Predicting Bankruptcy Using Machine Learning

by

Muhammad Qasim Idrees

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF APPLIED SCIENCE

In the Department of Electrical and Computer Engineering

We acknowledge and respect the Lək̓ʷəŋən (Songhees and Xʷsepsəm/Esquimalt) Peoples on whose territory the university stands, and the Lək̓ʷəŋən and W̱SÁNEĆ Peoples whose historical relationships with the land continue to this day.

Bankruptcy Prediction Using Machine Learning

by

Muhammad Qasim Idrees

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor

(Department of Electrical and Computer Engineering)

Dr. Mihai Sima, Departmental Member

(Department of Electrical and Computer Engineering)

Abstract

Bankruptcy is a serious issue in the financial sector. ML-based approaches have been widely used to predict bankruptcy, but selecting the right classifier and identifying the key features influencing Model results is essential for achieving good performance. This is especially important in the corporate sector where decision-makers may have a limited understanding of these models and their function. There is also a need for transparency and trust in their outcomes. This thesis utilizes twelve classifiers: CatBoost, RF, SVM, KNN, NB, LR, BDT, Stacking Ensemble, AdaBoost with RF, NN, FT-Transformer, and TabNet. These classifiers are evaluated in various settings using the Polish companies dataset to predict bankruptcy. The results obtained show that CatBoost is the best performing classifier. Further, SP-LIME is used to identify features that impact model decisions. SP-LIME ranks features based on their importance. A comparative analysis of SP-LIME and PCA indicates that SP-LIME identifies more influential features.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
AUC	Area under the Curve
AdaBoost	Adaptive Boosting
BDT	Bagged Decision Tree
BLOF	Bagging-Based Local Outlier Factor
BSM	Borderline SMOTE
CSFNN	Cuckoo Search Feed Forward Neural Network
CS-XGB	Cost-Sensitive Extreme Gradient Boosting
DT	Decision Tree
DL	Deep Learning
FNN	Feed Forward Neural Network
FT-Transformer	Feature Tokenizer+Transformer
KNN	K-Nearest Neighbors
LR	Logistic Regression
LIME	Local Interpretable Model-Agnostic Explanation
ML	Machine Learning
NN	Neural Network
NB	Naïve Bayes
PCA	Principal Component Analysis
RF-RFE	Random Forest-Recursive Feature Elimination
RF	Random Forest
SP-LIME	Submodular Pick Local Interpretable Model-Agnostic Explanations
SVM	Support Vector Machine
SHAP	Shapley Additive Explanation
SMOTE	Synthetic Minority Oversampling Technique
SAE	Stacked Auto-Encoders
SMOTE-ENN	SMOTE with Edited Nearest Neighbour
SGD	Stochastic Gradient Descent
SP	Submodular Pick
XG-Boost	Extreme Gradient Boost

Acknowledgment

First, I am very thankful to Almighty Allah for his countless blessings upon me. I also thank my parents and my wife for their continuous love, support, and encouragement.

I would like to express my deep gratitude to my supervisor, Dr. Thomas Aaron Gulliver, for accepting me into his research group as a master's student. His support and guidance are highly appreciated.

Finally, I am thankful to the Higher Education Department of Punjab, Pakistan, for granting me this opportunity.

Dedication

I dedicate this thesis to all those who have supported and inspired me throughout this academic pursuit. Your encouragement and belief in my abilities have been invaluable. This work reflects the collective support and guidance I have received, and I am grateful for the impact you have had on this journey.

Chapter 1 Introduction

Bankruptcy occurs when a business or individual is unable to pay their debts and is discharged from these debts following court proceedings. It is also called insolvency [1]. Bankruptcy is a serious problem that impacts many stakeholders [2]. The effects extend beyond companies and impact communities, workers, and investors. Thus, there is a need for effective bankruptcy prediction strategies within the financial sector. Bankruptcy prediction enables stakeholders to foresee potential financial losses before they materialize [2]. Because of its global impact on the banking sector, this topic has gained considerable attention. Bankruptcy usually occurs following a series of stages and does not occur suddenly, so different strategies are used to identify bankruptcy before it happens [2] [3]. Bankruptcy has gained attention in the latter half of the twentieth century throughout various recessions. The 2001-2003 recession saw the top ten largest bankruptcy filings, and 74 companies filed for bankruptcy during the 2008-2009 financial crisis with a total liability of $1.2 trillion [1]. Given global events and the current economic instability, even nations experiencing stable growth are concerned about financial viability. In this environment, anticipating bankruptcy is essential for stakeholders across various industries, especially in the corporate sector.

Due to the challenges of accurate prediction, the financial sector has made bankruptcy forecasting a priority. Stakeholders must thoroughly evaluate loan applications, emphasizing borrower ability to repay, which directly affects lending decisions. Financial institutions should offer clear and accessible loan terms to support the growth of both individuals and businesses. A withdrawal from lending could lead to economic inactivity, negatively impacting both national and global economies. Financial institutions face challenges in evaluating loan applications, so it is essential to leverage advances in technology. Improvements in computing have increased computational power and speed. This supports the use of artificial intelligence (AI) and machine learning (ML) to replace previous manual techniques for predicting bankruptcy. The research community is focused on developing better models to detect hidden patterns and provide more accurate results [4]. Logistic regression (LR) was first used in 1980 to assess borrower bankruptcy risk. However, a 1996 study suggested improvements were required to better model bankruptcy risk. ML models are commonly used for bankruptcy prediction, and their performance relies on multiple factors [5].

Data collection and accuracy are primary concerns because they directly impact ML model performance. In the financial sector, data availability and accuracy have become increasingly critical due to the high risk of errors. As a result, great care must be taken in processing financial sector-related data [4]. The increase in ML model usage is due to the higher importance of data in organizations. Many important decisions are made based on insights obtained from data using ML models. These models are now relevant to everyone, and their application has spread to every aspect of life. Furthermore, ML is rapidly evolving, resulting in the availability of improved algorithms to handle complex problems. This also makes ML models hard to understand due to their hidden internal operations. Therefore, they are called black-box models with visible inputs and outputs but hidden processes [6]. This makes it difficult to understand how these models make decisions [7]. Explainability is crucial for ML models. Executives and non-technical decision-makers need to understand the validity and interpretability of results. ML models should be explained in both technical and simple terms [8]. The major motivation behind the development of such new ML models is to find complex interactions and linear relationships in diverse data. However, their complex nature makes interpretation difficult [4]. ML techniques are considered better than linear techniques, but due to the lack of trust in ML-based methods, they are underutilized and often ignored by economists [9]. In [10], deep learning (DL) models, including conventional neural networks and artificial neural networks (ANN), were compared with LR models for predicting bankruptcy. The results indicate that traditional ML models like LR are less efficient than DL methods. This work serves as the primary inspiration for the research presented in this thesis.

The Polish companies dataset from Emerging Markets Information Services (EMIS) is available online at UCI Machine Learning Repository [11]. The goal of this work is to test and examine various classifiers for predicting bankruptcy using this dataset. The Submodular Pick Local Interpretable Model-agnostic Explanations (SP-LIME) technique is used to extract features from this dataset. A comparison of SP-LIME and PCA features is conducted. The primary objective is to use SP-LIME to identify relevant features, validate its outcomes through classification, and provide explainability for the model decision-making process.

The remainder of this thesis is organized as follows. Chapter 2 presents a review of the literature on ML-based bankruptcy prediction and associated studies using the Polish companies dataset.

The research gaps and thesis contributions are also given. Chapter 3 presents the methods used to accomplish the research objectives, and Chapter 4 presents the data preprocessing and experimental settings used. Chapter 5 presents the results obtained with 12 classifiers and the Polish companies dataset, and they are discussed in this chapter. Chapter 6 summarizes the thesis outcomes and provides directions for future work.

Chapter 2 Related Work

Bankruptcy has been around since ancient times along with research efforts to predict it. In [12], a survey of bankruptcy prediction strategies from 1930 was presented. These investigations employed single-variable (univariate) approaches for predicting bankruptcy. In 1968, the first multivariate model was published, and it has since acquired significant importance. Traditional methods remain valuable for their simplicity and consistency, but the analysis in [5] for default events and predicting bankruptcy suggests that ML models are superior to traditional approaches. In [13] and [14], Support Vector Machine (SVM) models were shown to provide better accuracy than other models.

Three ensemble-based ML classifiers were considered in [15] with a focus on classification noise, feature noise, and missing values, and how they affect the performance. The classifiers used were Adaptive Boosting (AdaBoost), Bagging, and DECORATE. In the case of feature noise, all three techniques perform better than the base learner, J48, while with missing values, DECORATE is the best option. Bagging outperforms the others when there is classification noise. In [16], a feature selection method called the Shapley Additive explanation approach (SHAP) was introduced to provide feature importance called SHAP values. This helps find important features from the dataset to provide better interoperability. The results obtained indicate that the performance with SHAP is as good as other feature selection methods.

ML classifier performance is highly dependent on the quality of data, so preprocessing is important. Missing values, data scaling, and data balancing are the most common steps in preprocessing. The preprocessing in [17] included data balancing using the Synthetic Sample Generation Technique (SMOTE), data normalization using the Standard Scaler approach, and feature reduction via Principal Component Analysis (PCA). The Random Forest (RF), Decision Tree (DT), K Nearest Neighbor (KNN), LR, and Neural Network (NN) classifiers were evaluated. The results obtained suggest that the PCA technique for reducing features does not improve the prediction performance with the Polish companies dataset. The KNN and LR classifiers had the worst performance.

An extended Extreme Gradient Boosting-based approach was used in [18] to predict bankruptcy using the Polish companies dataset. This method randomly generates synthetic features using

random arithmetic operations. The results obtained show significant improvement in prediction quality. In [19], the effect of imbalanced class distribution in the data was studied. Seven different ensemble classifiers were utilized with different class ratios to evaluate the effect of the distribution on classifier performance. In [20], five ensemble classifiers were employed with three types of financial datasets: quantitative, qualitative, and a combination of these two. In terms of misclassification rate and processing, AdaBoost outperformed the other models as it had the lowest misclassification rate. These results indicate that ensemble models perform better with financial data.

Outliers are data instances that appear distinct from the rest of the dataset and include abnormalities. In [21], a bagging-based local outlier factor (BLOF) algorithm was proposed as the first step of a three-step framework. BLOF reduces the computational complexity and provides better outlier detection. The training set is divided into training subsets, and sub outliers are identified using the local outlier factor algorithm (LOF). These are then aggregated through a voting strategy to obtain aggregated outliers and are boosted back into the training set to form the outlier adapted training set. Thus, this set contains both the original and aggregated outliers. In the second step of the framework, the outlier adapted training set is input to the gradient boosting decision tree method. In the final stage, self-adaptive parameter optimization using stacking-based ensemble learning is used. The results indicate that the proposed method outperforms existing approaches. In [22], BLOF was shown to better predict bankruptcy than AdaBoost, Decision Tree, J48, and Random Forest using the Polish companies dataset with 97% accuracy on first- and second-year data.

DL algorithms have complex internal structures that can better identify hidden patterns in large datasets. Stacked Auto-Encoders (SAE) were used in [23] to forecast bankruptcy. A two-layer autoencoder was used for feature extraction followed by a SoftMax classifier in the second stage to obtain class labels. Results obtained using the Polish companies and Darden datasets show that this technique outperforms other methods. A two-layer SAE and SoftMax classifier was presented in [24]. The Borderline Synthetic Minority oversampling approach (BSM), also known as Borderline-SMOTE, was used to handle imbalanced data, with SAE and SoftMax as in [23]. Data characteristics were learned using a two-layer autoencoder. Label classification was done using SoftMax, and the performance was improved by fine-tuning the hidden layers. The results obtained

show that using BSM with SAE is better in terms of area under the curve (AUC), but the training time is high.

A comparative analysis of three bagging ensembles (SVM, RF, and KNN), two boosting ensembles (AdaBoost and Extreme Gradient Boosting), and three DL methods (Long Short Term Memory, Deep Belief Network, and Six-layer Multilayer Perceptron) was presented in [25]. Three unbalanced datasets and five oversampling and three under-sampling strategies were used. A six-layer multilayer perceptron with a dataset generated using the Synthetic Minority Oversampling Technique with Edited Nearest Neighbour (SMOTE-ENN) was shown to be more effective at predicting bankruptcy than the other classifiers. SMOTE-ENN was the most effective sampling method for all DL techniques. Furthermore, the results with these DL methods are better than those previously reported using the same dataset.

Noisy observations, class imbalance, and overlap were addressed in [26] using an ANN technique called Overlap-Sensitive. Unlike earlier approaches that identify only individual data instances, this strategy detects overlapping regions without relying on multiple classifiers. It utilizes weighting based on the locations of the data instances. Results were obtained using 12 simulated and 23 real datasets that indicate an improvement in the imbalance ratio and overlapping levels.

The cuckoo search feedforward neural network (CSFNN) was presented in [27] as an extension of the feedforward neural network (FNN) that uses the Cuckoo Search Algorithm to optimize the weights. Manufacturing data collected from two different periods was used for performance evaluation. The results obtained show that the CSFNN outperforms LR and the backpropagation feedforward method in terms of accuracy for both periods.

In [28], hybrid approaches were used to predict bankruptcy using the Polish companies dataset. Including descriptive statistical analysis method, fuzzy rough with real-coded genetic algorithm, rough set, hybridization of the binary-coded genetic algorithm with neural networks, and rough set with real-coded genetic algorithm. The results obtained show that rough set hybridization of the binary-coded genetic algorithm performs better than statistical approaches with a 98.3% accuracy. It was also determined that efficiency decreases with the number of features.

In [29], a comparison of Random Forest-Recursive Feature Elimination (RF-RFE), which chooses pertinent features, and PCA, which reduces the number of features in a dataset, was provided. The

AdaBoost and Stochastic Gradient Descent (SGD) classifiers were used with the datasets generated using RF-RFE and PCA. The results obtained show that the RF-RFE dataset is better overall for both classifiers.

Cost-Sensitive Extreme Gradient Boosting (CS-XGB) was proposed in [30] to solve the issue of noise that is added while balancing data. This method eliminates the need for data resampling. The main idea is to give greater weight to misclassified positive samples, which results in more attention being given to these samples. Results were obtained using the lending club and Polish companies datasets. The results obtained reveal that CS-XGB outperforms other models, including RF, Bagging, AdaBoost, Extreme Gradient Boost (XG-Boost), and SMOTE-XGB, and provides better prediction results.

In [31], oversampling techniques were considered with the Polish companies dataset. The oversampling techniques used were SMOTE, Borderline-SMOTE, and ADASYN. The resulting datasets were tested using two ensemble classifiers. The first ensemble method is a combination of LR and SVM, and the second is a combination of RF and XG-Boost. The results obtained show that using a combination of oversampling and ensemble techniques reduces the number of false detections. In [32], a hybrid approach based on a combination of XG-Boost and ANN models was investigated. The class imbalance was resolved through optimization using a genetic algorithm and particle swarm optimization. The results obtained show that this hybrid approach achieves better accuracy even when used without feature selection techniques.

A granular-based method for imputing missing values was presented in [33]. This technique uses semantics for the formation of granules around missing values. The results obtained show that this method provides equal or better accuracy than an autoencoder estimator. However, this technique is inefficient when the dataset has more than 50% missing values.

2.1 Existing Gaps in The Literature and Thesis Contributions

Most of the research conducted using the Polish companies dataset has focused on preprocessing strategies such as data balancing and handling missing information. Thus, this dataset requires a thorough analysis using ML classifiers. Further, it should be investigated under different training and testing conditions. Most studies focus on the non-bankrupt class and overlook the model

performance with the bankrupt class. The literature survey also revealed a research gap concerning the interpretability and explainability of model results with a focus on the most influential features.

This thesis considers twelve classifiers with the Polish companies dataset, including older and newer models. Further, this dataset is tested for combined and individual years to identify the most effective ML classifiers considering both bankrupt and non-bankrupt classes. Then SP-LIME is applied to the best-performing classifiers to identify the features that have the greatest impact on model decisions. These features are compared with those identified using PCA.

Chapter 3 Methodology

This chapter presents the methodology for bankruptcy prediction using the Polish companies dataset. Figure 1 gives an overview of the research methodology. This includes several steps, beginning with preprocessing which involves missing value imputation, normalization, and data balancing. Then the five years combined dataset is used with 12 classifiers, and results are obtained with both balanced and imbalanced data. The top 7 classifiers from this step are used for classification using the individual year datasets for training and the remaining individual and four years combined datasets for testing. From this step, the three best classifiers are chosen and used for classification using the five years combined dataset. Then the most influential features are identified using SP-LIME and PCA, and classification is performed using the three classifiers with the top features from both approaches. From the results, CatBoost was selected as the best performing classifier as it consistently had the best performance. In the last step, SP-LIME with CatBoost was again evaluated to confirm the previous results with the five years combined dataset.

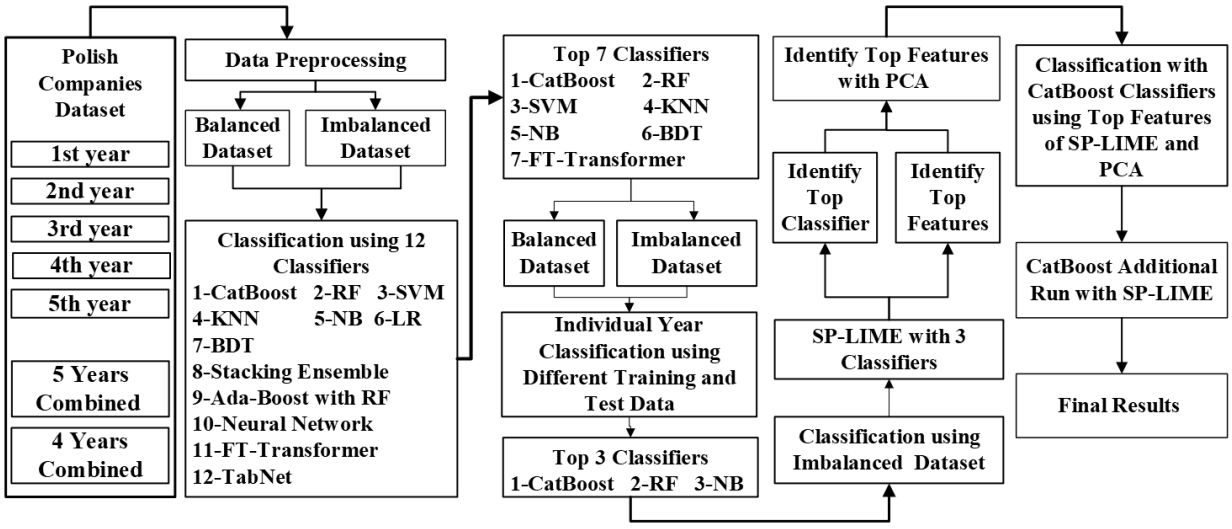

Figure 1: Research Methodology Overview

3.1 Supervised Machine Learning

With supervised machine learning, models are trained on labeled datasets. In these datasets, each instance is known to belong to a certain class. The classifiers learn from the training data and then

predict outcomes on new or unseen data. Supervised ML has been used to solve many complex real-world classification and regression problems. The twelve classifiers used in this thesis are supervised ML classifiers, and they are described below.

3.1.1 CatBoost

CatBoost is a variation of gradient-boosting that employs a binary decision tree as the base predictor. First, an initial decision tree is constructed and evaluated for errors, and then the next tree is constructed to correct these errors. This process continues for a predefined number of iterations. In this way, the error is minimized to improve model prediction. The goal of this classifier is to minimize the expected loss

$$L(H) = \mathbb{E}L\,(Y, H(X)) \qquad (1)$$

where H is a training function, $\mathbb{E}$ is the expectation operator, and X and Y are instances from the training data. The data instances are considered to be independent and identically distributed according to an unknown distribution [34]. CatBoost is good at predicting categorical features, making it ideal for complex tasks. CatBoost solves the issue of target leakage found in most gradient-boosting methods. Target leakage means the target variable is accidentally included in the training data features. In gradient boosting, this occurs when the entire dataset is split for use with the decision trees, including the target variable. CatBoost uses a permutation-based order-boosting technique, making it suitable for categorical features. This also improves efficiency and reduces training time [35].

3.1.2 Random Forest

Random Forest (RF) is a tree-based ensemble learning classifier. It is based on bootstrap aggregation where randomly selected samples of training data are tested on unique decision trees using a replacement strategy. This provides each tree learner with unique training data [16] and the randomness makes it robust to overfitting. RF is used for both classification and regression tasks [34].

3.1.3 Support Vector Machine

Support Vector Machine (SVM) is a popular technique for regression and classification tasks. It uses a kernel function to calculate the distance between similar data instances [5]. SVM determines the boundary (hyperplane) between two classes while keeping the maximum possible distance between them. The most frequently used kernels include linear, polynomial, sigmoid, and Gaussian radial basis functions [25].

3.1.4 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a widely used nonparametric algorithm. In KNN, the class of a data instance is decided by the *K* closest neighbors through majority voting, with typical value *K* = 5 [24]. Distance algorithms such as Euclidean, Mahalanobis, Minkowski, and Hamming are used to measure the distance between data points [25].

3.1.5 Naïve Bayes

Naïve Bayes (NB) is an algorithm used for classification tasks based on Bayes' Theorem. It assumes that feature pairs in the model are independent. NB is widely used for text classification. One of the key advantages of this model is its prediction speed, even with high-dimensional data [36].

3.1.6 Logistic Regression

Logistic Regression (LR) is a popular ML technique with applications in areas such as medicine, business, and economics. It is simple and so is a preferred choice for many classification tasks. Relationships between input features and the target variable are identified, and predictions are made based on these relationships. It uses features to classify the input data probabilistically [10].

3.1.7 Bagged Decision Tree

Decision Tree (DT) divides the dataset into smaller sets. A tree structure is used where features are represented by internal and leaf nodes connected through branches representing feature values [2]. Bagged Decision Tree (BDT) is an improved DT model that uses bagging, an ensemble

averaging approach also known as bootstrap aggregating. This model picks random data from the original dataset to train individual DT models to improve the stability and accuracy [37].

3.1.8 Stacking Ensemble

Stacking is a strategy that combines classifiers to improve performance by taking advantage of their strengths. A Stacking Ensemble consists of two primary components, base classifiers and a meta-classifier. Each base classifier is trained using a different learning algorithm and generates predictions for each instance separately. These predictions are then passed to the meta-classifier which is trained to combine the outputs of the base classifiers to make the final prediction. This technique improves the overall performance by leveraging base classifier capabilities [38].

3.1.9 AdaBoost with Random Forest

Adaptive Boost (AdaBoost) is a popular boosting technique and RF is an ensemble learning method often used for high-dimensional classification problems. AdaBoost is a learning algorithm that combines multiple weak classifiers (weak learners) to build a strong model. In this case, RF is used as a weak learner to generate prediction models with fewer incorrect predictions. The objective is to improve overall performance [39].

3.1.10 Neural Network

A Neural Network (NN) imitates the internal structure of the human brain and is used to solve classification problems. In its most basic form, it consists of one input layer, one hidden layer, and one output layer. The input layer receives the data, and each feature value is multiplied by a weight which indicates its importance. The hidden layer transforms the input features to provide a useful representation. An activation function, commonly a sigmoid function, is applied after the hidden layer, and this is followed by the output layer which gives the classification result. Backpropagation is used to adjust the weights to improve performance [34].

3.1.11 FT-Transformer

Feature Tokenizer+Transformer (FT-Transformer) is a simple application of the transformer architecture designed for tabular data (data organized in rows and columns). It consists of a tokenizer module that converts features to embeddings, which is a method to represent categorical

features as numbers. The transformer module processes the embeddings and generates tokens. These tokens are used for the final prediction. FT-Transformer requires more training resources than other models due to the additional feature transformation layer and increased model complexity [40].

3.1.12 TabNet

TabNet is a deep neural network technique designed for tabular data. It uses the sequential attention method to select the most relevant subset of features for each instance (instance-wise selection), which makes the model an efficient learner. TabNet is based on a recurrent architecture where each step depends on the outcome of the previous step. This design enables more interpretable decision-making and allows the model to make better decisions. The TabNet architecture consists of an encoder, decoder, feature transformer, and attentive transformer [41].

3.2 Local Interpretable Model-Agnostic Explanation (LIME)

LIME is a model-agnostic method that generates interpretable explanations for ML models. It explains the classifier local relationship to the instance for which it makes a prediction. LIME investigates how small changes in the input affect model predictions, then identifies which input features influence the final prediction. It then creates modified (perturbed) instances that are more like the original instances and uses them to create an interpretable (understandable) model [42]. The goal of LIME (or any model explainer) is to explain model results so non-experts can understand them and thereby increase trust in the model. There are four key properties that a model explainer should have and they are discussed below.

Interpretable

A model explainer should interpret the model results in an easy to understand manner and clearly explain how input variables relate to the model results. LIME offers a simple way to interpret data instances. The meaning of interpretable may vary from case to case. In the case of text classification, interpretable means a binary vector (list of values) indicating the presence or absence of certain words [42].

Model Agnostic

A model explainer is considered model agnostic if it can explain any ML model, regardless of its structure. A model agnostic explainer focuses on features. LIME is model agnostic since it can explain any ML model [42].

Local Fidelity

Local fidelity is one of the key aspects of a model explainer. The idea of altering the original data instances, using these as input to a black box model, and then observing the model results was considered in [42]. These modified instances are weighted based on how similar they are to the original instances, and these modified instances form the local proximity around an instance. The weighted instances are then used to fit a simpler model, such as a linear model. It is easier to explain a simple model from a local perspective rather than providing a global explanation. Considering this, the model explainer should explain an instance in relation to data instances in its proximity. Thus, the model explanation might not be true globally, but it should be faithful in its local proximity, and this is called local fidelity.

Global Perspective

The fourth property of any explainer is that it should provide a representative set of data instances that provide a global explanation to the user regarding model results. This helps the user understand model behavior across the entire dataset.

LIME is an efficient tool for building trust in ML models [42]. Two traits of LIME, interpretable data representation and fidelity interpretability, make LIME an efficient explainer. The tradeoff between them means LIME must find a balance between showing the model decisions correctly (fidelity) and making the explanation simple enough for people to understand (interpretability) [42]. Since the explainer needs to be locally faithful, local fidelity is very important, and interpretability must be maintained. This means the explainer should replicate model behavior from the local perspective while maintaining interpretability. The explanation balance between local fidelity and interpretability is expressed as [42]

$$Explanation(x) = \underset{g \varepsilon G}{\operatorname{argmin}}\, L(f, g, \pi_x) + \Omega(g) \qquad (2)$$

where x is the instance being explained and g is a simple, interpretable model such as a linear or decision tree model. $L(f, g, \pi_x)$ is the locality aware loss function. It measures how well g

approximates the original model *f* in local surroundings. It is defined by the proximity measure π_x which defines the locality around *x*. $\Omega(g)$ is the model complexity which should be low. The complexity will vary depending on the model. For example, in the case of a decision tree $\Omega(g)$ can be the depth of the tree. To maintain both interpretability and local fidelity, LIME needs to minimize $L(f, g, \pi_x)$ with low complexity $\Omega(g)$, so the model explanation is understandable by humans [42].

Algorithm 1 LIME with Sparse Linear Explanation

Require: classifier *f*, number of samples *N*
Require: instance *x* and its interpretable version x'
Require: similarity kernel π_x and length of explanation *K*
$Z \leftarrow \{\}$
for $i \in \{1,2,3, ..., N\}$ **do**
$z'_i \leftarrow$ sample around (x')
$Z \leftarrow Z \cup \langle z'_i, f(z_i), \pi_x(z_i)\rangle$
end for
$w \leftarrow$ *K*-Lasso (Z, K) with z'_i as features, $f(z)$ as target
return *w*

Algorithm 1 shows Lime with sparse linear explanation where *f* is the complex model for which an explanation is required, π_x is a weighting function that assigns higher importance (weight) to instances close to *x*, the sample being explained, and x' is its interpretable version (binary vector), which is understandable by humans. *N* is the number of modified (perturbed) instances for training the local model and *K* is the number of features selected for explanation. Samples z'_i are created around x'. z'_i is a modified (perturbed) version of x'. This results in *N* random samples z'_i that contain a fraction of the nonzero elements of x' generated around x'. These samples are input to *f* which assigns labels and outputs $f(z_i)$. These labeled samples are weighted using $\pi_x(z_i)$. Samples closer to x' are given higher weight. Since *f* is the original model that operates on the original feature space, each modified z'_i is converted back to the original format (e.g. text) to obtain z_i, which is the modified version of the original data *x*. The triplet $(z'_i, f(z_i), \pi_x(z_i))$ is stored in *Z*. *Z* is used to train the interpretable model using K-Lasso regression which selects *K* important features to generate a human interpretable explanation. Finally, the feature importance score (weight) *w* is returned for *x* that explains how *f* made its prediction [42].

3.2.1 Sub Modular Pick LIME

Sub Modular Pick LIME provides a global explanation of the model using the Sub Modular Pick (SP) approach. SP-LIME is a model-agnostic method that selects a diverse set of instances to provide non-redundant, global explanations of model behavior. It uses LIME to give an overall understanding of how the model works [43].

Algorithm 2 Sub Modular Pick LIME (SP-LIME)

Require: instances X, budget B
for all $x_i \in X$ **do**
$w_i \leftarrow$ **explain** (x_i, x'_i) using Algorithm 1
end for
for $j \in \{1 \ldots m\}$ **do**
$I_j \leftarrow \sqrt{\sum_{i=1}^{n} |w_{ij}|}$ compute feature importance
end for
$V \leftarrow \{\}$
while $|V| < B$ **do** greedy optimization of (4)
$V \leftarrow V \cup \text{argmax}_i\ c(V \cup \{i\}, W, I)$
end while
return V

Algorithm 2 shows SP-LIME which is used to provide model explanations. In this algorithm, B is a budget which is the number of instances the user wants to view. This can be interpreted as the user available time or patience. Local feature importance for the instances in X is represented by an $n \times m$ matrix W where n is the number of instances and m is the number of features. The local explanation for an instance x_i is generated using LIME (Algorithm 1) and placed in row w_i. The global importance of feature j is I_j, which shows how important this feature is across all instances. A higher value means a feature explains more instances. The global importance is given by $I_j = \sqrt{\sum_{i=1}^{n} |w_{ij}|}$ is the feature importance score (weight) assigned to feature j, e.g. i using the interpretable model g for the explanation of instance i. The features selected for explanation are obtained using a greedy algorithm. In each iteration, it adds the most important instances using the coverage function

$$c\,(V, W, I) \;=\; \sum_{j=1}^{P} \mathbb{1}_{[\exists\, i \in V:\, |w_{ij}|>0]} I_j \qquad (3)$$

which measures the global importance of a feature that appears at least once in the set V given the values of W and I. P is the number of features. $\mathbb{1}_{[\exists\, i \in V:\, |w_{ij}|>0]} I_j$ is an indicator function used to identify whether feature j is present in the explanation of instance i in the set V, and w_{ij} is the importance score of feature j in the explanation for instance i. If feature j appears in at least one instance i in V, then the feature contributes to the explanation. The set V is selected through the function

$$\underset{V,\, |V| \leq B}{\operatorname{argmax}}\, c(V, W, I) \qquad (4)$$

This maximizes the coverage function that is executed iteratively and adds the instances with the highest coverage.

Figure 2 presents an example of instance selection where n=5 and m=5, and W is a binary matrix (for simplicity). Instances are represented along the y-axis, and features are represented along the x-axis. The grey shaded cells indicate that a particular feature contributes to the explanation for that instance. The goal is to select a diverse set of instances that collectively cover the most important features without redundancy. The column corresponding to feature f2 is highlighted in red because it provides an explanation for four different instances. In comparison, feature f1 only provides an explanation of one instance. As a result, f2 is assigned a higher feature importance score than f1 and the other features, i.e. $I_2 > I_1$. Features f3 and f5 are equally important because they explain the same number of instances. The rows highlighted in green represent the selected instances. Instance 2 is selected because it includes information about features f2 and f3. Although instance 3 also includes these features, it was not selected to avoid redundancy. From instances 4 and 5, instance 5 is selected because it provides information about features f4 and f5. Instance 4 provides information on features f4 and f2, but since instance 2 has already covered feature f2, it is not selected. The selection of these two instances covers the maximum number of features without repetition.

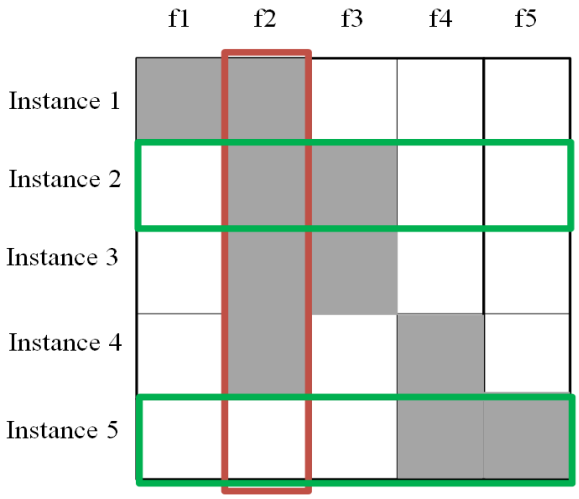

Figure 2: Five Instances and Five Features with Their Coverage [42]

Chapter 4 Experimental Setup

4.1 Polish Companies Dataset

This work uses the Polish companies dataset [11]. It covers five years with distinct data provided for each year from 2007 to 2013. It refers to companies that were active at the time, and the data collected covers 5 years before bankruptcy (if it occurred). This dataset is highly imbalanced and contains the 64 features given in Table 1. The terms used in this table are defined in Table 49. Companies are classified into two classes: class 1 is the negative class that indicates bankrupt companies, while class 0 is the positive class that indicates non-bankrupt companies. The dataset was originally in ARFF format and was converted to CSV format for this research. Table 2 presents the number of instances for each year. The five years combined dataset was created by merging the data from all five years. Four years combined datasets were created for testing by merging data from four years excluding one year for training. Three distinct four years combined datasets were created by excluding the first, third, and fifth years. The training was performed using data from these individual years, while the remaining four years of data was used for testing. Table 3 presents the number of instances in these datasets.

4.2 Data Preprocessing

The Polish companies dataset is imbalanced and has missing values that must be dealt with. Furthermore, data normalization should be done before data analysis. First, mean imputation was used to address the missing values. This widely used method involves replacing missing values with the mean for each feature. Then Min-Max normalization (scaling the data to the [0,1] range), was done to standardize the features. Next, dataset imbalance was corrected using SMOTE to balance the number of instances in the datasets. Tables 4 and 5 present the number of instances in these datasets. The Scikit-Learn and Pandas Python libraries were used for data preprocessing.

4.3 Dataset Splits

The five years combined datasets (balanced and imbalanced) were split into distinct training, validation, and testing sets, with 60% allocated for training, 20% for validation to refine the model, and the remaining 20% for testing. This means that for the imbalanced five years combined dataset, the test dataset contains 4.8% negative class instances and 95.2% positive class instances.

Table 1: The Features in the Polish Companies Dataset

Attribute	Description	Attribute	Description
X1	Net Profit / Total Assets	X33	Operating Expenses / Short-Term Liabilities
X2	Total Liabilities / Total Assets	X34	Operating Expenses / Total Liabilities
X3	Working Capital / Total Assets	X35	Profit On Sales / Total Assets
X4	Current Assets / Short-Term Liabilities	X36	Total Sales / Total Assets
X5	[(Cash + Short-Term Securities + Receivables – Short-Term Liabilities) / (Operating Expenses – Depreciation)] × 365	X37	(Current Assets - Inventories) / Long-Term Liabilities
X6	Retained Earnings / Total Assets	X38	Constant Capital / Total Assets
X7	EBIT / Total Assets	X39	Profit On Sales / Sales
X8	Book Value of Equity / Total Liabilities	X40	(Current Assets - Inventory - Receivables) / Short-Term Liabilities
X9	Sales / Total Assets	X41	Total Liabilities / ((Profit on Operating Activities + Depreciation) × (12/365))
X10	Equity / Total Assets	X42	Profit On Operating Activities / Sales
X11	(Gross Profit + Extraordinary Items + Financial Expenses) / Total Assets	X43	Rotation Receivables + Inventory Turnover in Days
X12	Gross Profit / Short-Term Liabilities	X44	(Receivables × 365) / Sales
X13	(Gross Profit + Depreciation) / Sales	X45	Net Profit / Inventory
X14	(Gross Profit + Interest) / Total Assets	X46	(Current Assets - Inventory) / Short-Term Liabilities
X15	(Total Liabilities × 365) / (Gross Profit + Depreciation)	X47	(Inventory × 365) / Cost of Products Sold
X16	(Gross Profit + Depreciation) / Total Liabilities	X48	EBITDA (Profit on Operating Activities - Depreciation) / Total Assets

X17	Total Assets / Total Liabilities	X49	EBITDA (Profit on Operating Activities - Depreciation) / Sales
X18	Gross Profit / Total Assets	X50	Current Assets / Total Liabilities
X19	Gross Profit / Sales	X51	Short-Term Liabilities / Total Assets
X20	(Inventory × 365) / Sales	X52	(Short-Term Liabilities × 365) / Cost of Products Sold)
X21	Sales (*N*) / Sales (*N*-1)	X53	Equity / Fixed Assets
X22	Profit on Operating Activities / Total Assets	X54	Constant Capital / Fixed Assets
X23	Net Profit / Sales	X55	Working Capital
X24	Gross Profit (In 3 Years) / Total Assets	X56	(Sales - Cost of Products Sold) / Sales
X25	(Equity - Share Capital) / Total Assets	X57	(Current Assets - Inventory - Short-Term Liabilities) / (Sales - Gross Profit - Depreciation)
X26	(Net Profit + Depreciation) / Total Liabilities	X58	Total Costs /Total Sales
X27	Profit On Operating Activities / Financial Expenses	X59	Long-Term Liabilities / Equity
X28	Working Capital / Fixed Assets	X60	Sales / Inventory
X29	Logarithm Of Total Assets	X61	Sales / Receivables
X30	(Total Liabilities - Cash) / Sales	X62	(Short-Term Liabilities × 365) / Sales
X31	(Gross Profit + Interest) / Sales	X63	Sales / Short-Term Liabilities
X32	(Current Liabilities × 365) / Cost of Products Sold	X64	Sales / Fixed Assets

Table 2: The Numbers of Instances in the Polish Companies Dataset

Year	Number of Instances	Number of Negative Instances (Class 1)	Number of Positive Instances (Class 0)
1	7027	271	6756
2	10173	400	9773
3	10503	495	10008
4	9792	515	9277
5	5910	410	5500

Table 3: The Numbers of Instances in the Additional Datasets

Dataset Name	Number of Instances	Number of Negative Instances (Class 1)	Number of Positive Instances (Class 0)
5 Years Combined	43405	2091	41314
4 Years Combined (excluding 1st year)	36378	1820	34558
4 Years Combined (excluding 3rd year)	32902	1596	31306
4 Years Combined (excluding 5th year)	37495	1681	35814

Table 4: The Numbers of Instances in the Balanced Polish Companies Dataset

Year	Number of Instances	Number of Negative Instances (Class 1)	Number of Positive Instances (Class 0)
1	13512	6756	6756
2	19547	9773	9773
3	20016	10008	10008
4	18554	9277	9277
5	11000	5500	5500

Table 5: The Numbers of Instances in the Balanced Additional Datasets

Dataset Name	Number of Instances	Number of Negative Instances (Class 1)	Number of Positive Instances (Class 0)
5 Years Combined	82628	41314	41314
4 Years Combined (excluding 1st year)	69116	34558	34558

4.4 Performance Metrics

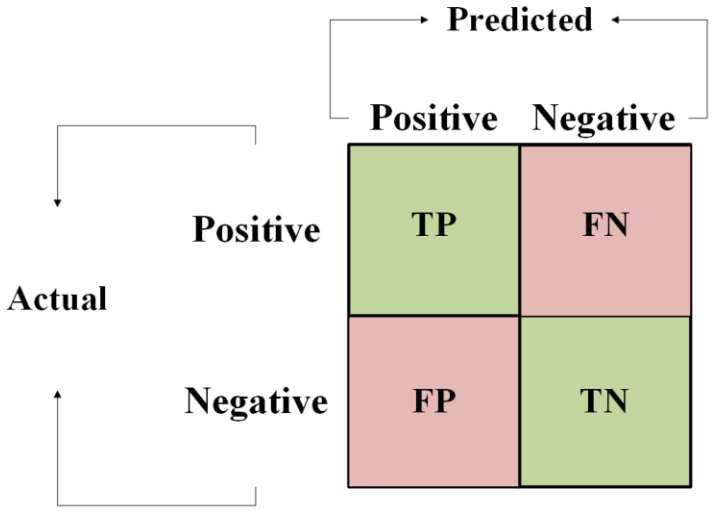

Figure 3: The Confusion Matrix

Figure 3 presents the confusion matrix which contains the four values True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). TP are those instances that are in the positive class and predicted as positive by the model. FP are those instances that are in the negative class but predicted as positive. TN are those instances that are in the negative class and predicted as negative. FN are those instances that are in the positive class but are predicted as negative. These values are used to compute the performance metrics.

The time to train an ML model is the training time. Accuracy is the proportion of instances correctly predicted by the model out of all instances and is given by

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \qquad (5)$$

Precision is the proportion of correctly predicted positive instances out of all instances predicted as positive and is given by

$$\text{Precision} = \frac{TP}{TP+FP} \qquad (6)$$

Negative class accuracy is a metric that measures the ability of a model to predict the negative class correctly. It is also known as specificity and is given by

$$\text{Negative Class Accuracy} = \frac{TN}{TN+FP} \qquad (7)$$

Chapter 5 Results

In this section, the test results are presented in the order in which they were obtained using Google Colab and Python with the default settings for the classifiers.

5.1 Classification with the 5 Years Combined Dataset

In the first step, 12 classifiers were evaluated with the imbalanced five years combined dataset and balanced five years combined dataset. These classifiers were chosen to provide a wide variety of classifier types. In this step, the dataset was split into training, validation, and testing subsets for the classification process. Table 6 presents the test results for the 12 classifiers obtained with the imbalanced five years combined dataset. CatBoost was the best performing classifier with an accuracy of 0.987 and precision of 0.986, with FN = 15 and FP = 97, and BDT was the second best with an accuracy of 0.982 and precision of 0.981, with FN = 36 and FP = 120. The LR classifier achieved the lowest accuracy of 0.785 and precision of 0.948. RF was the third best with an accuracy of 0.978 and precision of 0.977, with FN = 13 and FP = 177. The other classifiers had comparable performance with accuracy between 94% and 96%.

Table 6: Results with the Imbalanced Five Years Combined Dataset

Classifier Name	Accuracy	Precision	Confusion Matrix
CatBoost	0.987	0.986	[8248 15] [97 321]
RF	0.978	0.977	[8250 13] [177 241]
SVM	0.966	0.962	[8227 36] [257 161]
KNN	0.960	0.953	[8172 91] [258 160]
NB	0.957	0.958	[8066 197] [176 242]
LR	0.785	0.948	[6475 1788] [82 336]
BDT	0.982	0.981	[8227 36] [120 298]
Stacking Ensemble	0.948	0.922	[8209 54] [395 23]

AdaBoost with RF	0.968	0.966	[8249 14] [268 150]
NN	0.964	0.960	[8172 91] [220 198]
FT-Transformer	0.967	0.963	[8210 53] [236 182]
TabNet	0.966	0.961	[8208 55] [243 175]

Table 7: Classifier Negative Class Performance

Classifier Name	**Negative Class Accuracy**
CatBoost	0.768
RF	0.577
SVM	0.385
KNN	0.383
NB	0.579
LR	0.804
BDT	0.713
Stacking Ensemble	0.055
AdaBoost with RF	0.359
NN	0.474
FT-Transformer	0.435
TabNet	0.419

The goal of this research is to predict bankruptcy. Thus, Class 1 (the minority class) is more important because it represents bankrupt companies. Table 7 presents the negative class accuracy for the 12 classifiers. This shows that the LR classifier was best at predicting bankruptcies with an accuracy of 0.804. The CatBoost classifier was second with an accuracy of 0.768, followed by BDT with an accuracy of 0.713. The NB and RF classifiers also achieved above 50% accuracy, while the accuracy of other classifiers was below 50%. The Stacking Ensemble classifier performed the worst with an accuracy of only 0.055. Note that the LR classifier has the lowest

accuracy in Table 6, but it performs well in predicting the minority class as shown in Table 7. This is due to the low performance for the positive class. Conversely, the accuracy of the CatBoost and BDT classifiers was among the top in both Tables 6 and 7.

Table 8: Results with the Balanced Five Years Combined Dataset

Classifier Name	Accuracy	Precision	Confusion Matrix
CatBoost	0.995	0.997	[8195 68] [19 8244]
RF	0.989	0.994	[8120 143] [45 8218]
SVM	0.783	0.780	[6517 1746] [1841 6422]
KNN	0.954	0.998	[7553 710] [21 8242]
NB	0.763	0.787	[5964 2299] [1615 6648]
LR	0.794	0.798	[6516 1747] [1647 6616]
BDT	0.987	0.996	[8078 185] [32 8231]
Stacking Ensemble	0.977	0.980	[8052 211] [162 8101]
AdaBoost with RF	0.988	0.994	[8118 145] [52 8211]
NN	0.551	0.553	[4363 3900] [3523 4740]
FT-Transformer	0.889	0.968	[6639 1624] [217 8046]
TabNet	0.923	0.990	[7063 1200] [69 8194]

Table 8 presents the results for the 12 classifiers with the balanced five years combined dataset. This shows that the SVM, NN, and NB classifiers have the worst results with this dataset. The KNN, TabNet, and FT-Transformer classifiers have improved precision despite a decreased accuracy compared to the results with the imbalanced dataset. On the other hand, CatBoost, RF, BDT, Stacking Ensemble, and AdaBoost with RF show better results with the balanced dataset. The accuracy of the LR classifier is improved and the precision is decreased with the balanced dataset. Overall, CatBoost performs best with an accuracy of 0.995 and precision of 0.997, closely

followed by RF with an accuracy of 0.989 and precision of 0.994. Both classifiers showed good performance in detecting positive and negative classes. CatBoost had FN = 68 and FP = 19, while RF had FN = 143 and FP = 45. BDT achieved an accuracy of 0.987 and precision of 0.996, which is similar to RF and CatBoost. NN achieved an accuracy of 0.551 with the balanced dataset. SVM achieved an accuracy of 0.783, while NB had an accuracy of 0.763. The LR classifier had an accuracy of 0.794 and FT-Transformer an accuracy of 0.889. The performance of the other classifiers was between 92% and 97%. These results show that CatBoost is a highly effective classifier with both balanced and imbalanced datasets. Furthermore, data balancing slightly improved the performance of this classifier.

Based on the above results, seven classifiers were selected for the next step, namely CatBoost, RF, SVM, KNN, BDT, NB, and FT-Transformer. CatBoost, BDT, and FT-Transformer are all recent classifiers, and CatBoost, BDT, and RF are the best performers. NB was chosen as the benchmark, and SVM and KNN are well-known classifiers used in the literature.

5.2 Classification Using Different Training and Testing Datasets

In the next step, results are obtained for the seven classifiers with different training and testing datasets. Data from one year is used for training with the remaining four years of data used for testing.

5.2.1 First Year Dataset as Training Data

In this section, the first year dataset is used for training, while data from the other years is used for testing. First, the models were tested using balanced data for training and imbalanced data for testing. The balanced datasets contain equal numbers of instances for both classes as shown in Tables 4 and 5, and the results are presented in Tables 9-15. The imbalanced training and test datasets were then used with the classifiers and the results are presented in Tables 16-22. The training time for each classifier is also given.

CatBoost with the imbalanced training and test datasets had a high accuracy of 0.972 and precision of 0.976 with the second year test dataset, and a low accuracy of 0.946 and precision of 0.945. With the balanced training dataset, CatBoost had a high accuracy of 0.955 and precision of 0.966 with the fourth year test dataset and a low accuracy of 0.934 and precision of 0.983 with the second

year test dataset. These results show that performance decreases when a classifier is trained with the balanced training dataset. The NB classifier with the balanced training dataset had a high accuracy of 0.267 and precision of 0.962 with the fifth year test dataset. However, NB performs better with the imbalanced training dataset with a high accuracy of 0.951 and precision of 0.961 with the second year test dataset. The RF classifier with the imbalanced training dataset had a high accuracy of 0.961 and precision of 0.964 with the second year test dataset. RF performance decreases with the balanced training dataset with a high accuracy of 0.947 and precision of 0.958 with the fifth year test dataset. The BDT classifier with the imbalanced training dataset had a high accuracy of 0.955 and precision of 0.961 with the second year test dataset. With the balanced training dataset, BDT performance decreases with a high accuracy of 0.921 and precision of 0.963 with the fifth year test dataset. The other classifiers had similar performance.

Table 9: CatBoost Test Results Using the Balanced First Year Dataset for Training

CatBoost	**2nd Year**	**3rd Year**	**4th Year**	**5th Year**	**4 Years Combined**
Accuracy	0.934	0.948	0.955	0.950	0.946
Precision	0.983	0.977	0.966	0.957	0.972
Confusion Matrix	[9258 515] [159 241]	[9694 314] [227 268]	[9160 117] [322 193]	[5448 52] [242 168]	[33560 998] [950 870]

Table 10: RF Test Results Using the Balanced First Year Dataset for Training

RF	**2nd Year**	**3rd Year**	**4th Year**	**5th Year**	**4 Years Combined**
Accuracy	0.869	0.944	0.945	0.947	0.924
Precision	0.980	0.974	0.962	0.958	0.969
Confusion Matrix	[8621 1152] [176 224]	[9676 332] [261 234]	[9106 171] [365 150]	[5420 80] [236 174]	[32823 1735] [1038 782]

Table 11: SVM Test Results Using the Balanced First Year Dataset for Training

SVM	2nd Year	3rd Year	4th Year	5th Year	4 Years Combined
Accuracy	0.483	0.396	0.418	0.527	0.508
Precision	0.978	0.963	0.967	0.961	0.978
Confusion Matrix	[4616 5157] [106 294]	[3808 6200] [145 350]	[3707 5570] [128 387]	[2820 2680] [113 297]	[17065 17493] [390 1430]

Table 12: KNN Test Results Using the Balanced First Year Dataset for Training

KNN	2nd Year	3rd Year	4th Year	5th Year	4 Years Combined
Accuracy	0.632	0.630	0.616	0.678	0.634
Precision	0.975	0.969	0.963	0.959	0.967
Confusion Matrix	[6184 3589] [160 240]	[6322 3686] [204 291]	[5734 3543] [220 295]	[3758 1742] [160 250]	[21998 12560] [744 1076]

Table 13: NB Test Results Using the Balanced First Year Dataset for Training

NB	2nd Year	3rd Year	4th Year	5th Year	4 Years Combined
Accuracy	0.176	0.185	0.202	0.267	0.200
Precision	0.964	0.958	0.965	0.962	0.962
Confusion Matrix	[1447 8326] [54 346]	[1509 8499] [66 429]	[1521 7756] [55 460]	[1218 4282] [48 362]	[5695 28863] [223 1597]

Table 14: BDT Test Results Using the Balanced First Year Dataset for Training

BDT	2nd Year	3rd Year	4th Year	5th Year	4 Years Combined
Accuracy	0.712	0.886	0.914	0.921	0.851
Precision	0.982	0.977	0.962	0.963	0.971

Confusion Matrix	[6971 2802] [130 270]	[9025 983] [213 282]	[8779 498] [343 172]	[5233 267] [200 210]	[30008 4550] [886 934]

Table 15: FT-Transformer Test Results Using the Balanced First Year Dataset for Training

FT-Transformer	**2nd Year**	**3rd Year**	**4th Year**	**5th Year**	**4 Years Combined**
Accuracy	0.603	0.550	0.859	0.491	0.930
Precision	0.979	0.976	0.951	0.981	0.952
Confusion Matrix	[5860 3913] [125 275]	[5421 4587] [135 360]	[8332 945] [432 83]	[2541 2959] [48 362]	[33725 833] [1718 102]

Table 16: CatBoost Test Results Using the Imbalanced First Year Dataset for Training

CatBoost	**2nd Year**	**3rd Year**	**4th Year**	**5th Year**	**4 Years Combined**
Training Time (s)	7.64	9.35	5.33	6.69	5.58
Accuracy	0.972	0.966	0.960	0.946	0.963
Precision	0.976	0.969	0.963	0.945	0.966
Confusion Matrix	[9727 46] [240 160]	[9966 42] [317 178]	[9243 34] [354 161]	[5496 4] [318 92]	[34432 126] [1229 591]

Table 17: RF Test Results Using the Imbalanced First Year Dataset for Training

RF	**2nd Year**	**3rd Year**	**4th Year**	**5th Year**	**4 Years Combined**
Training Time (s)	6.23	4.81	4.79	5.68	5.78
Accuracy	0.961	0.952	0.948	0.931	0.950
Precision	0.964	0.955	0.951	0.932	0.953
Confusion Matrix	[9744 29] [367 33]	[9975 33] [470 25]	[9248 29] [477 38]	[5492 8] [402 8]	[34459 99] [1716 104]

Table 18: SVM Test Results Using the Imbalanced First Year Dataset for Training

SVM	2nd Year	3rd Year	4th Year	5th Year	4 Years Combined
Training Time (s)	4.20	2.96	2.90	2.89	3.39
Accuracy	0.937	0.932	0.926	0.917	0.929
Precision	0.961	0.953	0.949	0.934	0.951
Confusion Matrix	[9516 257] [383 17]	[9770 238] [478 17]	[9040 237] [489 26]	[5387 113] [378 32]	[33713 845] [1728 92]

Table 19: KNN Test Results Using the Imbalanced First Year Dataset for Training

KNN	2nd Year	3rd Year	4th Year	5th Year	4 Years Combined
Training Time (s)	0.005	0.005	0.007	0.005	0.003
Accuracy	0.931	0.922	0.915	0.916	0.922
Precision	0.961	0.954	0.949	0.935	0.952
Confusion Matrix	[9454 319] [382 18]	[9652 356] [464 31]	[8921 356] [476 39]	[5379 121] [376 34]	[33406 1152] [1698 122]

Table 20: NB Test Results Using the Imbalanced First Year Dataset for Training

NB	2nd Year	3rd Year	4th Year	5th Year	4 Years Combined
Training Time (s)	0.020	0.011	0.010	0.010	0.015
Accuracy	0.951	0.942	0.938	0.926	0.943
Precision	0.961	0.953	0.948	0.932	0.950
Confusion Matrix	[9662 111] [389 11]	[9885 123] [484 11]	[9178 99] [505 10]	[5461 39] [396 14]	[34277 281] [1789 31]

Table 21: BDT Test Results Using the Imbalanced First Year Dataset for Training

BDT	2nd Year	3rd Year	4th Year	5th Year	4 Years Combined
Training Time (s)	20.7	21.5	19.8	19.4	21.5
Accuracy	0.955	0.952	0.947	0.932	0.950
Precision	0.961	0.954	0.948	0.933	0.951
Confusion Matrix	[9705 68] [390 10]	[9987 21] [482 13]	[9263 14] [510 5]	[5497 3] [398 12]	[34514 44] [1789 31]

Table 22: FT-Transformer Test Results Using the Imbalanced First Year Dataset for Training

FT-Transformer	2nd Year	3rd Year	4th Year	5th Year	4 Years Combined
Training Time (s)	71.0	109.0	55.2	92.7	70.3
Accuracy	0.960	0.949	0.946	0.931	0.949
Precision	0.961	0.953	0.947	0.931	0.950
Confusion Matrix	[9767 6] [399 1]	[9965 43] [493 2]	[9265 12] [514 1]	[5498 2] [407 3]	[34501 57] [1808 12]

In Section 5.1, classifier performance was evaluated using both balanced and imbalanced datasets. The results indicate that some classifiers performed worse while some performed better when trained on the balanced five year combined dataset compared to the imbalanced five year combined dataset. With the balanced first year training dataset, the performance of all the classifiers decreased as compared to the imbalanced first year training dataset. These results suggest that data balancing negatively affects classifier performance. To verify this, additional experiments were conducted. In the first scenario, the number of negative class (minority) instances in the first year training dataset was increased to 1000, while in the second scenario, the number of instances was increased to 2000. Initially, the negative class contained 271 instances in the imbalanced first year dataset. The results for these two scenarios show that performance decreases as the number of minority class instances increases, further confirming that balancing the dataset leads to worse

results. For example, Table 23 presents the results for the CatBoost classifier trained on the balanced datasets containing 1,000 and 2,000 instances, as well as the imbalanced datasets using the combined four year and second year test datasets. Based on the results obtained, only the imbalanced datasets are considered in the remainder of the thesis.

Table 23: Results with Different Data Balancing Scenarios

CatBoost	**4 Years Combined**			**2nd Year**		
	Imbalanced	**Balanced with 1000 Instances**	**Balanced with 2000 Instances**	**Imbalanced**	**Balanced with 1000 Instances**	**Balanced with 2000 Instances**
Accuracy	0.963	0.952	0.945	0.972	0.944	0.923
Precision	0.966	0.970	0.971	0.976	0.981	0.984

When trained with the imbalanced dataset, Catboost achieved a high accuracy of 0.972 and precision of 0.976 with the second year test dataset. CatBoost achieved accuracies of 0.966, 0.960, and 0.963, and precisions of 0.969, 0.963, and 0.966 with the third, fourth, and four years combined test datasets, respectively. RF, BDT, and FT-Transformer had similar performance, and achieved a high accuracy and precision with the second year test dataset. RF achieved an accuracy of 0.961 and precision of 0.964, BDT achieved an accuracy of 0.955 and precision of 0.961, and FT-Transformer achieved an accuracy of 0.960 and precision of 0.961. RF with the third year test dataset achieved an accuracy of 0.952 and precision of 0.955, and with the four years combined test dataset, achieved an accuracy of 0.950 and precision of 0.953. RF achieved accuracies of 0.948 and 0.931, and precisions of 0.951 and 0.932 with the fourth and fifth year test datasets. BDT and FT-Transformer achieved accuracies of 0.952 and 0.949, and precisions of 0.954 and 0.953, with the third year test dataset. The BDT classifier achieved accuracies of 0.947 and 0.932, and precisions of 0.948 and 0.933, with the fourth and fifth year test datasets. FT-Transformer achieved an accuracy of 0.946 and precision of 0.947 with the fourth year test dataset and an accuracy and precision of 0.931 with the fifth year test dataset. BDT and FT-Transformer with the four years combined test dataset achieved accuracies of 0.950 and 0.949 and precisions of 0.951 and 0.950. SVM achieved an accuracy of 0.937 and precision of 0.961, KNN achieved an accuracy of 0.931 and precision of 0.961, and NB achieved an accuracy of 0.951 and precision of 0.961 with the second year test dataset. SVM with the third, fourth and fifth year test datasets achieved accuracies

of 0.932, 0.926, and 0.917 and precisions of 0.953, 0.949, and 0.934. SVM with the four years combined test dataset achieved an accuracy of 0.929 and precision of 0.951. KNN with the third, fourth, and fifth year test datasets achieved accuracies of 0.922, 0.915, and 0.916 and precisions of 0.954, 0.949, and 0.935. KNN with the four years combined test dataset achieved an accuracy of 0.922 and precision of 0.952. NB achieved accuracies of 0.942, 0.938, and 0.926 and precisions of 0.953, 0.948, and 0.932 with the third, fourth, and fifth year test datasets. NB with the four years combined test dataset achieved an accuracy of 0.943 and precision of 0.950.

The training time for the classifiers with the imbalanced dataset is given in Tables 16-22. FT-Transformer was the highest with a high training time of 109.0 s with the third year test dataset and a low training time of 55.2 s with the fourth year test dataset. The training time with the second, fifth, and four years combined test datasets was 71.0 s, 92.7 s, and 70.3 s, respectively. BDT was second with a high training time of 21.5 s with the third and four years combined test datasets, and a low training time of 19.4 s with the fifth year test dataset. The training time with the second and fourth year test datasets was 20.7 s and 19.8 s, respectively. The KNN and NB classifiers had the lowest training times among the seven classifiers. KNN had a high training time of 0.007 s with the fourth year test dataset and a low training time of 0.003 s with the four years combined test dataset, while with the other test datasets the training times were 0.005 s. NB had a high training time of 0.020 s with the second year test dataset and a low training time of 0.010 s with the fourth and fifth year test datasets. NB had a training time of 0.011 s and 0.015 s with the third and four years combined test datasets, respectively. SVM had a high training time of 4.20 s with the second year test dataset and a low training time of 2.89 s with the fifth year test dataset. RF and CatBoost had high training times of 6.23 s and 7.64 s with the second year test dataset, respectively, and low training times of 4.79 s and 5.33 s with the fourth year test dataset.

5.2.2 Third Year Dataset as Training Data

In this section, the third year dataset is used for training, while data from the other years is used for testing. The third year data is excluded from the four years combined test dataset. The results obtained are given in Tables 24-30. The results for the CatBoost and RF classifiers are given in Tables 24 and 25, respectively. CatBoost had a high accuracy of 0.981 and precisions of 0.985 and 0.983 with the first and second year test datasets, respectively. With the fourth, fifth and four years combined test datasets CatBoost achieved accuracies of 0.969, 0.961 and 0.974 and precisions of

0.971, 0.961 and 0.976. RF achieved accuracies of 0.976 and 0.972 with the first and second year test datasets and precisions of 0.978 and 0.973. RF with the fourth year test dataset achieved an accuracy and precision of 0.955. RF with the four years combined and fifth year test datasets achieved accuracies of 0.962 and 0.935 and precisions of 0.963 and 0.936. The results for the SVM and KNN classifiers are given in Tables 26 and 27. SVM had a high accuracy of 0.937 and precision of 0.962 with the first year test dataset and a low accuracy of 0.915 and precision of 0.937 with the fifth year test dataset. SVM with the second, fourth, and four years combined test datasets achieved accuracies of 0.927, 0.920, and 0.925, and precisions of 0.961, 0.949, and 0.953, respectively. KNN had a high accuracy of 0.955 and precision of 0.962 with the first year test dataset and a low accuracy of 0.925 and precision of 0.933 with the fifth year test dataset. KNN achieved accuracies of 0.952, 0.939 and 0.943 and precisions of 0.962, 0.950, and 0.953 with the second, fourth, and four years combined test datasets, respectively.

Table 24: CatBoost Test Results Using the Third Year Dataset for Training

CatBoost	**1st Year**	**2nd Year**	**4th Year**	**5th Year**	**4 Years Combined**
Training Time (s)	7.51	7.92	9.49	8.03	9.47
Accuracy	0.981	0.981	0.969	0.961	0.974
Precision	0.985	0.983	0.971	0.961	0.976
Confusion Matrix	[6723 33] [101 170]	[9747 26] [169 231]	[9253 24] [275 240]	[5496 4] [226 184]	[31219 87] [771 825]

Table 25: RF Test Results Using the Third Year Dataset for Training

RF	**1st Year**	**2nd Year**	**4th Year**	**5th Year**	**4 Years Combined**
Training Time (s)	7.94	8.07	7.96	8.53	10.4
Accuracy	0.976	0.972	0.955	0.935	0.962
Precision	0.978	0.973	0.955	0.936	0.963
Confusion Matrix	[6739 17] [155 116]	[9764 9] [276 124]	[9273 4] [434 81]	[5491 9] [373 37]	[31275 31] [1205 391]

Table 26: SVM Test Results Using the Third Year Dataset for Training

SVM	1st Year	2nd Year	4th Year	5th Year	4 Years Combined
Training Time (s)	7.53	8.23	7.86	9.08	9.11
Accuracy	0.937	0.927	0.920	0.915	0.925
Precision	0.962	0.961	0.949	0.937	0.953
Confusion Matrix	[6573 183] [259 12]	[9411 362] [381 19]	[8976 301] [480 35]	[5356 144] [361 49]	[30316 990] [1481 115]

Table 27: KNN Test Results Using the Third Year Dataset for Training

KNN	1st Year	2nd Year	4th Year	5th Year	4 Years Combined
Training Time (s)	0.007	0.006	0.006	0.006	0.010
Accuracy	0.955	0.952	0.939	0.925	0.943
Precision	0.962	0.962	0.950	0.933	0.953
Confusion Matrix	[6698 58] [261 10]	[9674 99] [386 14]	[9169 108] [486 29]	[5451 49] [393 17]	[30976 330] [1526 70]

Table 28: NB Test Results Using the Third Year Dataset for Training

NB	1st Year	2nd Year	4th Year	5th Year	4 Years Combined
Training Time (s)	0.011	0.011	0.010	0.014	0.033
Accuracy	0.954	0.957	0.942	0.929	0.947
Precision	0.962	0.961	0.947	0.931	0.952
Confusion Matrix	[6695 61] [261 10]	[9734 39] [397 3]	[9221 56] [511 4]	[5486 14] [404 6]	[31151 155] [1583 13]

The NB classifier results are given in Table 28. NB had a high accuracy of 0.957 and precision of 0.961 with the second year test dataset, and a low accuracy of 0.929 and precision of 0.931 with

the fifth year test dataset. NB achieved accuracies of 0.954, 0.942, and 0.947 and precisions of 0.962, 0.947, and 0.952 with the first, fourth, and four years combined test datasets. Tables 29 and 30 present the results for the BDT and FT-Transformer classifiers. The BDT classifier achieved accuracies of 0.976 and 0.971 and precisions of 0.976 and 0.970, with the first and second year test datasets. BDT with the fourth, fifth, and four years combined test datasets achieved accuracies of 0.960, 0.931, and 0.961, and precisions of 0.959, 0.930, and 0.960. The FT-Transformer classifier achieved accuracies of 0.959 and 0.960, and precision of 0.961 with the first and second year test datasets. FT-Transformer had a low accuracy of 0.931 and precision of 0.934 with the fifth year test dataset. FT-Transformer with the fourth and four years combined test datasets, achieved accuracies of 0.947 and 0.950, and precisions of 0.948 and 0.952.

Table 29: BDT Test Results Using the Third Year Dataset for Training

BDT	**1st Year**	**2nd Year**	**4th Year**	**5th Year**	**4 Years Combined**
Training Time (s)	43.7	17.9	44.0	39.8	34.8
Accuracy	0.976	0.971	0.960	0.931	0.961
Precision	0.976	0.970	0.959	0.930	0.960
Confusion Matrix	[6751 5] [177 104]	[9768 5] [290 110]	[9274 3] [392 123]	[5499 1] [406 4]	[31283 23] [1255 341]

Table 30: FT-Transformer Test Results Using the Third Year Dataset for Training

FT-Transformer	**1st Year**	**2nd Year**	**4th Year**	**5th Year**	**4 Years Combined**
Training Time (s)	44.7	80.2	44.8	43.1	127
Accuracy	0.959	0.960	0.947	0.931	0.950
Precision	0.961	0.961	0.948	0.934	0.952
Confusion Matrix	[6740 16] [270 1]	[9770 3] [399 1]	[9275 2] [513 2]	[5487 13] [390 20]	[31251 55] [1593 3]

The FT-Transformer and BDT classifiers again had the highest training times. The FT-Transformer highest training time was 127 s with the four years combined test dataset, and lowest training time was 43.1 s with the fifth year test dataset. The BDT highest training time was 44.0 s with the fourth year test dataset, and lowest training time was 17.9 s with the second year test dataset. KNN and NB again had the lowest training times. The NB highest training time was 0.033 s with the four years combined test dataset and lowest training time was 0.010 s with the fourth year test dataset. The KNN highest training time was 0.010 s with the four years combined test dataset and lowest training time was 0.006 s with the second, fourth, and fifth year test datasets. CatBoost, RF, and SVM had similar training times. The CatBoost highest training time was 9.49 s with the fourth year test dataset, and lowest training time was 7.51 s with the first year test dataset. The RF highest training time was 10.4 s with the four years combined test dataset, and lowest training time was 7.94 s with the first year test dataset. The SVM highest training time was 9.11 s with the four years combined test dataset, and lowest training time was 7.53 s with the first year test dataset.

5.2.3 Fifth Year Dataset as Training Data

In this section, the fifth year dataset is used for training, while data from the other years is used for testing. The results obtained are given in Tables 31-37. Tables 31 and 32 present the results for the CatBoost and RF classifiers. CatBoost and RF achieved high accuracies of 0.956 and 0.963, and precisions of 0.985 and 0.969 with the first year test dataset. The lowest accuracies for CatBoost and RF were 0.901 and 0.935, and precisions of 0.981 and 0.969, respectively, with the second year test dataset. For the third, fourth, and four years combined test datasets, CatBoost achieved accuracies of 0.946, 0.953 and 0.937 and precisions of 0.978, 0.972 and 0.979, and RF achieved accuracies of 0.951, 0.950 and 0.949 and precisions of 0.964, 0.959 and 0.965. The results for the SVM and KNN classifiers are given in Tables 33 and 34. SVM and KNN achieved high accuracies of 0.942 and 0.952 and precisions of 0.962 and 0.963 with the first year test dataset. SVM with the second, third, and fourth year test datasets achieved accuracies of 0.923, 0.918 and 0.914, with precisions of 0.961, 0.953 and 0.950. With the four years combined test dataset, SVM achieved an accuracy of 0.923 and precision of 0.956. The KNN classifier achieved accuracies of 0.942, 0.950 and 0.945 and precisions of 0.962, 0.954 and 0.949 with the second, third, and fourth year test

datasets. KNN achieved an accuracy of 0.937 and precision of 0.958 with the four years combined test dataset.

Table 31: CatBoost Test Results Using the Fifth Year Dataset for Training

CatBoost	**1st Year**	**2nd Year**	**3rd Year**	**4th Year**	**4 Years Combined**
Training Time (s)	6.44	7.17	5.18	7.20	5.28
Accuracy	0.956	0.901	0.946	0.953	0.937
Precision	0.985	0.981	0.978	0.972	0.979
Confusion Matrix	[6546 210] [99 172]	[8938 835] [168 232]	[9650 358] [214 281]	[9084 193] [264 251]	[34218 1596] [745 936]

Table 32: RF Test Results Using the Fifth Year Dataset for Training

RF	**1st Year**	**2nd Year**	**3rd Year**	**4th Year**	**4 Years Combined**
Training Time (s)	4.92	6.30	4.88	5.48	4.63
Accuracy	0.963	0.935	0.951	0.950	0.949
Precision	0.969	0.969	0.964	0.959	0.965
Confusion Matrix	[6710 46] [214 57]	[9418 355] [304 96]	[9864 144] [367 128]	[9181 96] [396 119]	[35173 641] [1281 400]

Table 33: SVM Test Results Using the Fifth Year Dataset for Training

SVM	**1st Year**	**2nd Year**	**3rd Year**	**4th Year**	**4 Years Combined**
Training Time (s)	4.13	3.53	4.03	3.81	2.60
Accuracy	0.942	0.923	0.918	0.914	0.923
Precision	0.962	0.961	0.953	0.950	0.956
Confusion Matrix	[6611 145] [262 9]	[9365 408] [371 29]	[9617 391] [471 24]	[8908 369] [470 45]	[34501 1313] [1574 107]

Table 34: KNN Test Results Using the Fifth Year Dataset for Training

KNN	1st Year	2nd Year	3rd Year	4th Year	4 Years Combined
Training Time (s)	0.002	0.002	0.002	0.002	0.003
Accuracy	0.952	0.942	0.950	0.945	0.937
Precision	0.963	0.962	0.954	0.949	0.958
Confusion Matrix	[6671 85] [255 16]	[9564 209] [373 27]	[9962 46] [484 11]	[9241 36] [500 15]	[34992 822] [1532 149]

Table 35: NB Test Results Using the Fifth Year Dataset for Training

NB	1st Year	2nd Year	3rd Year	4th Year	4 Years Combined
Training Time (s)	0.006	0.006	0.006	0.006	0.008
Accuracy	0.951	0.934	0.927	0.922	0.932
Precision	0.963	0.961	0.955	0.951	0.957
Confusion Matrix	[6666 90] [257 14]	[9483 290] [380 20]	[9694 314] [453 42]	[8973 304] [464 51]	[34816 998] [1554 127]

Table 36: BDT Test Results Using the Fifth Year Dataset for Training

BDT	1st Year	2nd Year	3rd Year	4th Year	4 Years Combined
Training Time (s)	19.2	17.6	19.1	17.8	17.8
Accuracy	0.972	0.969	0.961	0.958	0.965
Precision	0.971	0.969	0.964	0.961	0.966
Confusion Matrix	[6751 5] [192 79]	[9770 3] [309 91]	[9974 34] [376 119]	[9243 34] [379 136]	[35738 76] [1256 425]

Table 37: FT-Transformer Test Results Using the Fifth Year Dataset for Training

FT-Transformer	1st Year	2nd Year	3rd Year	4th Year	4 Years Combined
Training Time (s)	45.5	36.1	88.3	84.2	33.2
Accuracy	0.942	0.955	0.951	0.928	0.954
Precision	0.964	0.961	0.953	0.950	0.955
Confusion Matrix	[6591 165] [244 27]	[9712 61] [394 6]	[9992 16] [494 1]	[9051 226] [475 40]	[35759 55] [1677 4]

Table 35 presents the results for the NB classifier. A high accuracy of 0.951 and precision of 0.963, were achieved with the first year test dataset. NB with the second, third, and fourth year test datasets achieved accuracies of 0.934, 0.927, and 0.922, and precisions of 0.961, 0.955, and 0.951. NB with the four years combined test dataset achieved an accuracy of 0.932 and precision of 0.957. Table 36 presents the results for the BDT classifier. A high accuracy of 0.972 and precision of 0.971 were obtained with the first year test dataset. BDT achieved an accuracy and precision of 0.969 with the second year test dataset. BDT with the third, fourth, and four years combined test datasets achieved accuracies of 0.961, 0.958, and 0.965, and precisions of 0.964, 0.961, and 0.966. Table 37 presents the results for the FT-Transformer classifier. A high accuracy of 0.955 and precision of 0.961, and a low accuracy of 0.928 and precision of 0.950, were achieved with the second and fourth year test datasets, respectively. FT-Transformer with the first year test dataset achieved an accuracy of 0.942 and precision of 0.964, and with the third and four years combined test datasets, accuracies of 0.951 and 0.954 and precisions of 0.953 and 0.955.

The FT-Transformer and BDT classifiers again had the highest training times. The FT-Transformer highest training time was 88.3 s with the third year test dataset, and lowest training time was 33.2 s with the four years combined test dataset. The BDT highest training time was 19.2 s with the first year test dataset, and lowest training time was 17.6 s with the second year test dataset. The KNN and NB classifiers again had the lowest training times. The KNN training times were between 0.003 s and 0.002 s with the test datasets. The NB highest training time was 0.008 s with the four years combined test dataset and lowest training time was 0.006 s with the other test datasets. The SVM, RF, and CatBoost classifiers had similar training times. The CatBoost highest

training time was 7.17 s and the RF highest training time was 6.30 s with the second year test dataset. The SVM highest training time was 4.13 s with the first year test dataset.

Table 38 gives the accuracy for each classifier obtained with the four years combined datasets in Section 5.2. CatBoost had a high accuracy of 0.974 and RF had a high accuracy of 0.962 when trained with the third year dataset, BDT had a high accuracy of 0.965 when trained with the fifth year dataset. SVM had a high accuracy of 0.929 when trained with the first year dataset. KNN and NB had high accuracies of 0.943 and 0.947 when trained with the third year dataset. FT-Transformer had a high accuracy of 0.954 with the fifth year dataset. The results show that CatBoost, RF, and BDT performed better than the other classifiers.

The individual year test results with the seven classifiers were used to choose three classifiers for the next step. The selection of these classifiers was based on accuracy, precision, and training time. The NB classifier was selected as the benchmark, whereas CatBoost and RF were selected as the best performers. CatBoost consistently outperformed the other classifiers with slightly worse performance when trained with the fifth year dataset. It had a high accuracy of 0.981 and precision of 0.985 with the first year test dataset when trained with the third year dataset. Aside from this, CatBoost had training times ranging between 5.18 s and 9.49 s. RF and BDT had similar performance, but the training time for RF was lower, so RF was selected.

Table 38: Accuracy with the 4 Years Combined Datasets

Classifier	**1st Year Training**	**3rd Year Training**	**5th Year Training**
CatBoost	0.963	0.974	0.937
RF	0.950	0.962	0.949
SVM	0.929	0.925	0.923
KNN	0.922	0.943	0.937
NB	0.943	0.947	0.932
BDT	0.950	0.961	0.965
FT-Transformer	0.949	0.950	0.954

5.3 Model Explanation Using SP-LIME

In this section, the CatBoost, RF, and NB classifiers are used with SP-LIME. Each classifier was separately run with SP-LIME to identify the important features that affect the model results using the imbalanced five year combined dataset. SP-LIME was configured to select three instances and the corresponding top 20 features were obtained. The feature selection process in SP-LIME is random, so each instance yields a different set of top features. Table 39 presents the top 20 features for CatBoost and the importance scores for the three instances. This shows that features 46, 27, 5, 32, 35, 13, 34, 48, 26, and 21 appear for all three instances. The top three features based on importance scores were 46, 27, and 5. These scores indicate how much a feature affects model prediction.

Figures 4, 5, and 6 show the importance score ranges of the top 20 features for the three instances. A red bar means the feature contributes to the classifier outcome towards class 0, while a green bar means it contributes to the classifier outcome towards class 1. Since negative values simply reflect the influence toward class 0, only the magnitudes are given in the tables. Figure 4 shows that features 46 (Attr46) and 27 (Attr27) contribute most to the classifier outcome towards class 1, while feature 5 (Attr5) contributes most to the classifier outcome towards class 0, so they have the greatest impact on the classifier outcome.

Table 39: SP-LIME Top 20 Features Using CatBoost

Instance 1			**Instance 2**		**Instance 3**	
Rank	**Feature No**	**Importance Score**	**Feature No**	**Importance Score**	**Feature No**	**Importance Score**
1	46	0.284	5	0.149	46	0.147
2	27	0.269	34	0.129	27	0.141
3	5	0.186	46	0.100	24	0.096
4	32	0.115	39	0.069	58	0.071
5	39	0.113	24	0.055	5	0.062
6	35	0.100	27	0.053	13	0.056
7	13	0.082	32	0.045	3	0.055
8	34	0.071	21	0.041	32	0.035
9	58	0.059	30	0.041	9	0.034

10	19	0.055	11	0.040	59	0.033
11	52	0.045	26	0.038	30	0.031
12	10	0.043	25	0.036	19	0.031
13	57	0.040	35	0.034	21	0.029
14	26	0.037	29	0.034	34	0.029
15	48	0.035	13	0.033	48	0.024
16	36	0.035	48	0.031	26	0.023
17	21	0.033	10	0.031	40	0.021
18	38	0.031	40	0.031	25	0.020
19	9	0.029	44	0.030	35	0.018
20	11	0.027	38	0.028	28	0.018

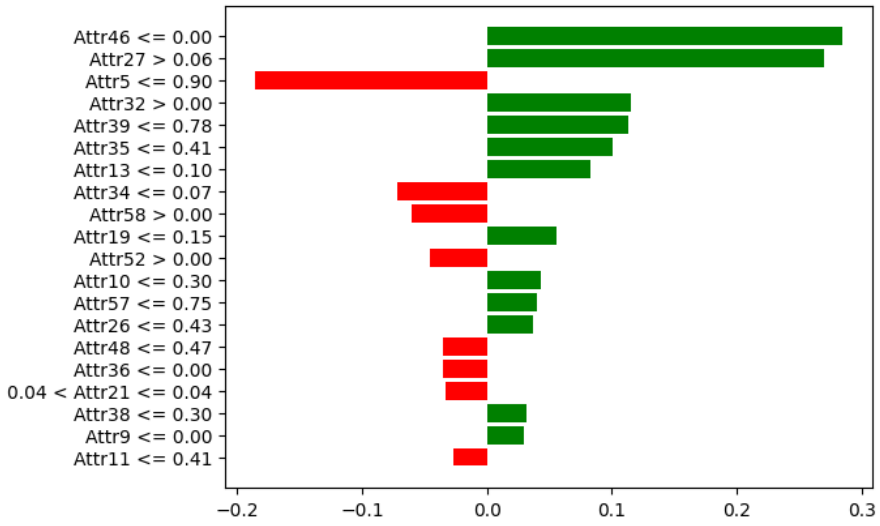

Figure 4: CatBoost First Instance Top Features

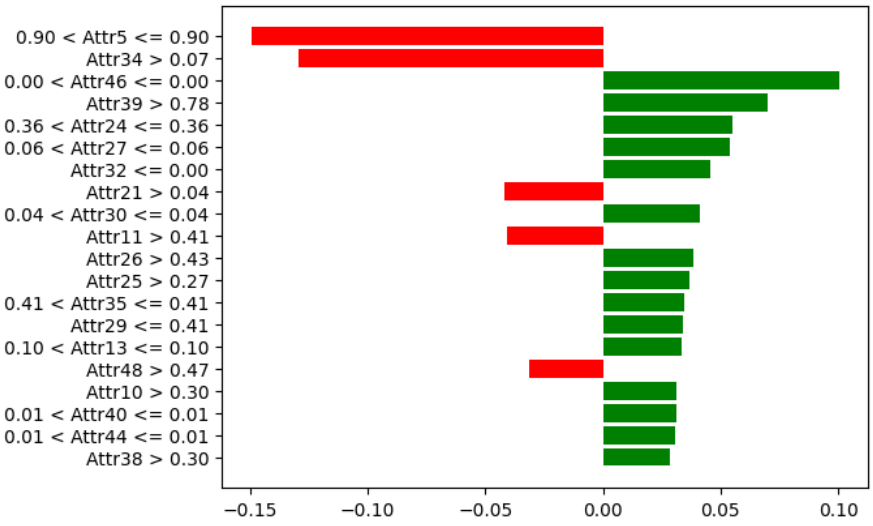

Figure 5: CatBoost Second Instance Top Features

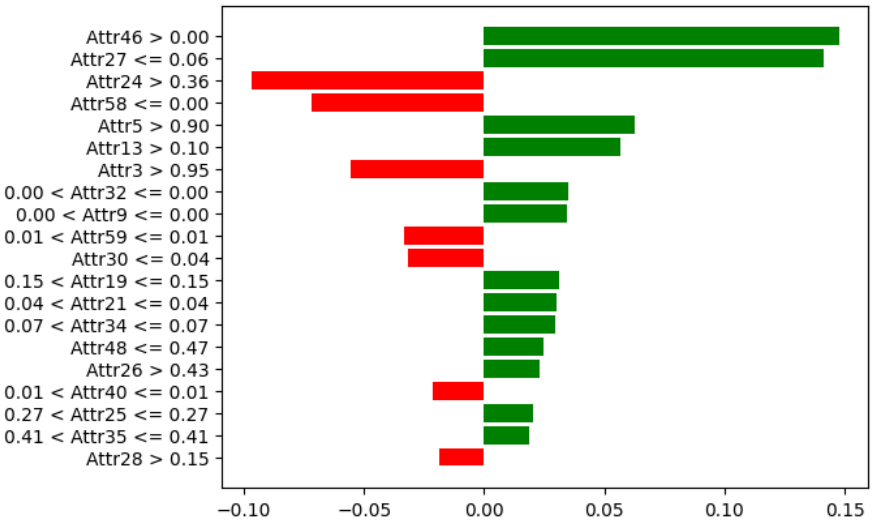

Figure 6: CatBoost Third Instance Top Features

Table 40: SP-LIME Top 20 Features Using RF

Instance 1			**Instance 2**		**Instance 3**	
Rank	**Feature No**	**Importance Score**	**Feature No**	**Importance Score**	**Feature No**	**Importance Score**
1	35	0.050	27	0.046	24	0.017
2	38	0.031	42	0.022	27	0.016
3	39	0.027	22	0.021	38	0.016

4	46	0.019	35	0.016	35	0.014
5	42	0.015	15	0.013	25	0.014
6	22	0.015	11	0.012	49	0.013
7	27	0.014	34	0.011	6	0.009
8	20	0.012	39	0.010	48	0.008
9	6	0.012	37	0.010	23	0.008
10	56	0.012	24	0.010	10	0.008
11	36	0.012	25	0.010	36	0.008
12	2	0.012	46	0.009	11	0.008
13	15	0.011	6	0.009	39	0.007
14	58	0.011	36	0.009	9	0.007
15	34	0.010	49	0.008	2	0.007
16	26	0.009	23	0.007	41	0.006
17	62	0.008	41	0.007	46	0.006
18	24	0.008	38	0.007	4	0.006
19	54	0.007	10	0.007	54	0.006
20	13	0.006	61	0.007	45	0.006

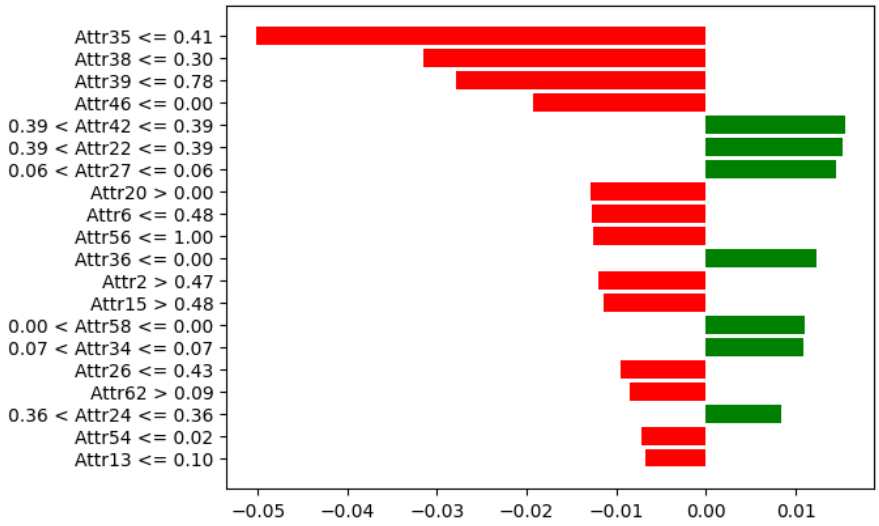

Figure 7: RF First Instance Top Features

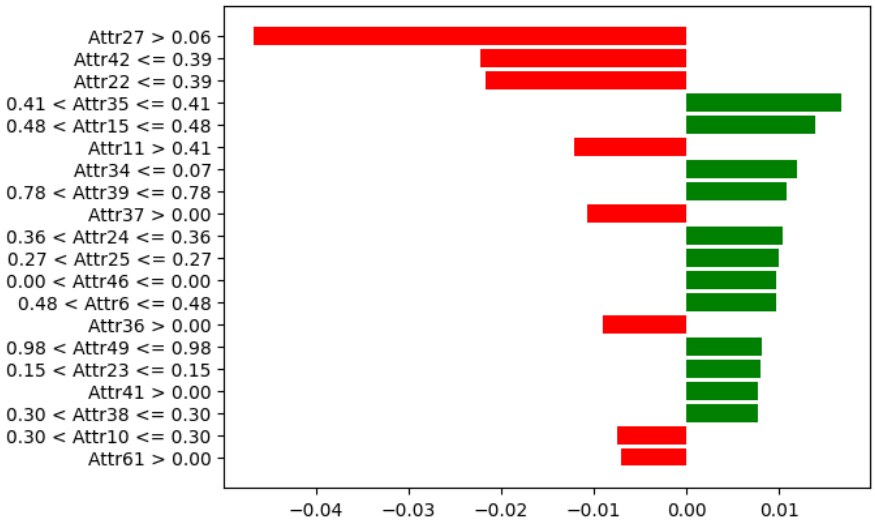

Figure 8: RF Second Instance Top Features

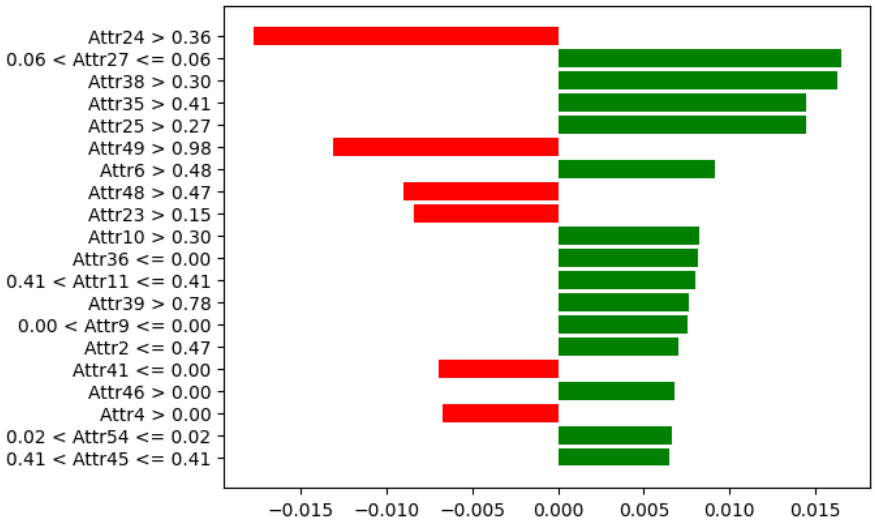

Figure 9: RF Third Instance Top Features

Table 40 presents the top 20 features for the RF classifier and the importance scores for the three instances. This shows that features 35, 38, 39, 46, 27, 6, 36, and 24 appear for all three instances. The top three features based on importance scores were 35, 27, and 38, so they have the greatest impact on the classifier outcome. Figure 7 shows that features 35 (Attr35), 38 (Attr38), and 39 (Attr39) contribute to the classifier outcome toward class 0. Table 41 presents the top 20 features for the NB classifier and their importance scores for the three instances. This shows that features 50, 17, 4, 60, 51, 29, 36, 37, 52, 63, 47, and 8 appear for all three instances. Features 51, 17, and 4 are the three top features based on their importance scores, so they have the greatest impact on

the classifier outcome. Figure 10 shows that features 50 (Attr50), 17 (Attr17), and 4 (Attr4) contribute to the classifier outcome toward class 0.

Table 41: SP-LIME Top Features Using NB

Instance 1			**Instance 2**		**Instance 3**	
Rank	**Feature No**	**Importance Score**	**Feature No**	**Importance Score**	**Feature No**	**Importance Score**
1	50	0.00087	51	0.00114	4	0.00083
2	17	0.00083	52	0.00043	51	0.00046
3	4	0.00075	17	0.00029	47	0.00042
4	60	0.00056	50	0.00027	17	0.00033
5	51	0.00042	4	0.00024	29	0.00033
6	29	0.00032	60	0.00019	50	0.00023
7	36	0.00032	36	0.00016	60	0.00018
8	37	0.00021	47	0.00015	52	0.00017
9	52	0.00019	32	0.00013	36	0.00015
10	63	0.00019	29	0.00009	39	0.00010
11	33	0.00018	29	0.00009	39	0.00010
12	61	0.00017	12	0.00008	63	0.00009
13	9	0.00017	63	0.00008	8	0.00008
14	47	0.00014	9	0.00007	55	0.00008
15	19	0.00011	19	0.00006	37	0.00007
16	8	0.00010	34	0.00006	53	0.00006
17	10	0.00009	37	0.00006	45	0.00006
18	46	0.00009	46	0.00006	20	0.00005
19	21	0.00007	23	0.00006	64	0.00005
20	3	0.00007	8	0.00005	21	0.00005

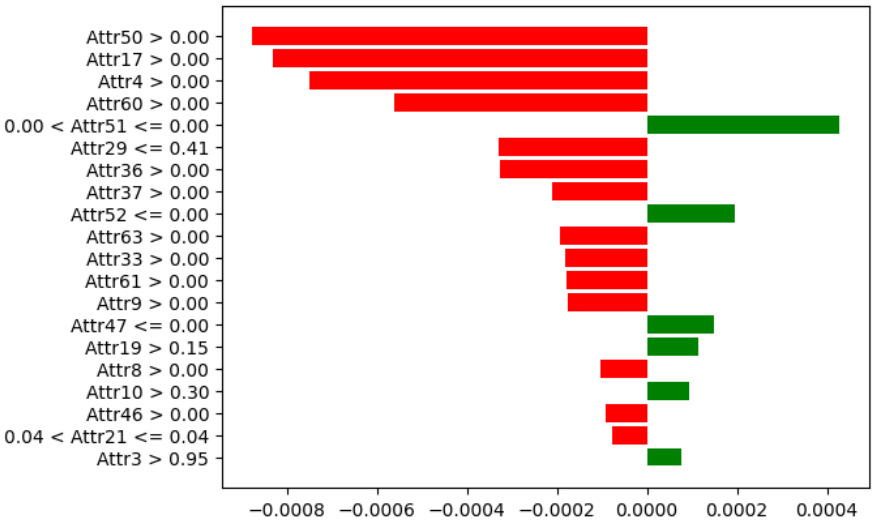

Figure 10: NB First Instance Top Features

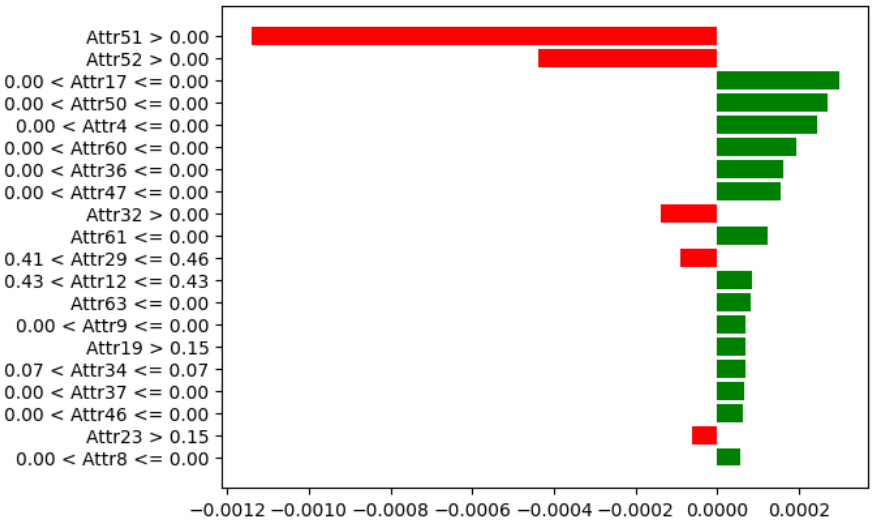

Figure 11: NB Second Instance Top Features

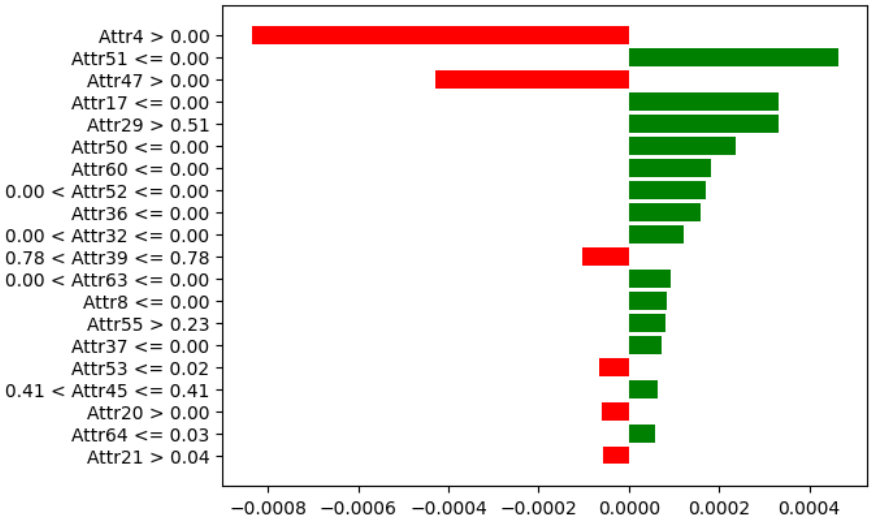

Figure 12: NB Third Instance Top Features

Table 42: Top 22 Features for Three Classifiers

Top Features					
Rank	**Feature No**	**Count**	**Rank**	**Feature No**	**Count**
1	46	8	12	38	5
2	36	7	13	4	4
3	27	6	14	11	4
4	39	6	15	13	4
5	34	6	16	19	4
6	35	6	17	25	4
7	9	5	18	26	4
8	10	5	19	29	4
9	21	5	20	37	4
10	24	5	21	48	4
11	32	5	22	52	4

Table 43: PCA Feature Ranking

Rank	**Feature**	**Variance**	**Rank**	**Feature**	**Variance**
1	29	0.720	33	1	0.003
2	51	0.015	34	63	0.003

3	3	0.013	35	61	0.003
4	17	0.010	36	34	0.003
5	8	0.010	37	30	0.003
6	38	0.010	38	62	0.003
7	10	0.010	39	59	0.003
8	55	0.009	40	18	0.002
9	64	0.006	41	49	0.002
10	27	0.006	42	7	0.002
11	52	0.006	43	14	0.002
12	25	0.005	44	15	0.002
13	40	0.005	45	44	0.002
14	6	0.005	46	43	0.002
15	21	0.005	47	58	0.002
16	54	0.005	48	20	0.002
17	60	0.005	49	56	0.002
18	53	0.005	50	41	0.002
19	36	0.004	51	16	0.002
20	9	0.004	52	5	0.002
21	2	0.004	53	22	0.002
22	57	0.004	54	11	0.002
23	24	0.004	55	12	0.002
24	28	0.004	56	26	0.002
25	13	0.004	57	31	0.002
26	50	0.003	58	35	0.002
27	37	0.003	59	19	0.002
28	4	0.003	60	42	0.002
29	46	0.003	61	23	0.002
30	32	0.003	62	48	0.002
31	47	0.003	63	39	0.001
32	33	0.003	64	45	0.001

From the SP-LIME results in Tables 39, 40, and 41, it was found that several features consistently appeared in the top features lists of the classifiers. Based on these results, the top features were selected. Each classifier provides three instances, so with three classifiers, a feature can appear up to 9 times. Table 42 presents the top 22 features with the number of times they appear. The first six features appear at least six times, the first twelve features appear at least five times, and the remaining ten features appear at least four times. For comparison, PCA was applied to rank the features, and the results are given in Table 43. The top six features identified by SP-LIME and PCA are completely different, with no common features. Among the top 12 features, only features 10 and 38 are common to both. In the top 22 features, five features are common, namely 10, 38, 25, 29, and 52. The three classifiers were used with the top 6, 12, and 22 features, from SP-LIME and PCA separately, using the combined five year imbalanced dataset with the data splitting mentioned in Section 4.3.

The results for the top 6 features from SP-LIME and PCA are presented in Table 44. The RF classifier was the best performer using the SP-LIME top 6 features with an accuracy of 0.895 and precision of 0.981, and the training time was 5.06 s. CatBoost had an accuracy of 0.890 and precision of 0.981, and the training time was 3.64 s. NB had the worst performance with 0.782 accuracy and precision of 0.987, and the training time was 0.009 s. CatBoost was the best performer using the PCA top 6 features, with an accuracy of 0.850 and precision of 0.962, and the training time was 5.39 s. RF had an accuracy of 0.772 and precision of 0.967, and the training time was 7.16 s. NB had an accuracy of 0.822 and precision of 0.979, and the training time was 0.012 s. The NB classifier performed better with the PCA top 6 features, while CatBoost and RF performed better with the SP-LIME top 6 features.

Table 44: Performance Using the Top 6 Features

	SP-LIME Top 6 Features			**PCA Top 6 Features**		
	CatBoost	**RF**	**NB**	**CatBoost**	**RF**	**NB**
Accuracy	0.890	0.895	0.782	0.850	0.772	0.822
Precision	0.981	0.981	0.987	0.962	0.967	0.979
Confusion Matrix	[7448 815] [141 277]	[7496 767] [146 272]	[6450 1813] [82 336]	[7250 1018] [285 128]	[6496 1767] [214 204]	[6864 1399] [144 274]

Training Time (s)	3.64	5.06	0.009	5.39	7.16	0.012

Table 45: Performance Using the Top 12 Features

	SP-LIME Top 12 Features			**PCA Top 12 Features**		
	CatBoost	**RF**	**NB**	**CatBoost**	**RF**	**NB**
Accuracy	0.919	0.889	0.781	0.872	0.834	0.822
Precision	0.983	0.982	0.988	0.978	0.976	0.979
Confusion Matrix	[7685 578] [129 289]	[7432 831] [135 283]	[6447 1816] [81 337]	[7320 943] [167 251]	[6995 1268] [175 243]	[6860 1403] [144 274]
Training Time (s)	4.45	7.86	0.015	5.29	8.69	0.009

Table 46: Performance Using the Top 22 Features

	SP-LIME Top 22 Features			**PCA Top 22 Features**		
	CatBoost	**RF**	**NB**	**CatBoost**	**RF**	**NB**
Accuracy	0.930	0.884	0.781	0.919	0.851	0.813
Precision	0.983	0.982	0.988	0.981	0.978	0.980
Confusion Matrix	[7790 473] [138 280]	[7395 868] [139 279]	[6445 1818] [81 337]	[7708 555] [152 266]	[7137 1126] [164 254]	[6776 1487] [138 280]
Training Time (s)	6.52	10.9	0.015	4.41	11.5	0.012

Table 45 presents the results using the top 12 features from SP-LIME and PCA. CatBoost had a high accuracy of 0.919 and precision of 0.983, and the training time was 4.45 s with the SP-LIME features. With the PCA features, the accuracy was 0.872 and precision was 0.978, and the training time was 5.29 s. The RF classifier achieved an accuracy of 0.889 and precision of 0.982, and the training time was 7.86 s with the SP-LIME features. With the PCA features, the accuracy was 0.834 and precision was 0.976, and the training time was 8.69 s. The NB classifier achieved an accuracy of 0.822 and precision of 0.979, and the training time was 0.009 s, with the PCA features.

The accuracy was 0.781 and precision was 0.988, and the training time was 0.015 s, with the SP-LIME features. CatBoost was the best classifier with the top 12 features from SP-LIME and PCA. Table 46 presents the results for the top 22 SP-LIME and PCA features. CatBoost was the best classifier with an accuracy of 0.930 with the SP-LIME features and 0.919 with the PCA features, and the precision was 0.983 with SP-LIME features and 0.981 with PCA features. The training times were 6.52 s and 4.41 s, respectively. RF with the SP-LIME features achieved an accuracy of 0.884 and precision of 0.982, and the training time was 10.9 s. RF with the PCA features achieved an accuracy of 0.851 and precision of 0.978, and the training time was 11.5 s. NB with the PCA features had an accuracy of 0.813 and precision of 0.980, and the training time was 0.012 s. NB with the SP-LIME features had an accuracy of 0.781 and precision of 0.988, and the training time was 0.015 s.

The results presented in Tables 44, 45, and 46 indicate that CatBoost and RF performed better with the SP-LIME top features than the PCA top features, whereas the NB classifier achieved higher accuracy with the PCA top features but had better precision with SP-LIME top features. CatBoost had a high accuracy of 0.930 and precision of 0.983 with SP-LIME features, and with PCA features, a high accuracy of 0.919 and precision of 0.981. RF had a high accuracy of 0.895 and precision of 0.981 with SP-LIME features, and with PCA features, a high accuracy of 0.851 and precision of 0.978. NB had a high accuracy of 0.782 and precision of 0.987 with SP-LIME features and with PCA features a high accuracy of 0.822 and precision of 0.979. The dataset with the top 22 SP-LIME features provides the best performance for CatBoost. For RF and NB, the dataset with the top 6 SP-LIME features provides the best performance. Overall, SP-LIME features are more effective than PCA features.

5.4 SP-LIME with CatBoost

The CatBoost classifier consistently performed better than the other classifiers in all steps. Thus, it is used in the remaining experiments with SP-LIME to select 5,10, 20, and 50 instances. The goal is to check the consistency of SP-LIME. SP-LIME was run twice for 5 instances, while a single run was conducted for the other cases. Then the top 22 features were compiled based on their frequency. In the case of 5 instances, each feature can appear a maximum of 10 times. In the 10 instances case, a feature can appear up to 10 times, while in the 20 and 50 instances cases, a feature can appear up to 20 and 50 times, respectively. Table 47 presents the SP-LIME results with

the CatBoost classifier. It shows that features 27, 46, 5, 35, 39, 32, 24, 34, 30, and 58 are the top 10 features as they appear in the top 22 in all cases and are among the top 10 in most cases. In comparison, Table 42 lists features 46, 36, 34, 39, 27, 35, 9, 10, 21, and 24 as the top 10 for the CatBoost, RF, and NB classifiers. This shows that features 46, 34, 39, 27, 35, and 24 are common top features in both Tables 42 and 47, and so have high importance in classifier decisions. Features 46, 27, and 5 are the top three features in Table 47. These features are also among the top 3 in the combined top 22 features given in Table 42, except for feature 5. Feature 5 was ranked as a top feature by SP-LIME with the CatBoost classifier, but it was not a top feature with the other classifiers. Other features are also relevant, but they did not appear consistently in the top 22 features in all cases. Table 47 also shows that features 21, 55, 41, 40, 15, 9, and 6 did not appear in the top 22 features in every case. Table 48 presents the time taken by SP-LIME for each run of instance selection. SP-LIME takes 11 min 34 s to select 5 instances with 20 features. This increases to 13 min 10 s to select 10 instances, and 14 min 45 s to select 20 instances. To select 50 instances, SP-LIME takes 15 min 5 s. These times show that as the number of instances increases, the time also increases, but the increase was small, indicating that the process is efficient.

The top 3 features based on the above results are now explained. Feature 27 is

$$\text{profit on operating activities / financial expenses}$$

and is known as the interest coverage ratio. This ratio indicates the company long term debt paying ability. A higher value indicates financial strength, while a lower ratio suggests financial difficulty [44]. Feature 46 is

$$\text{(current assets) − inventory / short-term liabilities}$$

and is known as the current ratio or quick ratio. It shows the company ability to meet its liabilities each year using its assets. A higher value shows financial strength, while a lower ratio suggests potential difficulty in covering liabilities [44]. Feature 5 is

$$\text{((cash + short-term securities + receivables − short-term liabilities) / (operating expenses − depreciation)) × 365}$$

and is known as the defensive interval ratio, which measures company ability to pay its daily cash expenses by using its available liquid assets without adding additional cash. A higher value means

a strong liquidity position and shows short-term financial stability [45]. These three ratios are important for evaluating the financial health of a company, and SP-LIME ranks them as the top three, which confirms the results with SP-LIME. Thus, despite the random selection of instances, SP-LIME consistently identifies the important features.

Table 47: SP-LIME Feature Counts with CatBoost

5 Instances		**10 Instances**		**20 Instances**		**50 Instances**	
Feature	**Count**	**Feature**	**Count**	**Feature**	**Count**	**Feature**	**Count**
27	10	27	10	27	20	27	50
46	10	46	10	46	20	46	50
5	10	5	10	35	18	5	49
35	9	35	8	5	17	34	44
39	8	32	8	32	17	35	43
36	8	58	8	34	17	32	43
29	7	34	8	25	17	39	36
40	6	24	8	13	14	58	34
9	6	13	6	39	14	24	34
32	6	30	6	38	12	30	33
24	6	39	6	29	10	13	32
34	6	9	5	30	10	48	29
30	5	10	5	48	10	19	23
58	5	11	4	58	9	3	22
15	5	48	4	44	9	36	22
55	5	36	4	3	9	38	21
21	5	3	4	36	8	26	20
13	5	29	4	19	8	29	20
48	5	21	3	26	8	52	19
26	5	38	3	52	7	11	18
41	4	26	3	10	6	21	17
6	4	19	3	41	6	55	16

Table 48: SP-LIME Time with CatBoost

Number of Instances	5 Instances	10 Instances	20 Instances	50 Instances
Time (min:s)	11:34	13:10	14:45	15:05

Chapter 6 Conclusion and Future Work

This thesis explored bankruptcy, which is a key concern in the financial sector. A step-by-step methodology was used for bankruptcy prediction using twelve classifiers with the Polish companies dataset. First, the classifiers were evaluated on the five years combined balanced and unbalanced datasets. Seven classifiers were selected for the next step. They were tested on the individual year data using different training and testing datasets. The results of the first two steps show that data balancing typically decreased classifier performance. Therefore, in the next steps, only imbalanced datasets were used. In the individual year analysis, datasets for the first, third, and fifth years were used for training. Minority class bankruptcy prediction was also considered. The three best classifiers were selected for the next step, namely NB, CatBoost and RF, used with SP-LIME and the imbalanced five years combined dataset. SP-LIME identifies the most important features and provides importance scores. After applying SP-LIME to each classifier, the top 22 important features were identified. PCA was also used to rank the features. Then the three classifiers were used with the top features from SP-LIME and PCA separately. The results showed that using the SP-LIME top features provides better results. CatBoost was selected as the best overall classifier. SP-LIME was further tested using CatBoost with different settings. These additional results show that 6 features from the top 10 in these additional tests, given in Table 47, are also among the top 10 features in the combined list of 22 features given in Table 42. Interest coverage ratio, quick ratio, and defensive interval ratio are the top three features. These ratios are important for evaluating the financial health of a company. This confirms the reliability and consistency of SP-LIME.

In the future, this work can be extended in several directions. The CatBoost classifier, which had the best performance and is a relatively new model, provides opportunities for further exploration. It can be considered with other types of data, such as text or image data. SP-LIME was employed to identify important features based on importance scores. A deeper exploration into interpretability and comparison with other explainability techniques would be beneficial. Furthermore, SP-LIME was evaluated only on financial data. Applying it to other domains like natural language processing or medical imaging may reveal new insights into model behavior.

Appendix A Financial Terms

Table 49: Definition of Financial Terms

Term	Definition
Net Profit	Money left after all expenses and taxes are deducted from revenue.
Total Assets	The total value of everything a company owns.
Total Liabilities	The total amount a company owes to creditors.
Working Capital	The difference between current assets and current liabilities.
Current Assets	Assets expected to be used or converted to cash within a year.
Short-Term Liabilities	Obligations due within one year.
Receivables	Money owed to the company.
Short-Term Securities	Investments easily converted to cash within a year.
Operating Expenses	Costs of running daily business operations.
Depreciation	Reduction in the value of fixed assets over time.
Retained Earnings	Profits kept in the company after dividends.
EBIT	Earnings before interest and taxes.
Book Value of Equity	Net value of assets after subtracting liabilities.
Sales	Total revenue from goods and services sold.
Equity	Ownership interest held by shareholders.
Gross Profit	Sales minus cost of goods sold.
Extraordinary Items	Unusual or infrequent financial events.
Financial Expenses	Costs related to borrowing funds.
Interest	The cost of borrowing money.
Profit on Sales	Share of sales revenue that becomes profit.
Total Sales	Combined value of all sales made.
Inventories	Goods and materials held for resale or production.
Long-Term Liabilities	Debts or obligations payable after one year.
Constant Capital	Permanent capital, including equity and long-term funds.
Profit on Operating Activities	Profit from main business operations only.
Rotation of Receivables	Speed at which receivables are collected.
Inventory Turnover in Days	Average number of days to sell inventory.
Cost of Products Sold (COGS)	Direct costs of producing goods sold.
EBITDA	Earnings before interest, taxes, depreciation, and amortization.
Sales (N)	Sales in the current year.
Sales (N–1)	Sales in the previous year.
Gross Profit (in 3 Years)	Gross profit accumulated over three years.
Share Capital	Funds raised by issuing shares.
Fixed Assets	Long-term assets like property and equipment.
Logarithm of Total Assets	Log-transformed value of total assets.
Current Liabilities	Current financial obligations

References

[1] E. I. Altman, E. Hotchkiss, and W. Wang, "Corporate financial distress," in *Corporate Financial Distress, Restructuring, and Bankruptcy*, 4th ed., John Wiley & Sons, Hoboken, NJ, USA, 2019, pp. 1–20.

[2] T. M. Alam *et al.*, "Corporate bankruptcy prediction: An approach towards better corporate world," *Comput. J.*, vol. 64, pp. 1731–1746, Nov. 2021.

[3] G. Jandaghi, A. Saranj, R. Rajaei, A. Ghasemi, and R. Tehrani, "Identification of the most critical factors in bankruptcy prediction and credit classification of companies," *Iran. J. Manag. Stud.*, vol. 14, pp. 817–834, Sep. 2021.

[4] A. Dasilas and A. Rigani, "Machine learning techniques in bankruptcy prediction: A systematic literature review," *Expert Syst. Appl.*, vol. 255, p. 124761, Dec. 2024.

[5] F. Barboza, H. Kimura, and E. Altman, "Machine learning models and bankruptcy prediction," *Expert Syst. Appl.*, vol. 83, pp. 405–417, Oct. 2017.

[6] C. Deb and A. Schlueter, "Review of data-driven energy modelling techniques for building retrofit," *Renew. Sustain. Energy Rev.*, vol. 144, pp. 110–990, Jul. 2021.

[7] Symbio6, "What are black box algorithms?," Accessed: May. 31, 2025. [Online]. Available: https://symbio6.nl/en/blog/what-are-black-box-algorithms.

[8] S. Jones, D. Johnstone, and R. Wilson, "Predicting corporate bankruptcy: An evaluation of alternative statistical frameworks," *J. Bus. Finan. Acct.*, vol. 44, pp. 3–34, Jan. 2017.

[9] S. Park and J. S. Yang, "Interpretable deep learning LSTM model for intelligent economic decision-making," *Knowl. Based. Syst.*, vol. 248, pp. 108–907, Jul. 2022.

[10] P. Zahiri, "Bankruptcy prediction by deep learning and machine learning methods," M.S. Thesis, Dept. of Mechanical, Industrial and Aerospace Engineering., Concordia University, 2022.

[11] S. Tomczak, "Polish companies bankruptcy," Accessed: May. 31, 2025. [Online]. Available: https://archive.ics.uci.edu/dataset/365/polish+companies+bankruptcy+data.

[12] J. L. Bellovary, D. E. Giacomino, and M. D. Akers, "A review of bankruptcy prediction studies: 1930 to present," *J. Financ. Educ.*, vol. 33, pp. 1–42, Jan. 2007.

[13] P. Danenas and G. Garsva, "Selection of support vector machines based classifiers for credit risk domain," *Expert Syst. Appl.*, vol. 42, pp. 3194–3204, Apr. 2015.

[14] L. Cleofas-Sánchez, V. García, A. I. Marqués, and J. S. Sánchez, "Financial distress prediction using the hybrid associative memory with translation," *Appl. Soft Comput.*, vol. 44, pp. 144–152, Jul. 2016.

[15] P. Melville, N. Shah, L. Mihalkova, and R. J. Mooney, "Experiments on ensembles with missing and noisy data," in *Multiple Classifier Systems,Multiple Classifier Systems, Lecture Notes in Computer Science,* vol. 3077, Springer, Berlin, Germany, Jun. 2004, pp. 293–302.

[16] X. Xiaomao, Z. Xudong, and W. Yuanfang, "A Comparison of feature selection methodology for solving classification problems in finance," *J. Phys.: Conf. Ser.*, vol. 1284, art. 012026, May 2019.

[17] U. Mahapatra, S. M. Nayak, and M. Rout, "A systematic approach to enhance the forecasting of bankruptcy data," in *Progress in Computing, Analytics and Networking, Advances in Intelligent Systems and Computing,* vol. 1119, Springer, Singapore, Jan. 2020, pp. 641–650.

[18] M. Zięba, S. K. Tomczak, and J. M. Tomczak, "Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction," *Expert Syst. Appl.*, vol. 58, pp. 93–101, Oct. 2016.

[19] V. García, A. I. Marqués, and J. S. Sánchez, "Exploring the synergetic effects of sample types on the performance of ensembles for credit risk and corporate bankruptcy prediction," *Inf. Fusion.*, vol. 47, pp. 88–101, May 2019.

[20] S. Lahmiri, S. Bekiros, A. Giakoumelou, and F. Bezzina, "Performance assessment of ensemble learning systems in financial data classification," *Intell. Syst. Account. Finance Manag*, vol. 27, pp. 3–9, Jan. 2020.

[21] W. Zhang, D. Yang, S. Zhang, J. H. Ablanedo-rosas, X. Wu, and Y. Lou, "A novel multi-stage ensemble model with enhanced outlier adaptation for credit scoring," *Expert Syst. Appl.*, vol. 165, art. 113872, Mar. 2021.

[22] M. S. Keya, H. Akter, M. A. Rahman, M. M. Rahman, M. U. Emon, and M. S. Zulfiker, "Comparison of different machine learning algorithms for detecting bankruptcy," in *Proceedings of the International Conference on Inventive Computation Technologies*, Coimbatore, India, Feb. 2021, pp. 705–712.

[23] M. Soui, S. Smiti, M. W. Mkaouer, and R. Ejbali, "Bankruptcy prediction using stacked auto-encoders," *Appl. Artif. Intell.*, vol. 34, pp. 80–100, Nov. 2019.

[24] S. Smiti and M. Soui, "Bankruptcy prediction using deep learning approach based on borderline SMOTE," *Inf. Syst. Front.*, vol. 22, pp. 1067–1083, Aug. 2020.

[25] H. Aljawazneh, A. M. Mora, P. Garcia-Sanchez, and P. A. Castillo-Valdivieso, "Comparing the performance of deep learning methods to predict companies' financial failure," *IEEE Access*, vol. 9, pp. 97010–97038, Jun. 2021.

[26] S. Ali and A. Usha, "An overlap sensitive neural network for class imbalanced data," *Data. Mining. Knowl. Discov.*, vol. 35, pp. 1654–1687, May 2021.

[27] S. Marso and M. El Merouani, “Predicting financial distress using hybrid feedforward neural network with cuckoo search algorithm,” *Procedia Comput. Sci.*, vol. 170, pp. 1134–1140, Apr. 2020.

[28] D. P. Acharjya and R. Rathi, “An extensive study of statistical, rough, and hybridized rough computing in bankruptcy prediction,” *Multimed. Tools Appl.*, vol. 80, pp. 35387–35413, Jan. 2021.

[29] Y. Aker, “Comparison of PCA and RFE-RF algorithm in bankruptcy prediction,” *Gümüşhane Univ. Sos. Bilim. Derg.*, vol. 13, pp. 1001–1008, Oct. 2022.

[30] W. Yotsawat, K. Phodong, T. Promrat, and P. Wattuya, “Bankruptcy prediction model using cost-sensitive extreme gradient boosting in the context of imbalanced datasets,” *Int. J. Electr. Comput. Eng.*, vol. 13, pp. 4683–4691, Aug. 2023.

[31] A. Chowdhury, S. Kaisar, and R. Naha, “Bankruptcy prediction for imbalanced dataset using oversampling and ensemble machine learning methods,” *AIP Conf. Proc.*, vol. 2968, art. 040003, Nov. 2023.

[32] U. H. Ainan, L. Y. Por, Y. L. Chen, J. Yang, and C. S. Ku, “Advancing bankruptcy forecasting with hybrid machine learning techniques: Insights from an unbalanced Polish dataset,” *IEEE Access*, vol. 12, pp. 9369–9381, Jan. 2024.

[33] D. B. Chakraborty and R. Ranjan, “Missing data imputation with contextual granules and AI-driven bankruptcy prediction,” in *Proceedings of the International Conference on Pattern Recognition Systems*, London, UK, Jul. 2024.

[34] A. A. Ibrahim, R. L. Ridwan, M. M. Muhammed, R. O. Abdulaziz, and G. A. Saheed, “Comparison of the CatBoost classifier with other machine learning methods,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, pp. 738–748, Dec. 2020.

[35] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, “CatBoost: unbiased boosting with categorical features,” in *Proceedings of the International Conference on Neural Information Processing Systems*, Montréal, QC, Canada, Dec. 2018, pp. 6639–6649.

[36] GeeksforGeeks, “Naive Bayes classifiers,” Accessed: May. 31, 2025. [Online]. Available: https://www.geeksforgeeks.org/naive-bayes-classifiers.

[37] Wikipedia, “Bootstrap aggregating,” Accessed: May. 31, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Bootstrap_aggregating.

[38] H. Kang and S. Kang, “A stacking ensemble classifier with handcrafted and convolutional features for wafer map pattern classification,” *Comput. Ind.*, vol. 129, art. 103450, Aug. 2021.

[39] J. Thongkam, G. Xu, and Y. Zhang, “AdaBoost algorithm with random forests for predicting breast cancer survivability,” in *Proceedings of the IEEE International Joint Conference on Neural Networks*, Hong Kong, China, Jun. 2008, pp. 3062–3069.

[40] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, “Revisiting deep learning models for tabular data,” in *Proceedings of the International Conference on Neural Information Processing Systems*, New York, NY, USA, Dec. 2021, pp. 18932–18943.

[41] S. Ö Arik and T. Pfister, “TabNet: Attentive interpretable tabular learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, Virtual, May 2021, pp. 6679–6687.

[42] M. T. Ribeiro, S. Singh, and C. Guestrin, “‘Why should I trust you?’ Explaining the predictions of any classifier,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, San Francisco, CA, USA, Aug. 2016, pp. 1135–1144.

[43] A. R. C. Maita, M. Fantinato, S. M. Peres, and F. M. Maggi, “Supporting interpretability in predictive process monitoring using process maps,” in *Enterprise Information Systems, Lecture Notes in Business Information Processing*, vol. 518, Springer, Cham, Switzerland, Jul. 2024, pp. 230–246.

[44] S. H. Atieh, “Liquidity analysis using cash flow ratios as compared to traditional ratios in the pharmaceutical sector in Jordan,” *Int. J. Fin. Res.*, vol. 5, pp. 146–158, Jun. 2014.

[45] C. Robinson, H. van Greuning, E. Henry, and M. A. Broihahn, “Financial analysis techniques,” in *International Financial Statement Analysis*, 1st ed., John Wiley & Sons, Hoboken, NJ, USA, 2008, pp. 259–321.