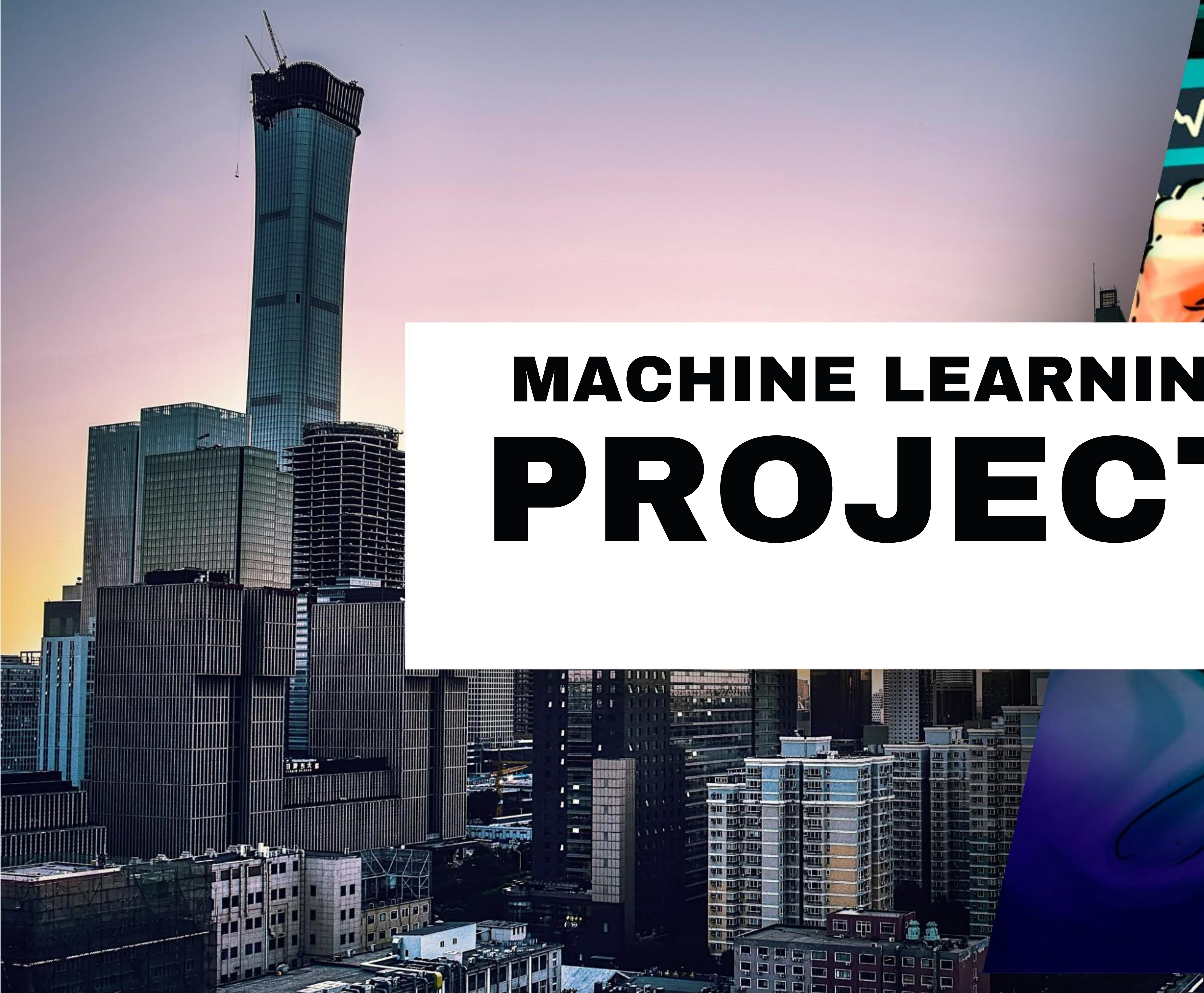


MACHINE LEARNING PROJECT

INTRODUCTION

Study of Bankruptcy data from the Taiwan Economic Journal for the years 1999–2009

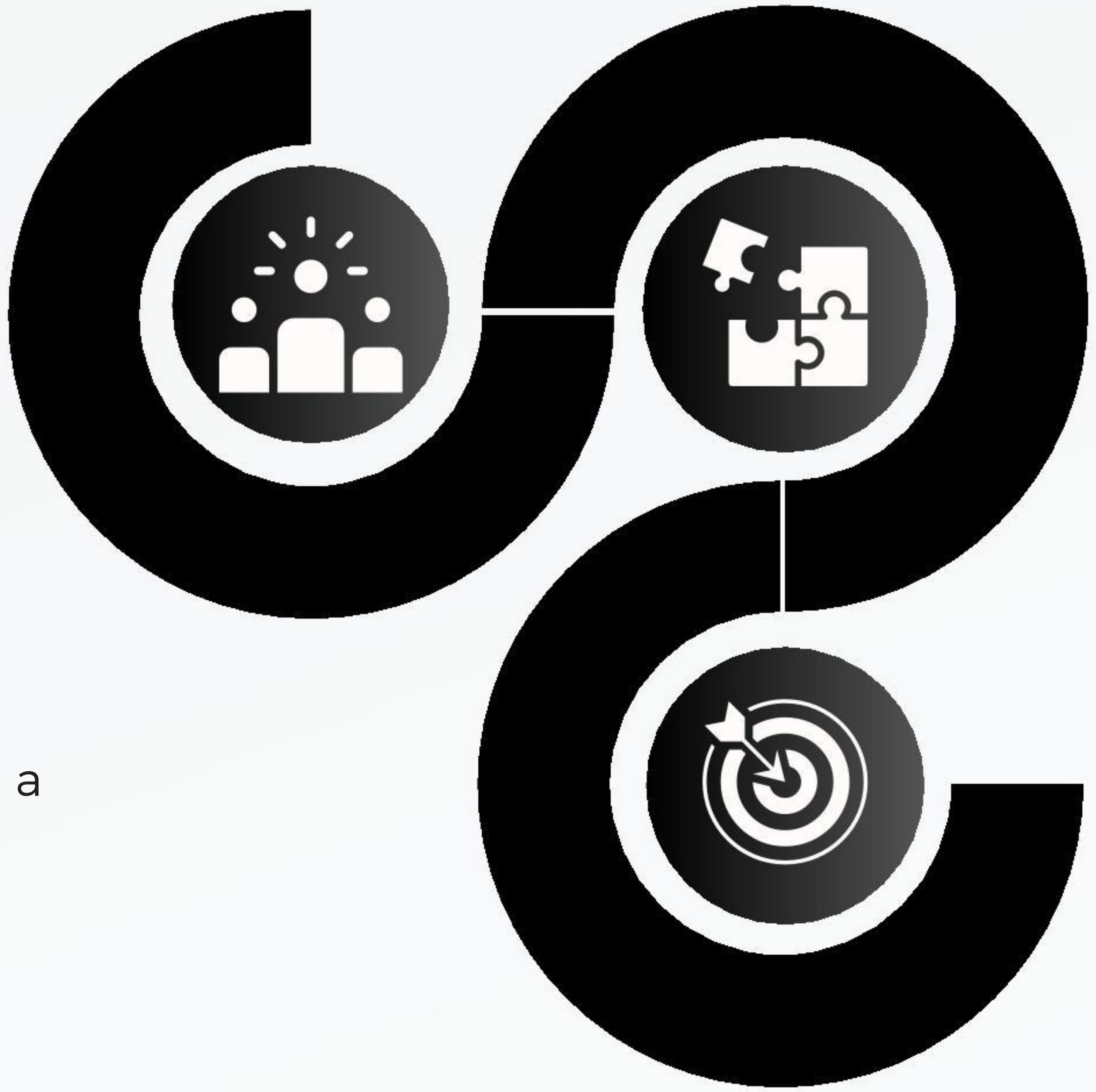
Company bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange

GOALS AND OBJECTIVES

01 Data Understanding and Preparation
Explore and prepare the dataset for effective machine learning model development.

02 Model Development and Training
Build and train machine learning models to predict bankruptcy based on the preprocessed dataset.

03 Model Evaluation and Deployment
Assess model performance, interpret results, and deploy a reliable bankruptcy prediction model.



CHALLENGES

01 **Financial Market Stability:** Bankruptcies can impact the stability of the financial market

02 **Investor Protection:** Early detection enables investors to take informed decisions to protect their investments

03 **Preservation of Employment:** Bankruptcy often leads to substancial job losses. Early prediction provides time to implement mitigation measures

04 **Risk Management for Creditors:** Creditors, such as banks and financial institutions, can anticipate and manage bankruptcy-related risks by adjusting their portfolios and lending practices.

PROJECT SUMMARY

The project is structured to begin with a data exploration and preprocessing phase, followed by a model development and training period, and concluding with a model evaluation and deployment phase for predicting bankruptcy among Taiwanese companies.

Model validation and deployment

Model development and training

Data exploration

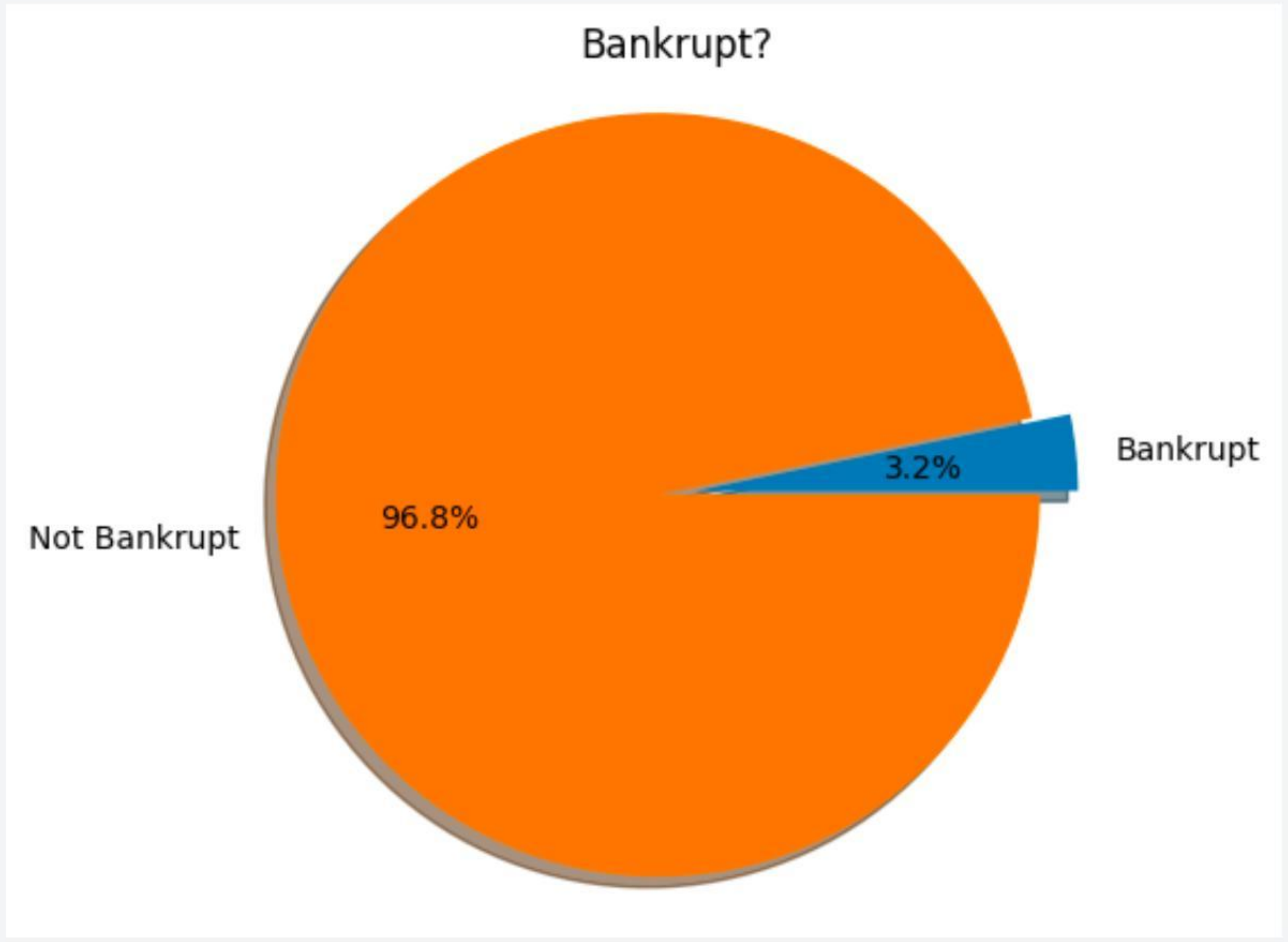
DATA EXPLORATION

Description of some features

The dataset is comprised of 95 features (96 columns - target).
The features are all some kind of financial constants, more or less correlated to each other and the target.
The dataset was quite challenging to visualise due to its very high number of features, but this feature quantity will be reduced for modelling.
Here is the definition of some features.

- **Debt Ratio:** %: Liability/Total Assets
- **Borrowing dependency:** Cost of Interest-bearing Debt
- **Current Liability to assets:** Total Assets /Total Current Liabilities
- **Current Liability to Equity** : Total Equity /Total Current Liabilities

Pie Chart of the distribution of the target variable



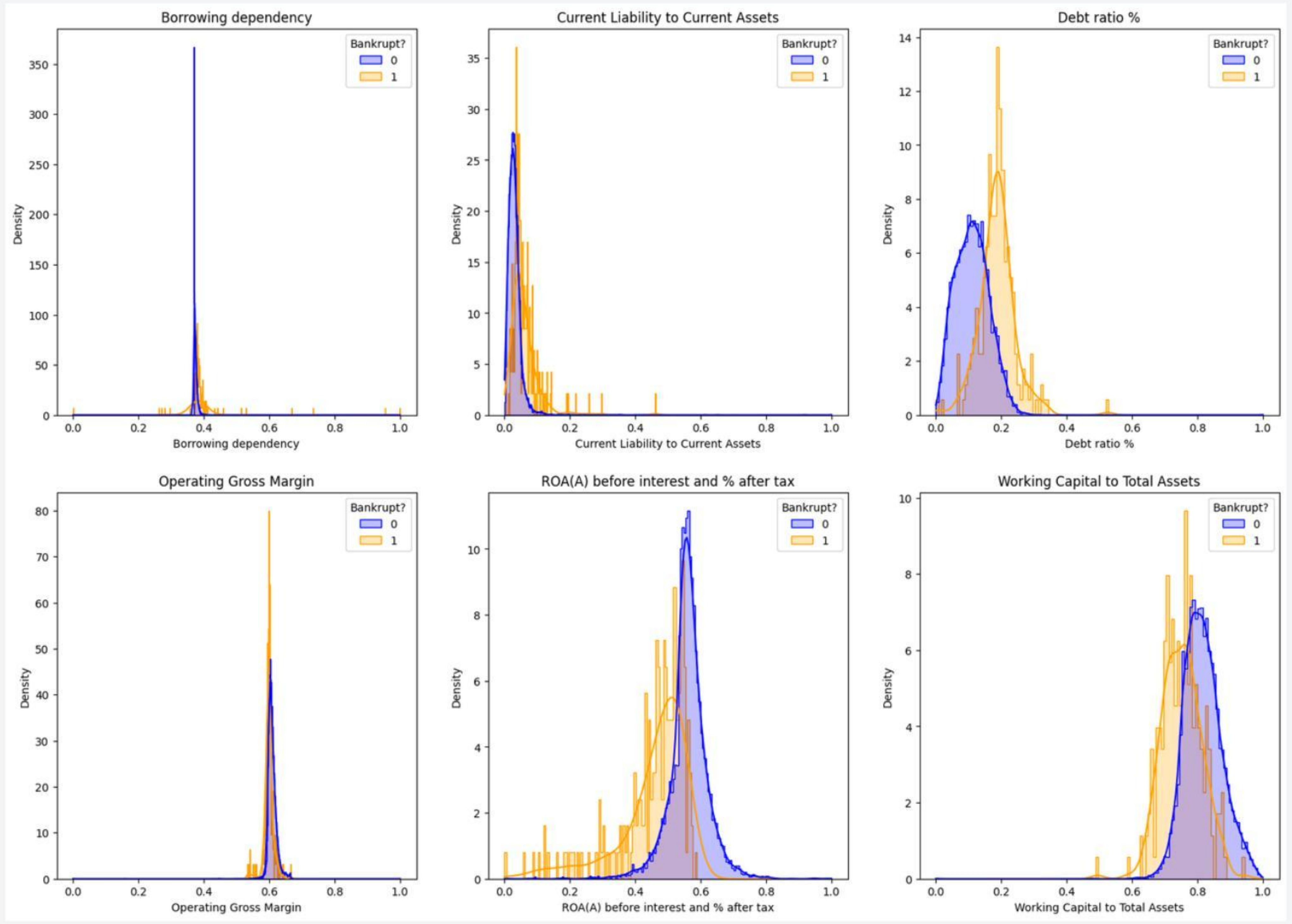
We are dealing with a highly imbalanced dataset, there are significantly more instances of class 0 than class 1. We will have to employ oversampling techniques such as SMOTE or ADASYN, to mitigate that imbalance and improve model performance

Distribution of some features

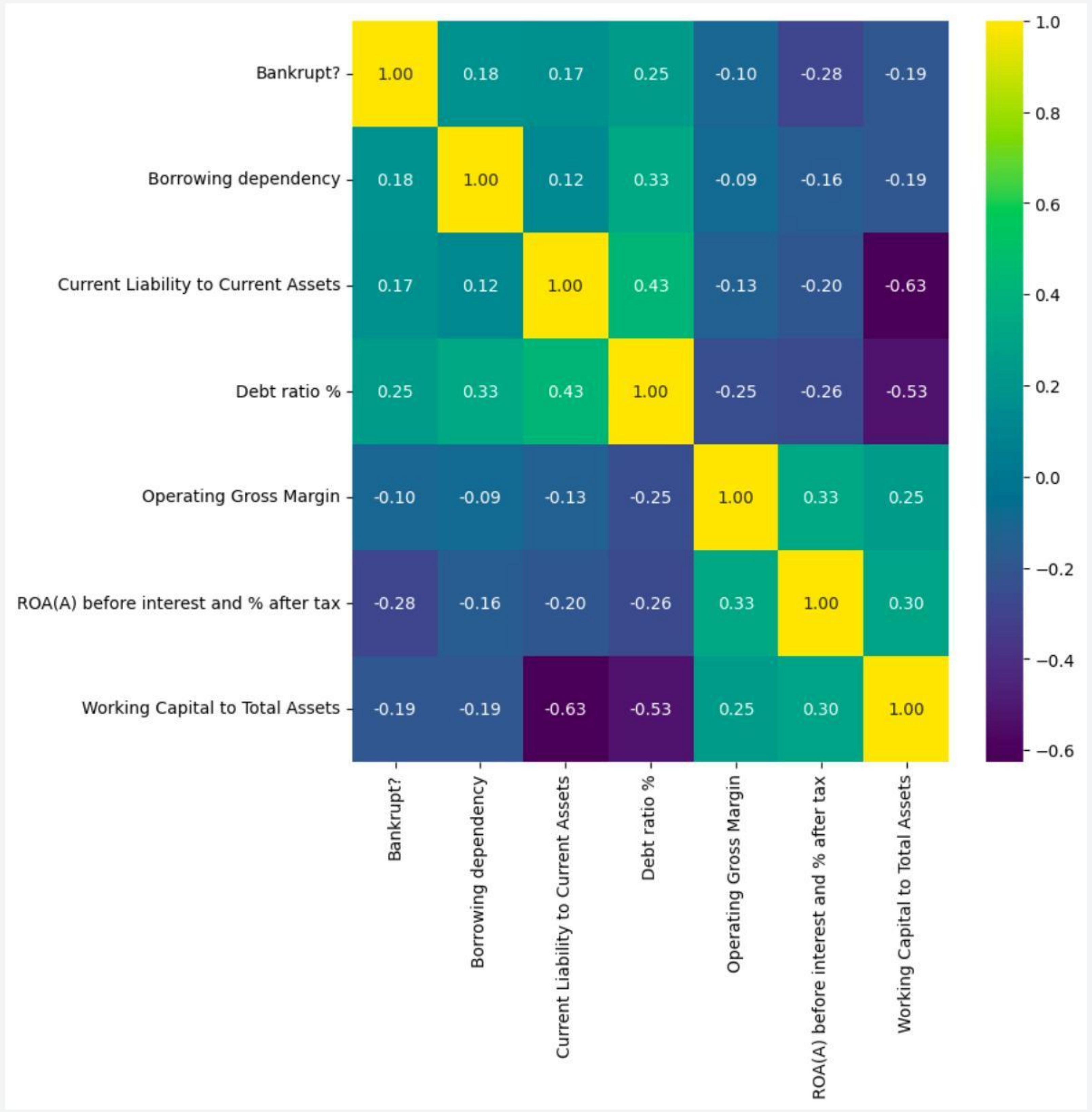
- The distribution of the features is a good indicator to predict if the company have a great probability of bankrupcy

- We are able to notice a distibution change between bankrupt and non-brankrupt companies

(The features shown here are the more correlated to the target ones)



Sample correlation matrix

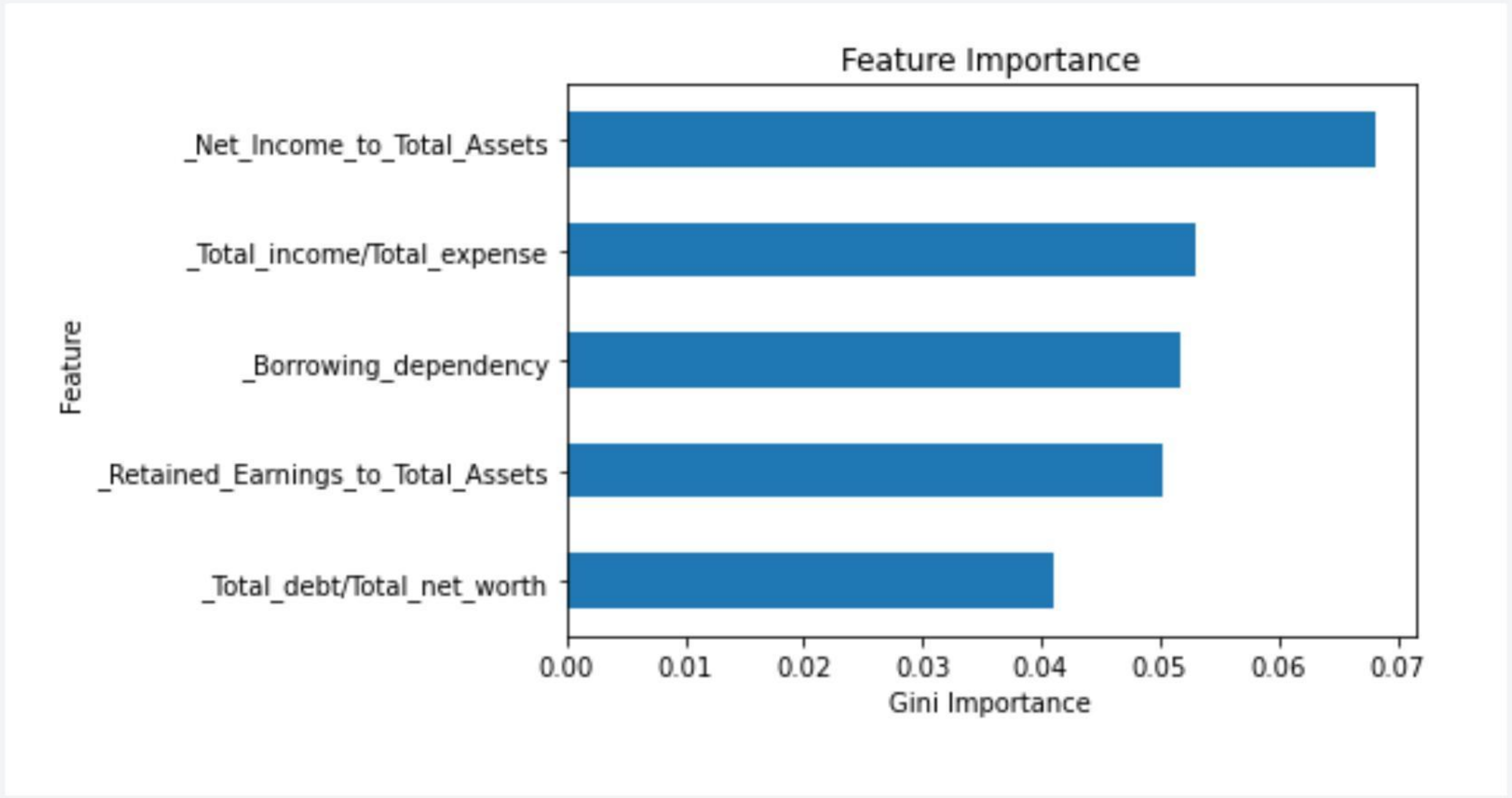


	Bankrupt?	Borrowing dependency	Current Liability to Current Assets	Debt ratio %	Operating Gross Margin	ROA(A) before interest and % after tax	Working Capital to Total Assets
Bankrupt?	1.00	0.18	0.17	0.25	-0.10	-0.28	-0.19
Borrowing dependency	0.18	1.00	0.12	0.33	-0.09	-0.16	-0.19
Current Liability to Current Assets	0.17	0.12	1.00	0.43	-0.13	-0.20	-0.63
Debt ratio %	0.25	0.33	0.43	1.00	-0.25	-0.26	-0.53
Operating Gross Margin	-0.10	-0.09	-0.13	-0.25	1.00	0.33	0.25
ROA(A) before interest and % after tax	-0.28	-0.16	-0.20	-0.26	0.33	1.00	0.30
Working Capital to Total Assets	-0.19	-0.19	-0.63	-0.53	0.25	0.30	1.00

- Here is a correlation matrix with the variables most correlated to brankruptcy

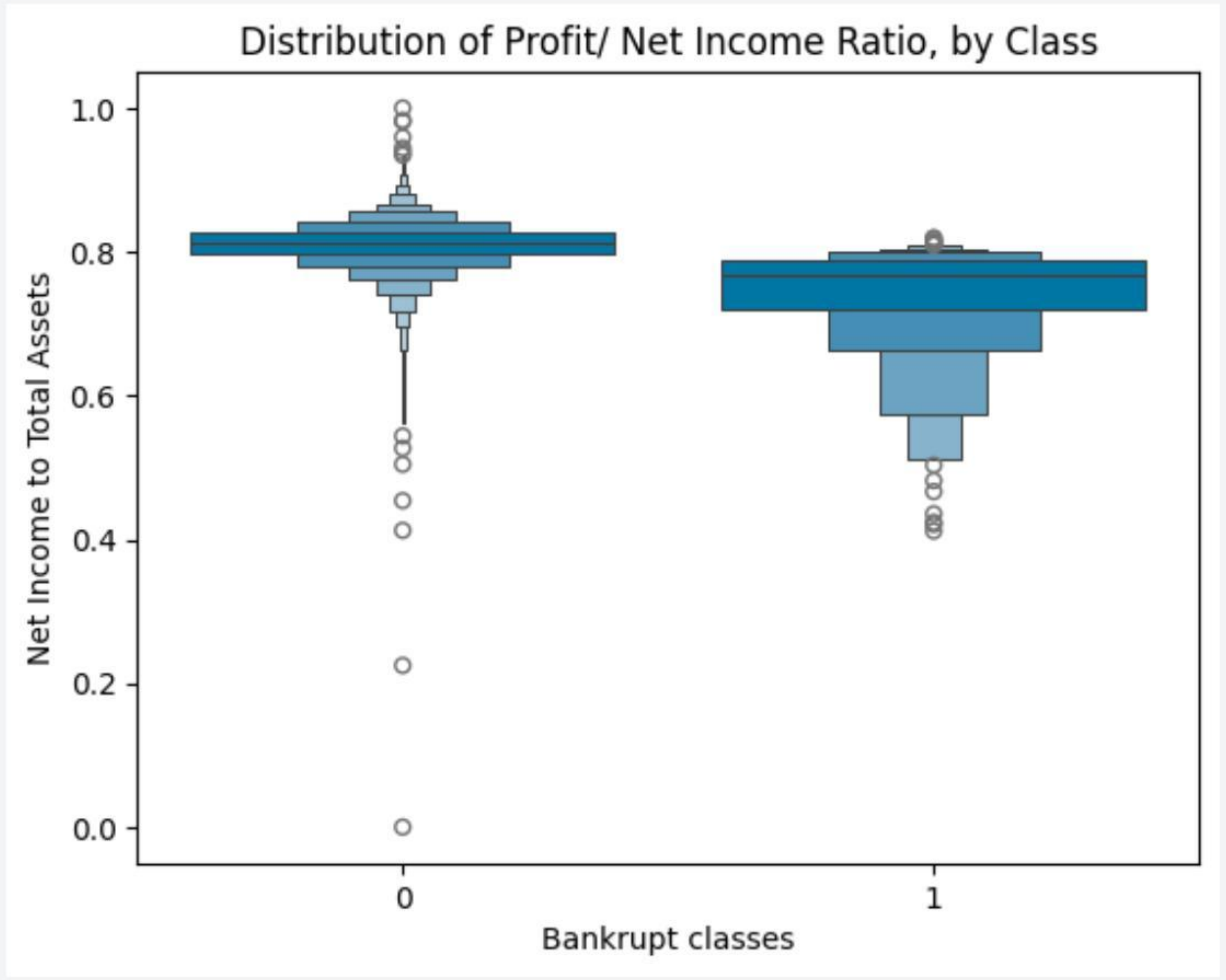
- There is a full correlation matrix in the notebook, but with 96 features it’s hard to see things

Feature importance on brankruptcy



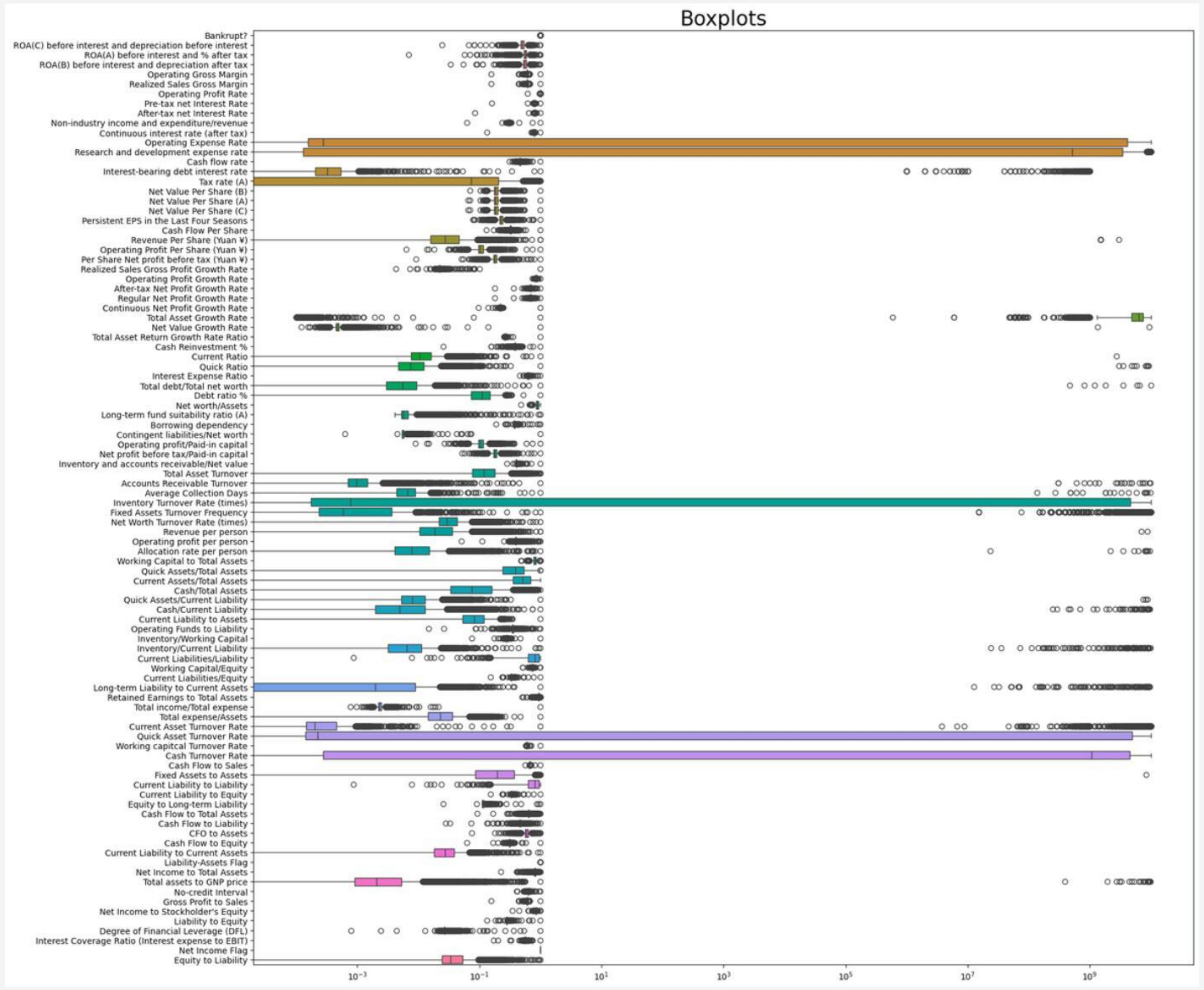
- This graph shows the features that most influence bankruptcy
- The data comes from a DecisionTree used further in the modeling process

Distribution of Profit/New Income Ratio, by Class



- We can see that when the percentage of net income on total assets is under 0.8, it is likely that the company is bankrupt

Outliers



- There are a lot of outliers in the dataset
- We will have to treat those before training the model
- The method used to treat them is discussed in the model development part of the presentation

MODEL DEVELOPMENT

Data cleaning

- Pretty clean dataset overall
  - No missing values
  - No duplicate rows
  - 96 features, will have to be reduced for modelling
  - Outliers are present, will have to be treated
  - 6819 columns

- Some labels are very skewed, like the target, we will have to treat those features

Data preprocessing

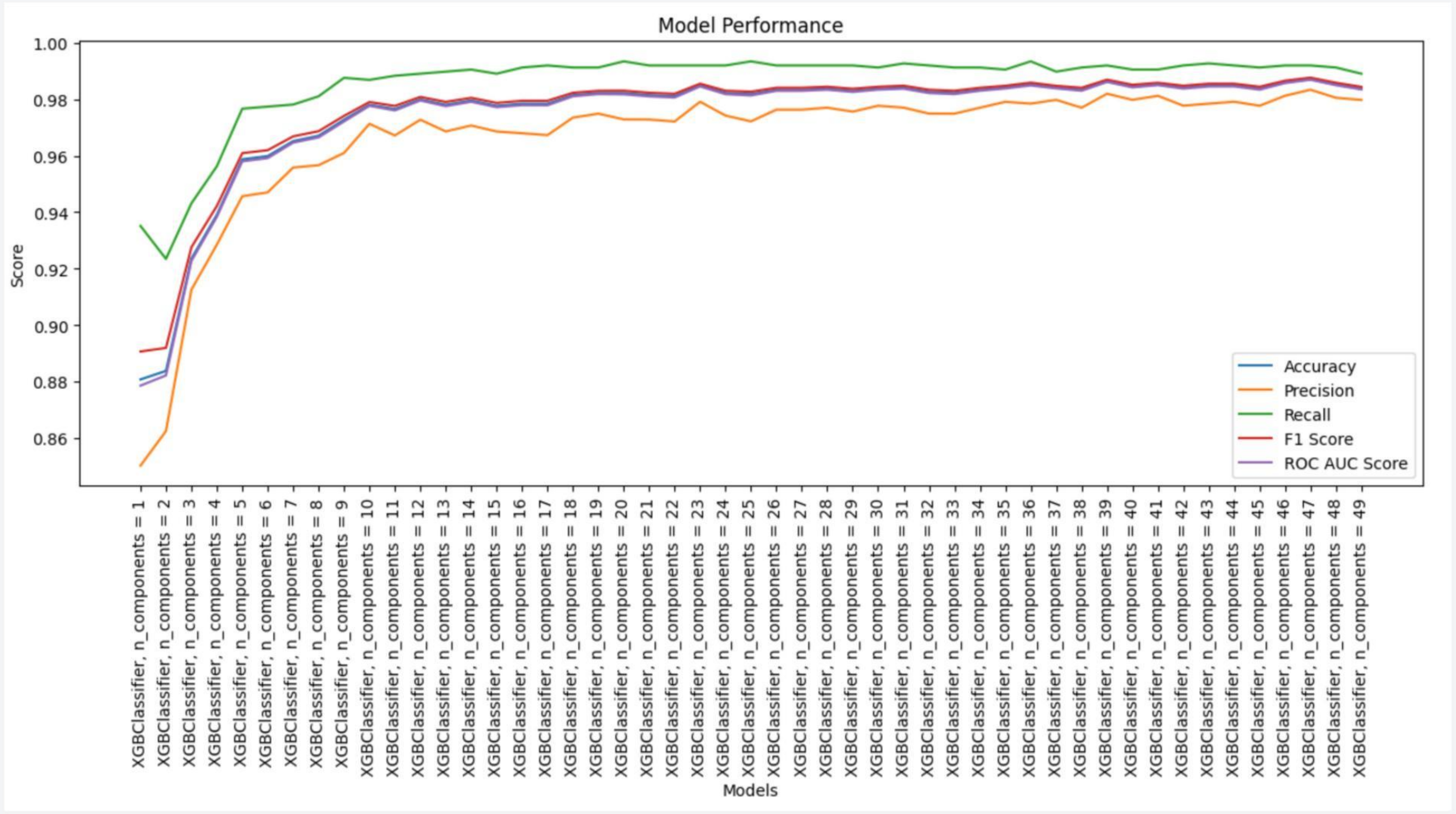
- Outliers treatment
    - Outliers above 85% quantile or below 15% quantile were replaced by threshold values
    - Chosen because it led to less data loss and better results overall

- Feature reduction
    - Applied PCA to reduce the features from 95 to 27

- No need for skewness treatment due to the use of PCA
    - New distributions show in notebook

- Used BorderlineSMOTE oversampling to equilibrate the target feature into an equal amount of Bankrupt and non-bankrupt companies

PCA n_components selection



We will choose 20 features, as it seems the model doesn't get better, and less features will be faster
to train

Models

- We are looking at a classification project

- We will use regular classification algorithms
  - LogisticRegression
  - RandomForestClassifier
  - XGBClassifier
  - CatBoostClassifier
  - LGBM Classifier
  - StackingClassifier

- We will use a function to streamline the process of spitting, training and evaluating

Grid Search

We will use grid search to search for optimal hyper-parameters

```
rf_params = {'n_estimators':[100,200,500,1000],
             'max_features':[3, 'sqrt', 'log2', None],
             'min_samples_split':[2,5,10,20],
             'max_samples': [0.25, 0.5, 0.75],
             'random_state':[42],
             'criterion':['gini','entropy', 'log_loss']}
rf = RandomForestClassifier()
rf_cv_model = GridSearchCV(rf, rf_params, cv=3, n_jobs=-1, verbose=2).fit(train_pca,y_train)
display(rf_cv_model.best_params_)
```

This will make sure we get the best results of our respective models

```
rf_tuned = RandomForestClassifier(**rf_cv_model.best_params_).fit(train_pca,y_train)
train_and_evaluate_model(rf_tuned)
```

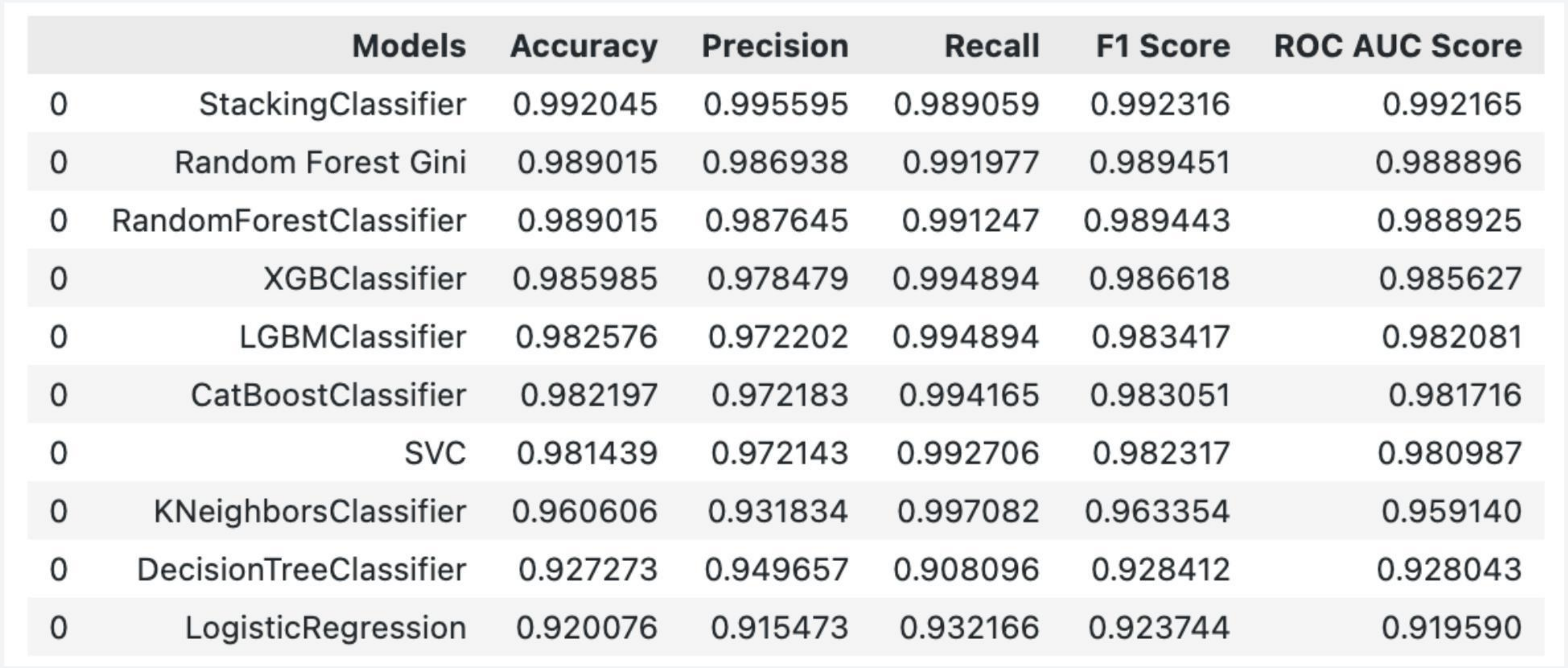
MODEL EVALUATION AND DEPLOYMENT

Model evaluation

We will use the usual metrics to measure the performance of our models:

- **Accuracy:** Measures the ratio of correctly predicted instances to the total instances in the dataset.
- **Precision:** Indicates the accuracy of positive predictions by measuring the ratio of correctly predicted positive observations to the total predicted positives.
- **Recall (Sensitivity):** Measures the ability of the model to capture all relevant instances by calculating the ratio of correctly predicted positive observations to all observations in the actual class.
- **F1 Score:** Represents a balanced measure, as the harmonic mean of precision and recall, considering both false positives and false negatives for comprehensive model evaluation.
- **ROC AUC Score:** quantifies the ability of a binary classification model to distinguish between classes by measuring the area under the ROC curve, where a higher score indicates better overall performance.
- **Confusion matrices**: table that summarizes the performance of a classification algorithm by displaying the counts of true positive, true negative, false positive, and false negative predictions.

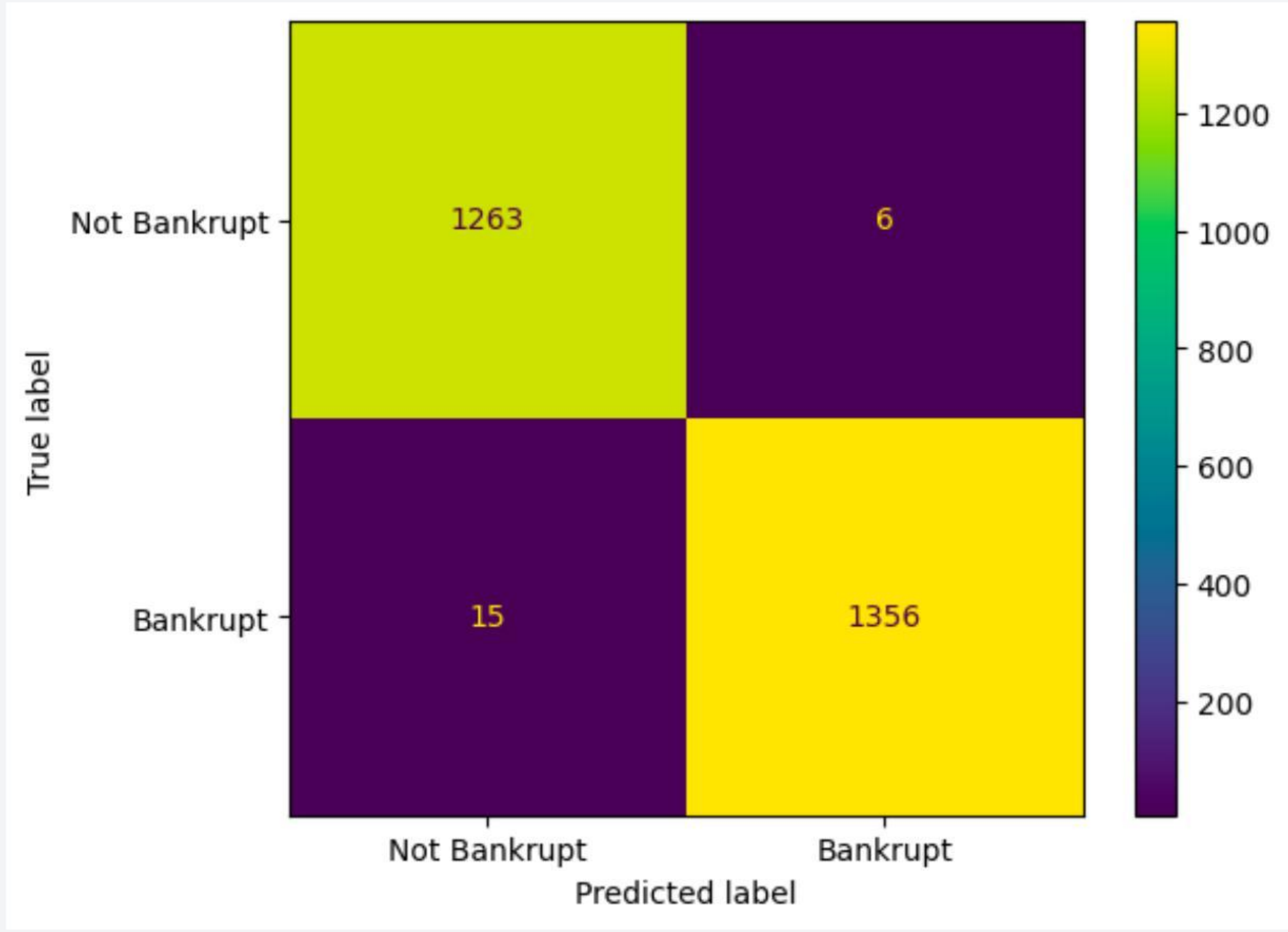
Results

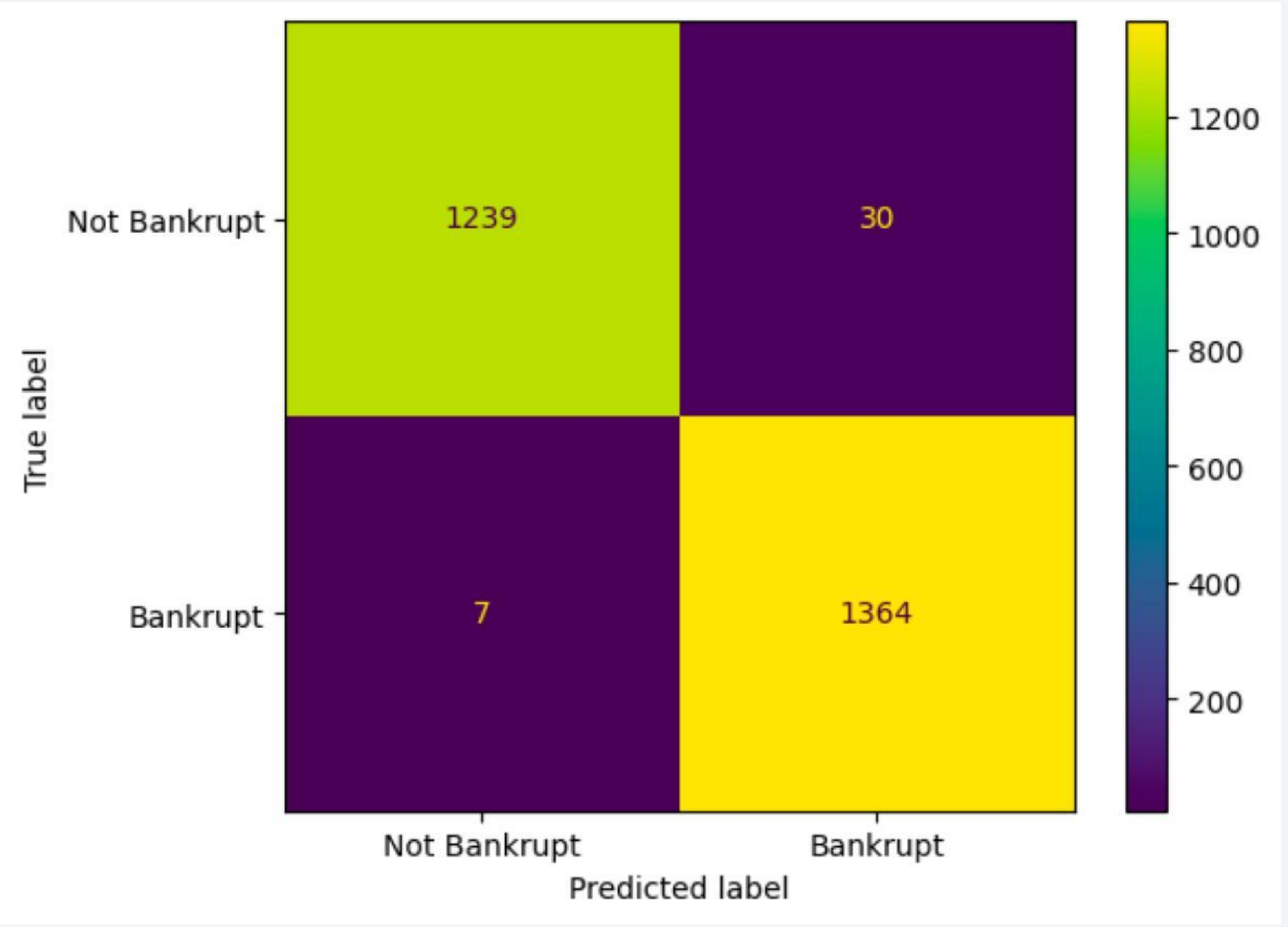


	Models	Accuracy	Precision	Recall	F1 Score	ROC AUC Score
0	StackingClassifier	0.992045	0.995595	0.989059	0.992316	0.992165
0	Random Forest Gini	0.989015	0.986938	0.991977	0.989451	0.988896
0	RandomForestClassifier	0.989015	0.987645	0.991247	0.989443	0.988925
0	XGBClassifier	0.985985	0.978479	0.994894	0.986618	0.985627
0	LGBMClassifier	0.982576	0.972202	0.994894	0.983417	0.982081
0	CatBoostClassifier	0.982197	0.972183	0.994165	0.983051	0.981716
0	SVC	0.981439	0.972143	0.992706	0.982317	0.980987
0	KNeighborsClassifier	0.960606	0.931834	0.997082	0.963354	0.959140
0	DecisionTreeClassifier	0.927273	0.949657	0.908096	0.928412	0.928043
0	LogisticRegression	0.920076	0.915473	0.932166	0.923744	0.919590

We can see a clear lead of tree-like classification algorithms, with the others still getting good results, but not on the same level

Confusion matrix

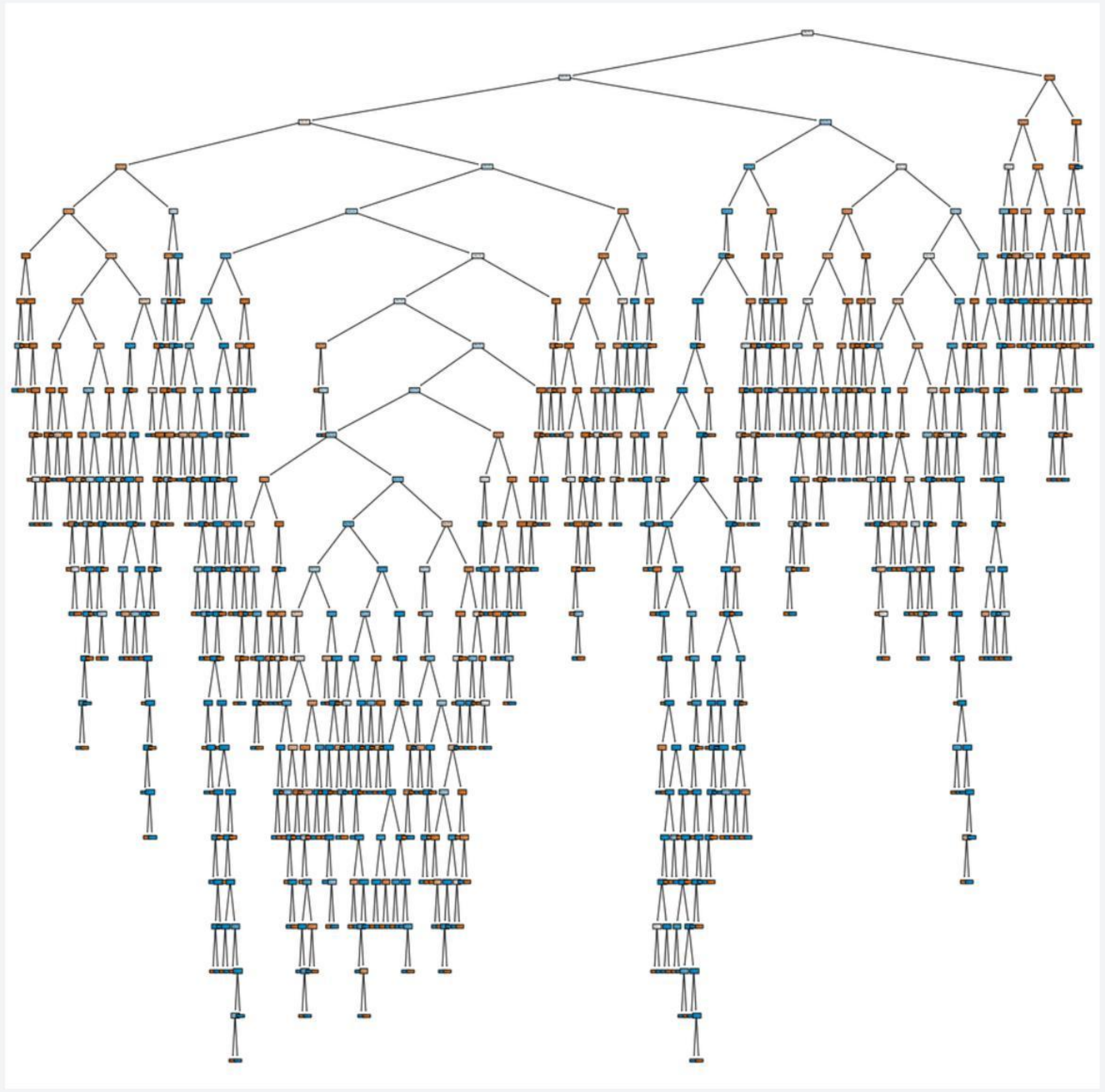




Stacking classifier

XGB classifier

Tree of the RFT Classifier



- We can see that the tree is very complex, with a high depth

- This level of complexity is to be awaited with almost 30 features, non correlating with each other

Thank you for reading.