class
   - FP: 16 --> wrongly predicts the positive class
   - FN: 142 --> wrongly predicts the negative class

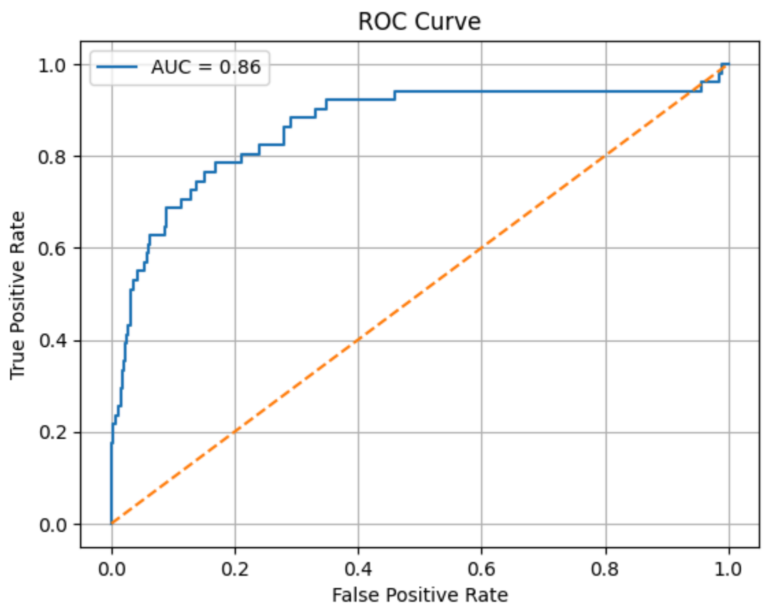


5. The ROC curve is a graph that shows how well a classification model can separate between classes.

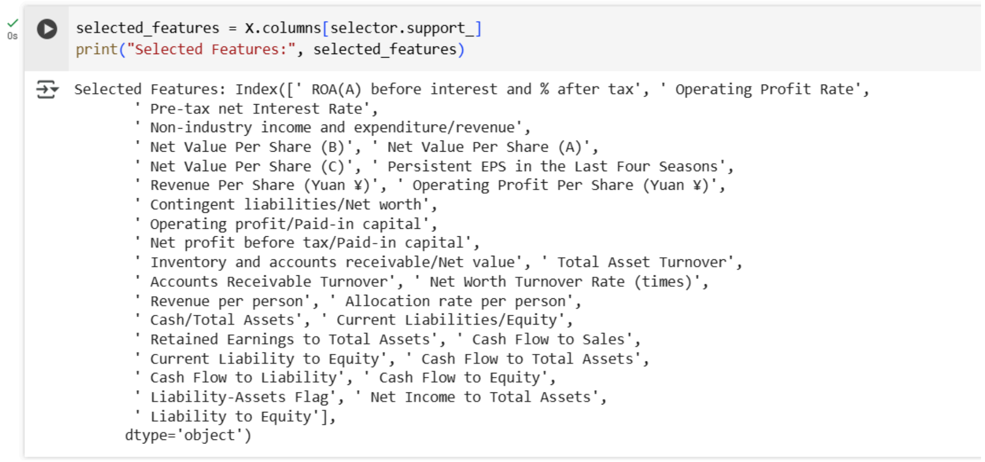
   It plots True Positive Rate (TPR) vs. False Positive Rate (FPR):

   - The Area Under the Curve (AUC) tells how good the model is:
   - AUC = 1: The model is good at telling the difference between positive and negative cases.
   - AUC = 0.5: The model isn't learning anything useful and is just guessing randomly.
   - AUC < 0.5 : The model struggles to tell the difference between the positive and negative classes.

     AUC close to 1: The model is good at telling the difference between positive and negative cases.



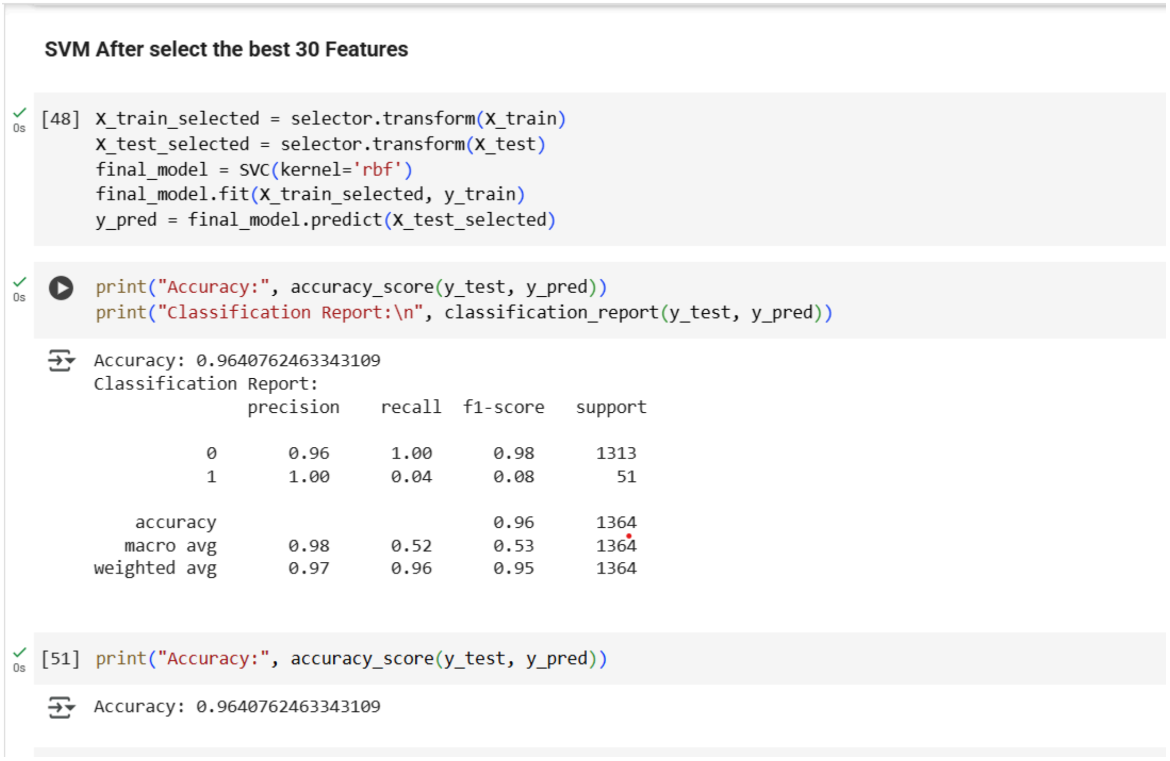
6. **RFE (Recursive Feature Elimination):**
    - RFE is a feature selection technique. It helps us choose the most important features from the dataset.
    - How it works:
        1. It starts with all features.
        2. It trains a model
        3. Then it removes the least important feature.
        4. It repeats this process recursively until the desired number of features is left.



```
selected_features = X.columns[selector.support_]
print("Selected Features:", selected_features)
```

```
Selected Features: Index([' ROA(A) before interest and % after tax', ' Operating Profit Rate',
       ' Pre-tax net Interest Rate',
       ' Non-industry income and expenditure/revenue',
       ' Net Value Per Share (B)', ' Net Value Per Share (A)',
       ' Net Value Per Share (C)', ' Persistent EPS in the Last Four Seasons',
       ' Revenue Per Share (Yuan ¥)', ' Operating Profit Per Share (Yuan ¥)',
       ' Contingent liabilities/Net worth',
       ' Operating profit/Paid-in capital',
       ' Net profit before tax/Paid-in capital',
       ' Inventory and accounts receivable/Net value', ' Total Asset Turnover',
       ' Accounts Receivable Turnover', ' Net Worth Turnover Rate (times)',
       ' Revenue per person', ' Allocation rate per person',
       ' Cash/Total Assets', ' Current Liabilities/Equity',
       ' Retained Earnings to Total Assets', ' Cash Flow to Sales',
       ' Current Liability to Equity', ' Cash Flow to Total Assets',
       ' Cash Flow to Liability', ' Cash Flow to Equity',
       ' Liability-Assets Flag', ' Net Income to Total Assets',
       ' Liability to Equity'],
      dtype='object')
```

7. After selecting the best 30 features and training on them, the model achieved a better performance of 96%.



SVM After select the best 30 Features

```
[48] X_train_selected = selector.transform(X_train)
     X_test_selected = selector.transform(X_test)
     final_model = SVC(kernel='rbf')
     final_model.fit(X_train_selected, y_train)
     y_pred = final_model.predict(X_test_selected)
```

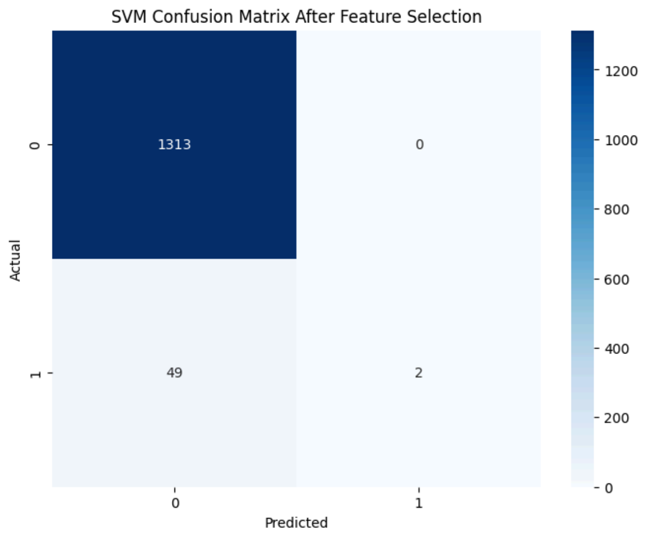
```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9640762463343109
Classification Report:
               precision    recall  f1-score   support

           0       0.96      1.00      0.98      1313
           1       1.00      0.04      0.08        51

    accuracy                           0.96      1364
   macro avg       0.98      0.52      0.53      1364
weighted avg       0.97      0.96      0.95      1364
```

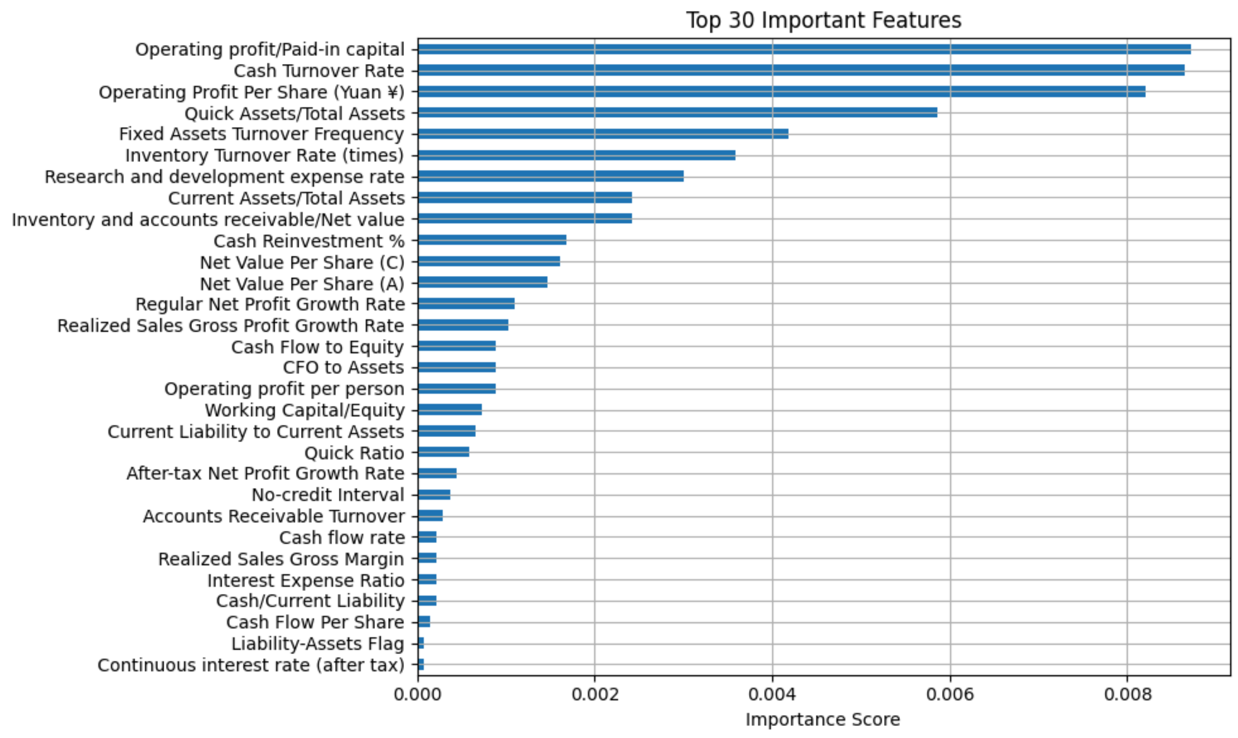
```
[51] print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9640762463343109
```

8. Confusion matrix after feature selection:



9. **Permutation importance** tells us how much each feature important in the model:
    - It works by randomly shuffling one feature at a time and checking how much the model's performance drops.
    - If the performance drops a lot, that feature is important.
    - If there's no change or improvement, that feature is not important.

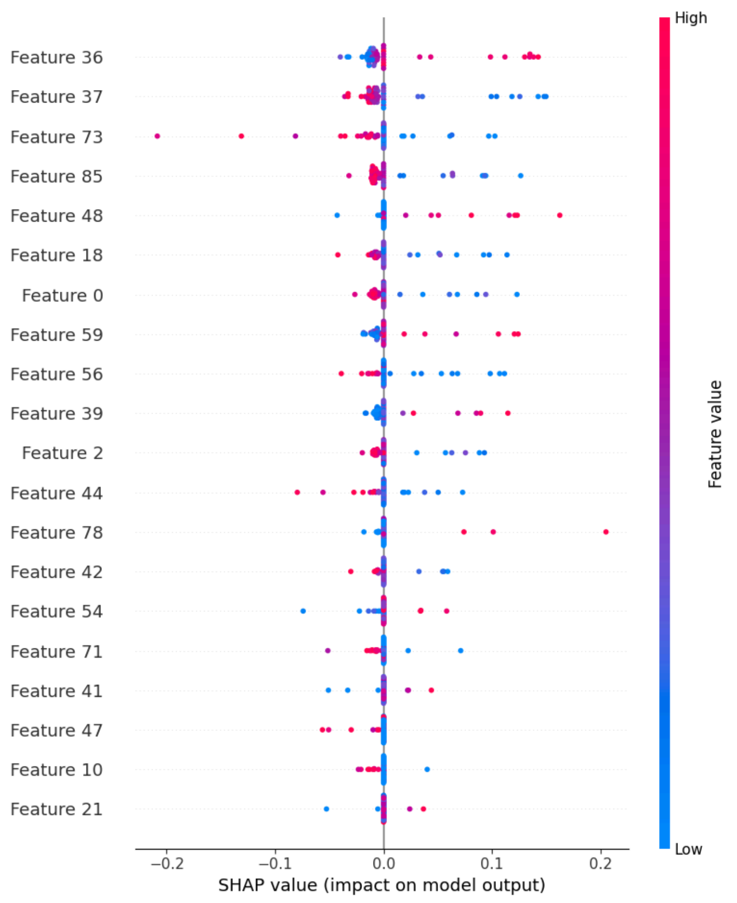


SHAP

- It works based on game theory:
- Imagine features are like players in a game working together to make a prediction. SHAP tells us how much each feature contributed

How SHAP works:

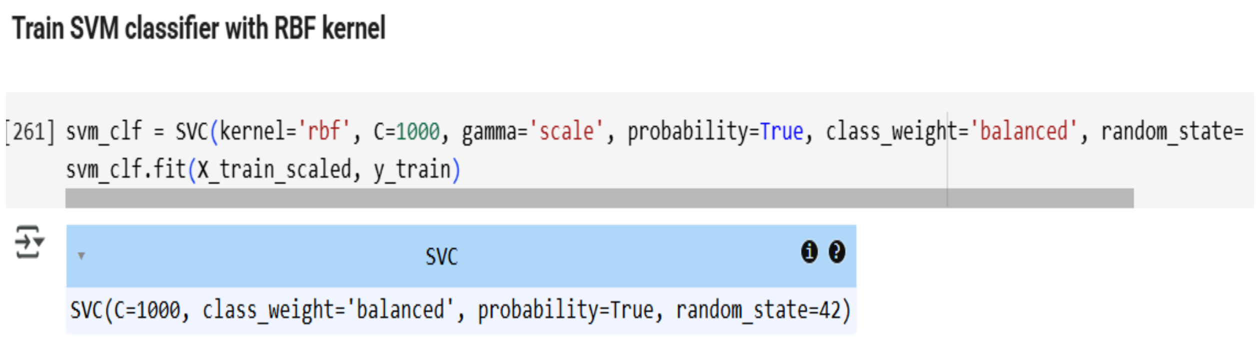
1. Treat each feature value as a player
2. Prediction is the game's final output
3. Calculate how much the outcome changes with and without each feature, and this is called the Shapley value
4. Average these changes across all combinations to get the final importance



Challenges:

Facing some difficulty in selecting the best features. I initially selected the top 10 features, but when I retrained the model, the accuracy dropped significantly due to the small number of features. Therefore, it was necessary to include a larger number of important features."

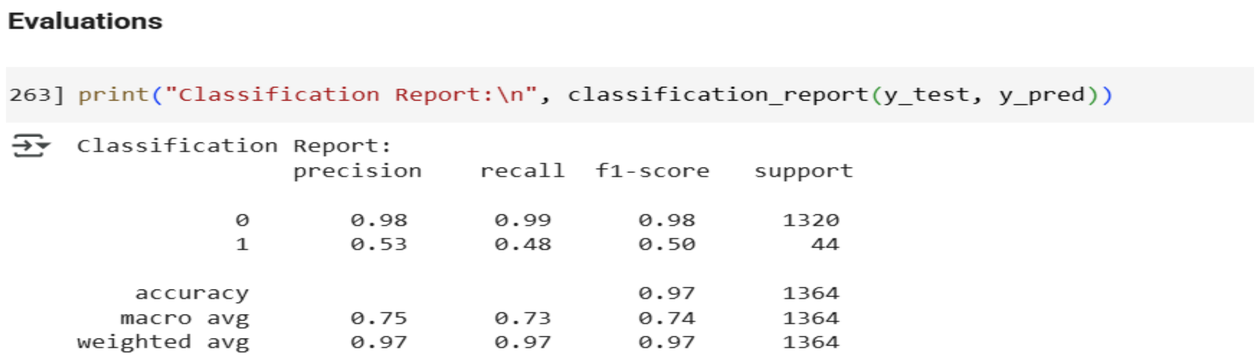
Model 3: SVM model for classification for paper:
https://www.scirp.org/journal/paperinformation?paperid=80398#t6



Train SVM classifier with RBF kernel

```
[261] svm_clf = SVC(kernel='rbf', C=1000, gamma='scale', probability=True, class_weight='balanced', random_state=
      svm_clf.fit(X_train_scaled, y_train)
```

```
SVC
SVC(C=1000, class_weight='balanced', probability=True, random_state=42)
```



Evaluations

```
263] print("Classification Report:\n", classification_report(y_test, y_pred))
```

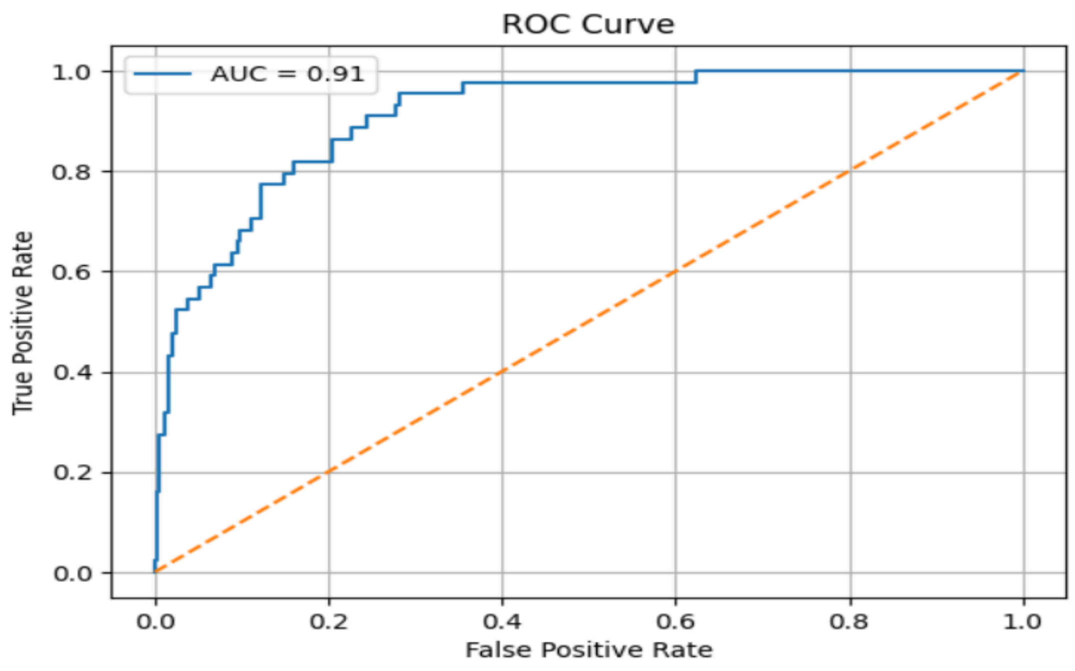
```
Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.99      0.98      1320
           1       0.53      0.48      0.50        44

    accuracy                           0.97      1364
   macro avg       0.75      0.73      0.74      1364
weighted avg       0.97      0.97      0.97      1364
```

1- The trained SVM model, after doing preprocessing, gives results with an accuracy of approximately 97%

2- The evaluations of the model give precision for both classes 0,1, scores of 0.98 and 0.53.

3-The evaluations of the model give recall for both classes 0,1, scores of 0.99 and 0.48.

4-The evaluations of the model give f1-score for both classes 0,1, scores of 0.98 and 0.50.



5- The evaluations of the model give 0.91 for the ROC AUC score.

6- Performing RFE "Recursive Feature Elimination"

It's a feature selection technique that helps to choose the most important features from your dataset.

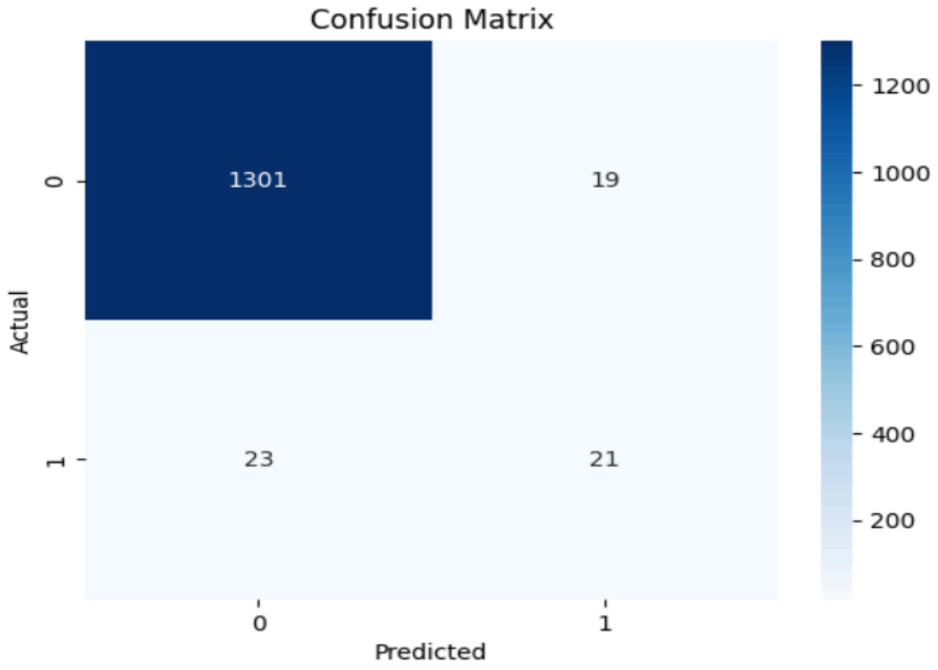
The mechanism: beginning with all your features, trains a model, then removes least important feature for it, and repeats this process recursively until the desired number of features is left.

It gives the following importance scores to the selected features

```
#Map importance scores to selected features
importance_scores = pd.Series(results.importances_mean, index=selected_features)
print("\n Permutation Importance (mean decrease in F1):")
print(importance_scores.sort_values(ascending=False))
```

```
 Permutation Importance (mean decrease in F1):
Net Income to Total Assets                                 0.347082
ROA(C) before interest and depreciation before interest    0.324569
Net Value Per Share (B)                                    0.320366
Persistent EPS in the Last Four Seasons                    0.274655
ROA(A) before interest and % after tax                     0.268201
After-tax Net Profit Growth Rate                           0.267647
ROA(B) before interest and depreciation after tax          0.246533
Regular Net Profit Growth Rate                             0.174684
Total debt/Total net worth                                 0.141290
Net Value Per Share (C)                                    0.140326
dtype: float64
```

7- The visualization of the confusion matrix of the model



XGBoost Model Development

Data Exploration

We started by exploring the structure and contents of the dataset. The initial steps involved loading the data and printing basic information such as column names, data types, and identifying any missing values. This provided an overview of the dataset's health and helped us determine the appropriate cleaning and preprocessing steps needed later.

Data Cleaning

In this phase, we handled missing values by either filling them with statistical replacements or dropping incomplete records. We also ensured categorical variables were correctly encoded and all data types were consistent with model requirements. This step is essential to avoid biases and runtime errors during model training.

Feature Engineering

We created new variables or transformed existing ones to improve model performance. For instance, some continuous variables were normalized, and categorical values were one-hot encoded. Feature engineering helps the model learn more effectively by presenting relevant signals clearly.

Model Training and Evaluation

We trained an XGBoost classifier on the preprocessed dataset. We used default hyperparameters initially and evaluated the model using accuracy, precision, recall, and F1-score. The confusion matrix also gave insights into true positives and false negatives, which are important for assessing real-world impact.

We then visualized feature importance to understand which variables most influenced predictions. This step revealed meaningful relationships between specific features and the target variable.

Next, we used cross-validation to assess the model's generalization ability. This confirmed the stability of performance across different subsets of the data.

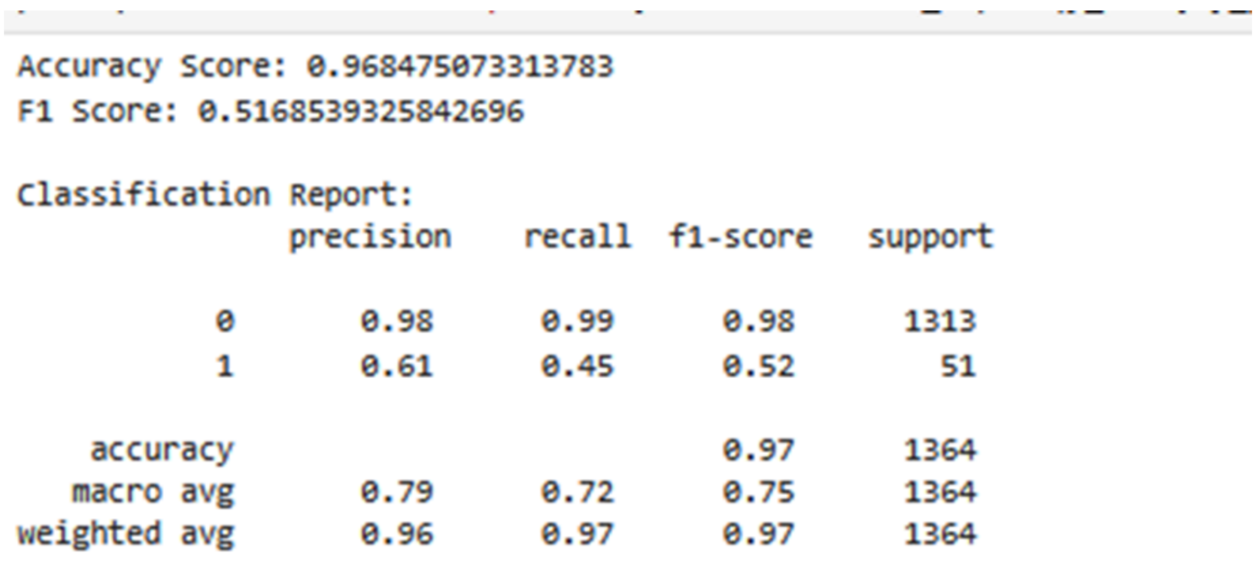

```
Accuracy Score: 0.968475073313783
F1 Score: 0.5168539325842696

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      1313
           1       0.61      0.45      0.52        51

    accuracy                           0.97      1364
   macro avg       0.79      0.72      0.75      1364
weighted avg       0.96      0.97      0.97      1364
```

Figure 1: Classification Report

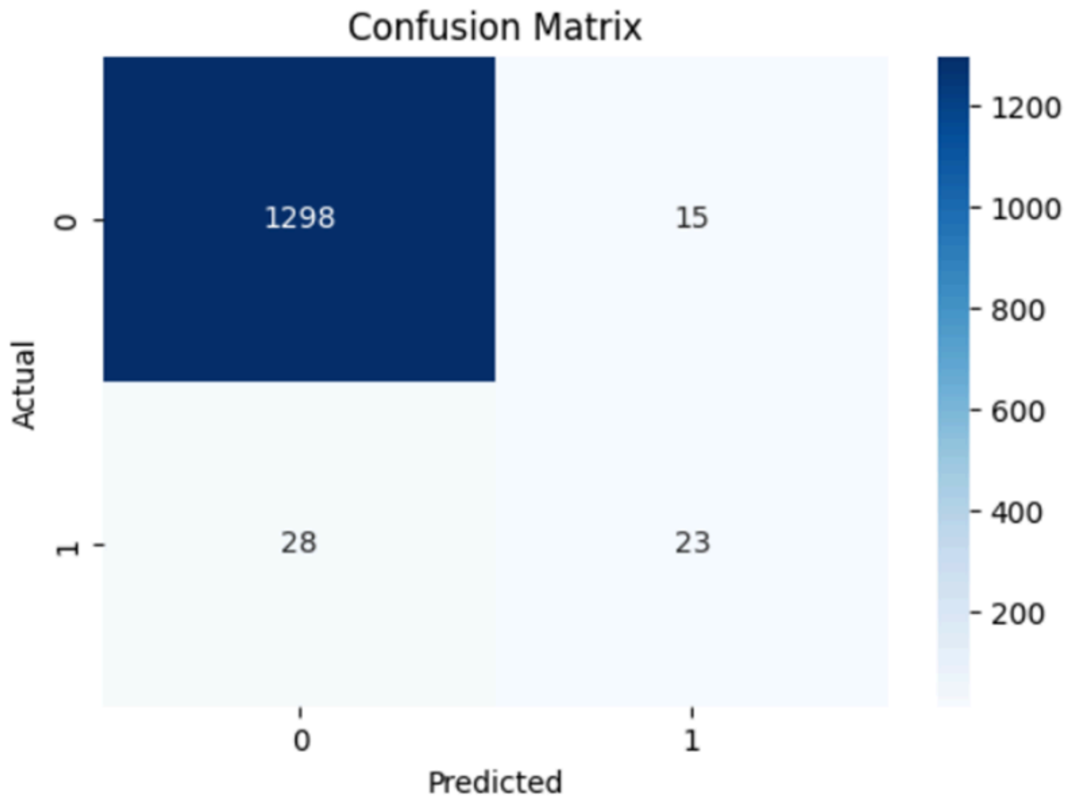


Figure 2: Confusion Matrix Visualization

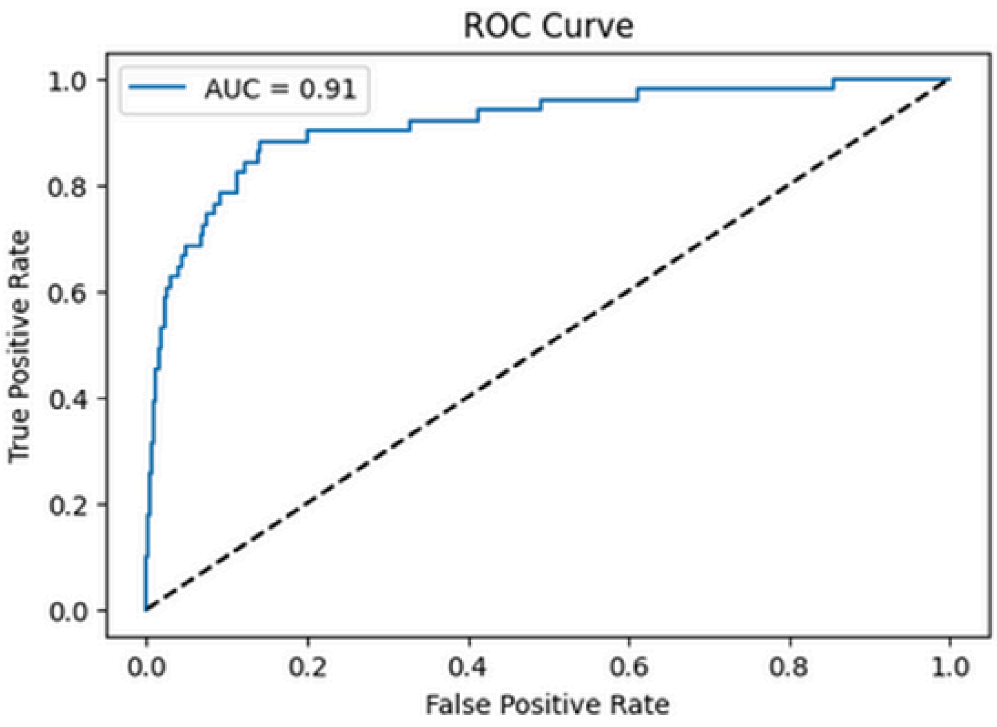


Figure 3:ROC Curve

Hyperparameter Tuning

We applied GridSearchCV to identify the best combination of parameters such as learning rate, max depth, and number of estimators. After tuning, the model showed an improvement in both accuracy and F1-score on the test set.

Conclusion

The XGBoost model proved to be a powerful and accurate classifier for our dataset. Through careful preprocessing, feature selection, and parameter optimization, we significantly improved predictive performance. Visual tools like the confusion matrix and feature importance plots helped in interpreting results and validating model decisions.