INSURANCE FRAUD DETECTION VIA MACHINE LEARNING AND EXPLAINABLE ARTIFICIAL INTELLIGENCE (XAI)

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CİHAN TEZEL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ACTUARIAL SCIENCES

SEPTEMBER 2025

Approval of the thesis:

INSURANCE FRAUD DETECTION VIA MACHINE LEARNING AND EXPLAINABLE ARTIFICIAL INTELLIGENCE (XAI)

submitted by **CİHAN TEZEL** in partial fulfillment of the requirements for the degree of **Master of Science in Actuarial Sciences Department, Middle East Technical University** by,

Prof. Dr. Ayşe Sevtap KESTEL _____
Dean, Graduate School of **Applied Mathematics**

Assist. Prof. Dr. Büşra Zeynep TEMOÇİN _____
Head of Department, **Actuarial Sciences**

Assist. Prof. Dr. Büşra Zeynep TEMOÇİN _____
Supervisor, **Actuarial Sciences, METU**

Examining Committee Members:

Prof. Dr. Ayşe Sevtap KESTEL
Actuarial Sciences, METU _____

Assist. Prof. Dr. Büşra Zeynep TEMOÇİN
Actuarial Sciences, METU _____

Assist. Prof. Dr. Sinem KOZPINAR
Department of Insurance, Başkent University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: CİHAN TEZEL

Signature :

ABSTRACT

INSURANCE FRAUD DETECTION VIA MACHINE LEARNING AND EXPLAINABLE ARTIFICIAL INTELLIGENCE (XAI)

TEZEL, CİHAN
M.S., Department of Actuarial Sciences
Supervisor : Assist. Prof. Dr. Büşra Zeynep TEMOÇİN

September 2025, 84 pages

This thesis studies insurance fraud detection with machine learning (ML) methods and explainable artificial intelligence (XAI) tools. The dataset contains insurance policies and their related claims, where fraud is marked as a binary variable. The preparation stage covers missing data handling, encoding of categorical fields, and several feature engineering steps. Since fraud cases are rare, the Synthetic Minority Oversampling Technique (SMOTE) is employed, and models are trained on both the original and the balanced data. The algorithms tested include Logistic Regression, Random Forest, Support Vector Machines (SVM), k-Nearest Neighbors (KNN), LightGBM, and XGBoost. For XGBoost, two variants are considered: one with manually designed features such as seasonality, polynomial terms, and interactions, and another with automated feature engineering tuned by cross-validated AUC. Model performance is evaluated with ROC-AUC, accuracy, recall, specificity, and balanced accuracy, with thresholds optimized via ROC analysis. In addition, DeLong tests are applied to assess whether the observed differences in AUC across models are statistically significant. For interpretability, SHAP (SHapley Additive exPlanations) is utilized. The results indicate that liability coverage and fault assignment are the strongest drivers of fraud detection. Timing and policyholder features, such as claim month and address changes, also contribute meaningfully, though at a lower level. The findings showed that boosting methods achieved the best overall performance.

LightGBM and XGBoost outperformed the other models, with XGBoost slightly ahead on the SMOTE-balanced data (AUC $\approx$ 0.85, recall $\approx$ 90%). Both manual and automated feature engineering improved recall, thereby shifting the trade-off toward catching more fraud cases. These findings emphasized the importance of designing fraud detection models that combined strong predictive accuracy with transparent and interpretable decision-making.

Keywords: Insurance Fraud Detection, Machine Learning, XGBoost, SHAP, Imbalanced Data, Explainable Artificial Intelligence (XAI)

ÖZ

MAKİNE ÖĞRENMESİ VE AÇIKLANABİLİR YAPAY ZEKA (XAI) İLE SİGORTA SAHTEKARLIĞININ TESPİTİ

TEZEL, CİHAN
Yüksek Lisans, Aktüerya Bilimleri Bölümü
Tez Yöneticisi : Dr. Öğr. Üyesi Büşra Zeynep TEMOÇİN

Eylül 2025, 84 sayfa

Bu tezde, sigorta sahtekârlığının tespiti makine öğrenmesi (ML) yöntemleri ve açıklanabilir yapay zekâ (XAI) araçlarıyla incelenir. Veri seti, sigorta poliçeleri ve bunlara bağlı taleplerden oluşur ve sahtekârlık ikili bir değişken olarak tanımlanır. Ön işleme süreci eksik verilerin giderilmesini, kategorik alanların kodlanmasını ve çeşitli öznitelik mühendisliği adımlarını kapsar. Sahtekârlık vakalarının azlığı nedeniyle SMOTE (Synthetic Minority Oversampling Technique) yöntemi uygulanır ve modeller hem orijinal hem de dengelenmiş veriler üzerinde eğitilir. Test edilen algoritmalar arasında Lojistik Regresyon, Rassal Orman, Destek Vektör Makineleri (SVM), En Yakın Komşular (KNN), LightGBM ve XGBoost yer alır. XGBoost için iki varyant değerlendirilir: ilki mevsimsellik, polinom terimleri ve etkileşimler gibi elle tasarlanmış özniteliklere dayanır; ikincisi ise çapraz doğrulama AUC'sine göre ayarlanmış otomatik öznitelik mühendisliğini içerir. Model performansı ROC-AUC, doğruluk, duyarlılık, özgüllük ve dengeli doğruluk ölçütleri üzerinden, ROC analiziyle optimize edilen eşikler kullanılarak değerlendirilir. Ayrıca, modeller arasındaki AUC farklarının istatistiksel olarak anlamlı olup olmadığını test etmek için DeLong testleri uygulanır. Açıklanabilirlik için SHAP (SHapley Additive exPlanations) yöntemi kullanılır. Sonuçlar, sorumluluk kapsamı ve kusur durumunun sahtekârlık tespitinde en belirleyici unsurlar olduğunu gösteriyor. Talep ayı ve adres değişikliği gibi zamanlama ve poliçe sahibine ilişkin değişkenler de daha düşük düzeyde ancak anlamlı katkılar sağlıyor.

Bulgular, boosting yöntemlerinin en iyi genel performansı elde ettiğini ortaya koyuyor. LightGBM ve XGBoost diğer modelleri geride bırakıyor; özellikle SMOTE ile dengelenmiş verilerde XGBoost biraz daha öne çıkıyor (AUC $\approx$ 0.85, duyarlılık $\approx$ %90). Hem manuel hem de otomatik öznitelik mühendisliği, hatırlama oranını artırarak daha fazla sahtekârlık vakasının yakalanmasına katkı sağlıyor. Bu bulgular, güçlü tahmin performansıyla birlikte kararların nasıl verildiğini de şeffaf biçimde ortaya koyan sahtekârlık tespit modellerinin geliştirilmesinin önemini vurguluyor.

Anahtar Kelimeler: Sigorta Sahtekarlığı Tespiti, Makine Öğrenmesi, XGBoost, SHAP, Dengesiz Veri, Açıklanabilir Yapay Zeka

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisor, Assist. Dr. Büşra Zeynep Temoçin, for her invaluable guidance, encouragement, and patience throughout the course of this thesis. Her continuous support and insightful feedback have been instrumental in shaping the direction and quality of this work.

I am sincerely thankful to the members of my thesis committee, Prof. Dr. Ayşe Sevtap Kestel and Assist. Dr. Sinem Kozpınar, for their constructive comments and valuable suggestions, which significantly improved this study. I also wish to acknowledge all the faculty members of the Department of Actuarial Science at Middle East Technical University for fostering an inspiring academic environment.

On a personal note, I am profoundly grateful to my parents, Ayşe Tezel and Necati Tezel, for their endless love, sacrifices, and unwavering belief in me. I am especially indebted to my beloved wife, Gürcü, for her constant support, understanding, and inspiration. Without her encouragement and patience, this thesis would not have been possible.

I owe special thanks to my dear friends Tuğba Narlıoğlu and Tuğçe Biçer Gürgen, whose friendship and encouragement have been an essential part of this journey.

Finally, I dedicate this work to all those who believe in the power of mathematics, data, and human insight to address meaningful challenges.

TABLE OF CONTENTS

ABSTRACT vii

ÖZ ix

ACKNOWLEDGMENTS xiii

TABLE OF CONTENTS xv

LIST OF TABLES xxi

LIST OF FIGURES xxiv

LIST OF ABBREVIATIONS xxvi

CHAPTERS

1 INTRODUCTION 1

1.1 Literature Review 3

1.1.1 Problem structure: rarity, heterogeneity, and leakage 3

1.1.2 Learning algorithms used in insurance fraud 4

1.1.3 Dealing with class imbalance 5

1.1.4 Feature engineering: manual and automated 5

1.1.5 Evaluation: metrics, calibration, and statistical testing 5

1.1.6 Interpretability and regulatory transparency 6

1.1.7 Evidence from insurance studies 6

1.1.8 Synthesis and positioning of this thesis 7

2 PRELIMINARIES 11

2.1 Applications on Insurance Fraud Detection Models 12

2.1.1 Problem Setting and Notation 13

2.1.1.1 Preprocessing (Leakage-Aware) 14

2.1.2 Balancing Strategies 15

2.1.2.1 Random Sampling 15

2.1.2.2 SMOTE and Extensions 16

2.1.2.3 Cost-Sensitive Learning 17

2.1.3 Machine Learning Models 17

2.1.3.1 Logistic Regression (LR) 18

2.1.3.2 Random Forest (RF) 19

2.1.3.3 Gradient-Boosted Trees (GBDT) . . . 20

2.1.3.4 XGBoost 21

2.1.3.5 LightGBM 21

2.1.3.6 Support Vector Machines (SVM) . . . 23

2.1.3.7 K-Nearest Neighbours (KNN) 23

2.1.3.8 Feed-forward Neural Networks (NN) . 24

2.1.3.9 Confusion Matrix and Metrics 25

Threshold dependency. 27

2.1.3.10 ROC and AUC 27
2.1.4 Thresholding 29
2.1.5 Statistical Significance of AUCs 29
Pairwise significance 30
2.1.6 Feature Engineering 30
Manual feature construction. 31
Automated feature generation. 31
Validation and robustness. 31
Preprocessing. 31
2.1.7 Explainable AI (XAI) 32
2.1.7.1 SHAP 32
2.1.7.2 LIME 34
2.1.7.3 Choice of Method 34
2.1.8 Validation Protocol and Reproducibility 35
3 IMPLEMENTATION AND EVALUATION 37
3.1 Dataset and Preprocessing 38
3.1.1 Introduction to the Dataset 38
3.1.2 Variables and Data Types 38
3.1.3 Descriptive Statistics (Original Dataset) 41
3.1.4 Data Preparation and Cleaning 43
3.1.5 Training and Testing Split 44

3.1.6 Handling Unseen Categories 44

3.1.7 Imbalanced Data and SMOTE 44

3.1.8 Variable Inclusion in Modeling 46

3.2 Results on Imbalanced Data 47

3.2.1 Logistic Regression 47

3.2.2 Random Forest 48

3.2.3 XGBoost 49

3.2.4 LightGBM 50

3.2.5 Support Vector Machine (SVM) 51

3.2.6 K-Nearest Neighbors (KNN) 52

3.3 Results on SMOTE Data 53

3.3.1 Logistic Regression 53

3.3.2 Random Forest 55

3.3.3 XGBoost 56

3.3.4 LightGBM 57

3.3.5 Support Vector Machine (SVM) 58

3.3.6 K-Nearest Neighbors (KNN) 59

3.4 Model Performance Comparison: Imbalanced vs. SMOTE Data 60

3.5 Feature Engineering: XGBoost-Centered Evaluation 62

3.5.1 Manual Feature Engineering with XGBoost 62

3.5.2 Automated Feature Engineering (Auto-FE) with XGBoost 64

3.5.3 Comparison of XGBoost Variants 66

3.6 One-Sample DeLong Tests of AUCs 67

3.7 Pairwise DeLong Tests of AUCs 68

3.7.1 Computational Cost and Environment 71

3.8 **SHAP (SHapley Additive exPlanations)–Based Explainability of XGBoost Variants** 72

3.8.1 XGB (SMOTE) — Global SHAP 73

3.8.2 XGB (SMOTE) — Local SHAP 74

3.8.3 XGB + Manual FE — Global SHAP 75

3.8.4 XGB + Manual FE — Local SHAP 76

3.8.5 XGB + Auto-FE — Global SHAP 77

3.8.6 XGB + Auto-FE — Local SHAP 78

4 CONCLUSION 79

REFERENCES 81

LIST OF TABLES

Table 1.1 Selected references on general ML methods and their contributions . 8

Table 1.2 Selected references specific to insurance fraud analytics 9

Table 2.1 Summary of models and typical properties under class imbalance. . 25

Table 2.2 Generic Confusion Matrix 25

Table 3.1 General Summary of Numeric Variables (Original Dataset) 41

Table 3.2 Descriptive Statistics of Numerical Variables 41

Table 3.3 Fraudulent vs. Non-Fraudulent Claims (Counts) 41

Table 3.4 Frequency Distribution of *Make* 42

Table 3.5 Frequency Distribution of *AccidentArea* 42

Table 3.6 Frequency Distribution of *Sex* 42

Table 3.7 Frequency Distribution of *MaritalStatus* 43

Table 3.8 Frequency Distribution of *VehicleCategory* 43

Table 3.9 Frequency Distribution of *FraudFound_P* 43

Table 3.10 Class distribution after stratified train/test split (original, imbalanced) 45

Table 3.11 Training set distribution before and after SMOTE 46

Table 3.12 Confusion Matrix — Logistic Regression (Imbalanced) 47

Table 3.13 Performance Metrics — Logistic Regression (Imbalanced) 47

Table 3.14 Confusion Matrix — Random Forest (Imbalanced) 48

Table 3.15 Performance Metrics — Random Forest (Imbalanced) 48

Table 3.16 Confusion Matrix — XGBoost (Imbalanced) 49

Table 3.17 Performance Metrics — XGBoost (Imbalanced) 49
Table 3.18 Confusion Matrix — LightGBM (Imbalanced) 50
Table 3.19 Performance Metrics — LightGBM (Imbalanced) 50
Table 3.20 Confusion Matrix — SVM (Imbalanced) 51
Table 3.21 Performance Metrics — SVM (Imbalanced) 51
Table 3.22 Confusion Matrix — KNN (Imbalanced) 52
Table 3.23 Performance Metrics — KNN (Imbalanced) 52
Table 3.24 Confusion Matrix — Logistic Regression (SMOTE) 53
Table 3.25 Performance Metrics — Logistic Regression (SMOTE) 54
Table 3.26 Confusion Matrix — Random Forest (SMOTE) 55
Table 3.27 Performance Metrics — Random Forest (SMOTE) 55
Table 3.28 Confusion Matrix — XGBoost (SMOTE) 56
Table 3.29 Performance Metrics — XGBoost (SMOTE) 56
Table 3.30 Confusion Matrix — LightGBM (SMOTE) 57
Table 3.31 Performance Metrics — LightGBM (SMOTE) 57
Table 3.32 Confusion Matrix — SVM (SMOTE) 58
Table 3.33 Performance Metrics — SVM (SMOTE) 58
Table 3.34 Confusion Matrix — KNN (SMOTE) 59
Table 3.35 Performance Metrics — KNN (SMOTE) 59
Table 3.36 Comparison of Model Performance Across Regimes 60
Table 3.37 Confusion Matrix — XGBoost with Manual FE 63
Table 3.38 Test Metrics — XGBoost with Manual FE 63
Table 3.39 Auto-FE Candidate Combinations — CV AUC 65
Table 3.40 Test Metrics — XGBoost with Auto-FE (+Season) 65
Table 3.41 XGBoost Variants — Side-by-Side Comparison 66
Table 3.42 One-sample DeLong test results for AUC significance across models 68

Table 3.43 Full pairwise DeLong test results (Cross-split, unpaired) 69

Table 3.44 Pairwise DeLong test results (SMOTE-balanced data) 70

Table 3.45 Pairwise DeLong test results for XGB variants (SMOTE baseline, manual FE, and Auto-FE) 71

Table 3.46 Estimated computational times for different models 72

LIST OF FIGURES

Figure 2.1 Illustrative ROC curve (placeholder). 28

Figure 3.1 Class distribution before and after SMOTE. 45

Figure 3.2 Fraud vs. Non-Fraud distributions across train/test split and after SMOTE (navy: Non-Fraud, burgundy: Fraud). 45

Figure 3.3 ROC Curve — Logistic Regression (Imbalanced) 47

Figure 3.4 ROC Curve — Random Forest (Imbalanced) 48

Figure 3.5 ROC Curve — XGBoost (Imbalanced) 49

Figure 3.6 ROC Curve — LightGBM (Imbalanced) 50

Figure 3.7 ROC Curve — SVM (Imbalanced) 51

Figure 3.8 ROC Curve — KNN (Imbalanced) 52

Figure 3.9 ROC Curve — Logistic Regression (SMOTE) 54

Figure 3.10 ROC Curve — Random Forest (SMOTE) 55

Figure 3.11 ROC Curve — XGBoost (SMOTE) 56

Figure 3.12 ROC Curve — LightGBM (SMOTE) 57

Figure 3.13 ROC Curve — SVM (SMOTE) 58

Figure 3.14 ROC Curve — KNN (SMOTE) 59

Figure 3.15 ROC Curve — XGBoost with Manual FE 63

Figure 3.16 ROC Curve — XGBoost with Auto-FE (+Season) 65

Figure 3.17 Global SHAP Summary — XGB (SMOTE) 73

Figure 3.18 Local SHAP Decomposition — XGB (SMOTE) 74

Figure 3.19 Global Feature Importance / SHAP — XGB + FE 75

Figure 3.20 Local SHAP — XGB + FE (Example Case) 76
Figure 3.21 Global Feature Importance — XGB + Auto-FE 77
Figure 3.22 Local SHAP — XGB + Auto-FE (Example Case) 78

LIST OF ABBREVIATIONS

LR	Logistic Regression
RF	Random Forest
GBDT	Gradient-Boosted Decision Trees
XGB	Extreme Gradient Boosting (XGBoost)
XGBoost	Extreme Gradient Boosting
LGBM	Light Gradient Boosting Machine (LightGBM)
SVM	Support Vector Machines
KNN	k-Nearest Neighbours
NN	Neural Networks
VAE	Variational Autoencoder
PCA	Principal Component Analysis
RBF	Radial Basis Function (kernel)
OOB	Out-of-Bag (validation)
GOSS	Gradient-based One-Side Sampling
EFB	Exclusive Feature Bundling
FE	Feature Engineering
Auto-FE	Automated Feature Engineering
IR	Imbalance Ratio
SMOTE	Synthetic Minority Oversampling Technique
ADASYN	Adaptive Synthetic Sampling
ROC	Receiver Operating Characteristic
AUC	Area Under the Curve
ROC–AUC	Area under the ROC curve
PR	Precision–Recall
PRC	Precision–Recall Curve
PR–AUC	Area under the Precision–Recall curve
AUPRC	Area under the Precision–Recall curve
F1	F1 score

MCC	Matthews Correlation Coefficient
TP	True Positives
FP	False Positives
TN	True Negatives
FN	False Negatives
TPR	True Positive Rate (Sensitivity / Recall)
FPR	False Positive Rate
TNR	True Negative Rate (Specificity)
Kappa	Cohen's Kappa
CI	Confidence Interval
SD	Standard Deviation
SE	Standard Error
Youden's J	Threshold index for ROC optimisation
DeLong test	Nonparametric test for AUC
CV	Cross-Validation
CV–AUC	Cross-validated AUC
wCE	Weighted Cross-Entropy (loss)
XAI	Explainable Artificial Intelligence
SHAP	Shapley Additive Explanations
TreeSHAP	Tree-specific SHAP algorithm
LIME	Local Interpretable Model-agnostic Explanations
PDP	Partial Dependence Plot
ICE	Individual Conditional Expectation
ALE	Accumulated Local Effects
$\mathcal{D}$	Dataset of (x_i, y_i) pairs
X	Feature matrix
x	Feature vector
y	True class label
$\hat{y}$	Predicted class / probability
$\hat{s}(x)$	Model score for input x
$\hat{p}$	Predicted probability
$p(y = 1\|x)$	Predicted probability of fraud given x
$\Pr(\cdot)$	Probability notation
n, n_0, n_1	Total, majority, minority sample sizes

π	Class prevalence (fraud rate)
t	Decision threshold
$t^\star$	Bayes-optimal decision threshold
$C_{\mathrm{FP}}, C_{\mathrm{FN}}$	Costs of FP and FN
$\mathbb{1}\{\cdot\}$	Indicator function
$\sigma(\cdot)$	Logistic (sigmoid) function
β	Parameter vector (LR)
$\ell(\beta)$	Log-likelihood (LR)
w, b	Weight vector and bias (SVM / NN)
ξ	Slack variables (SVM)
C	Penalty parameter (SVM)
α_i	Lagrange multipliers (SVM)
$K(\cdot, \cdot)$	Kernel function (SVM)
$\phi(\cdot)$	Feature map (SVM / SHAP notation)
$G(t)$	Gini impurity at node t (trees)
η	Learning rate (boosting)
γ, λ	Regularisation parameters
$\Omega(f)$	Regularisation term
T	Number of leaves (tree)
g_i, h_i	Gradient / Hessian statistics
$G_j(b), H_j(b)$	Aggregated gradient / Hessian per bin (LightGBM)
L	Loss function
$\mathcal{N}_k(x)$	k-nearest neighbour set of x
$d(\cdot, \cdot)$	Distance metric
ReLU	Rectified Linear Unit (activation)
ELU	Exponential Linear Unit (activation)
Softmax	Normalised exponential mapping (NN output)
ϕ_0	SHAP base value (expected model output)
$\phi_j(x)$	SHAP value (feature j's contribution)
I_j	Global importance (mean $\lvert\phi_j\rvert$)
Z	Test statistic in one-sample DeLong test

CHAPTER 1

INTRODUCTION

Insurance fraud is a long-standing concern for the financial industry. Although only a small fraction of claims turn out to be fraudulent, their cumulative cost reaches billions of dollars each year worldwide. For insurers, this means direct financial losses; for policyholders, it leads to higher premiums; and for regulators, it undermines trust in the stability of the system. The importance of developing reliable fraud detection methods is therefore widely acknowledged, yet the practical challenges remain considerable [7, 40].

One of the most persistent obstacles is the distribution of the data itself. Fraud cases are rare, often representing fewer than one in twenty observations. A classifier trained on such data can achieve high overall accuracy simply by predicting every claim as genuine, but such a model would be useless in practice. The imbalance between fraudulent and non-fraudulent claims makes it difficult for algorithms to learn meaningful patterns, and it requires careful treatment both in training and in evaluation [27]. In addition, insurance data are heterogeneous. They combine numerical and categorical variables, involve interactions between coverage types, and are influenced by contextual factors such as timing or reporting behaviour. All of these aspects must be taken into account when building a detection framework.

The present study addresses these challenges using a combination of modern machine learning techniques and resampling strategies. Two regimes are examined side by side. In the first, the raw data are left in their original, highly imbalanced form. In the second, the training set is rebalanced through the use of the Synthetic Minority Oversampling Technique (SMOTE) [10], while the test set remains untouched to preserve

a realistic distribution. This design makes it possible to observe how class imbalance affects model performance and to evaluate the benefits of balancing methods without introducing bias into the final assessment. To further refine evaluation, thresholds are not fixed at the conventional 0.5 but are instead adjusted on the basis of ROC analysis, so that sensitivity and specificity are taken into account simultaneously.

A broad range of models is applied in the literature: Logistic Regression, Random Forest, XGBoost, LightGBM, Support Vector Machines, k-Nearest Neighbours, and Neural Networks. Each is assessed not only by overall accuracy but also by sensitivity, specificity, balanced accuracy, and the area under the ROC curve. For every case, confusion matrices and ROC curves are reported, allowing a transparent view of how models succeed or fail under different data conditions. In addition to these baseline models, the thesis makes use of feature engineering at two levels. First, domain knowledge is applied to construct manual transformations that reflect known aspects of claim behaviour. Second, automated feature engineering procedures are introduced, which search the space of possible transformations algorithmically [29]. Comparing the two approaches helps to determine how far expert insight can be complemented—or in some cases outperformed—by systematic search.

A further emphasis is placed on interpretability. In the context of insurance, models must not be black boxes: both regulators and practitioners need to understand the reasoning behind predictions. To that end, the study employs SHAP (Shapley Additive Explanations) [35], which attributes predictions to individual features. This allows the main drivers of fraud detection to be identified, highlights whether the models rely on plausible signals, and ensures that the system can be audited and trusted.

The contributions of the thesis can be summarised as follows. It sets out to compare a range of widely used machine learning algorithms under both imbalanced and balanced training regimes, clarifying the role of data distribution in fraud detection. It also explores the value of feature engineering, contrasting hand-crafted transformations with automated approaches. Finally, it integrates model interpretability into the evaluation, illustrating ways in which predictive strength and transparency can be combined. Taken together, these elements form a framework for insurance fraud detection that is both practical and rigorous.

The overall structure of the thesis develops step by step from context to results. The opening chapter (Chapter 1) sets the scene by outlining the motivation for the research, the relevance of fraud detection for insurers, and the objectives pursued in this work. What follows in Section 1.1 is the basis for a broader literature review: it reflects on previous contributions to fraud analytics, examines the statistical difficulties created by class imbalance, and comments on approaches to model evaluation and the increasing role of interpretability. The discussion then moves, in Section 2.1, to the methodological framework. Here the choices made in data preparation are described, together with the balancing strategies, the learning algorithms considered, the principles for setting thresholds and for comparing models, and the techniques adopted for explaining predictions. After this groundwork, Chapter 3 presents the empirical study. The analysis contrasts models trained on the original imbalanced data with those fitted after SMOTE balancing, and the results are assessed both in terms of predictive accuracy and in terms of interpretability. The thesis closes with Chapter 4, which brings together the main findings, reflects on their significance for insurance practice, and points to questions that remain open for future research.

1.1 Literature Review

Research on insurance fraud detection has evolved from rule-based auditing and classical statistical analysis to modern, data-driven machine learning. Early overviews by Bolton and Hand (2002) and Phua *et al.* (2010) documented this shift and emphasised the practical constraints of the domain: rare positives, heterogeneous inputs, and strong regulatory expectations for transparency [7, 40]. Later, domain-focused work by Viaene *et al.* (2007), Baesens (2015), and Derikx *et al.* (2016) detailed how insurance datasets differ from other fraud settings and why evaluation must reflect operational costs and audit workflows [48, 2, 16].

1.1.1 Problem structure: rarity, heterogeneity, and leakage

Insurance fraud is a rare-event problem: fraudulent claims typically account for only a small share of observations. He and Garcia (2009) reviewed the consequences of class

imbalance for classifier training and validation, warning that naive accuracy can be misleading when one class dominates [27]. From the modelling side, King and Zeng (2001) discussed the bias of standard logistic regression under rare events and proposed corrective strategies [33]. Data are also heterogeneous: variables mix policy, claimant, vehicle, timing, and process attributes, and include many categorical levels. Handling missingness and mixed types is therefore non-trivial; recent latent-variable approaches (for example, variational autoencoders for incomplete, heterogeneous inputs) illustrate one line of work to address such issues [38].

Beyond these statistical features, evaluation can be compromised by methodological pitfalls. Varma and Simon (2006) showed that using cross-validation both for model selection and error estimation biases performance upward [46]. Kaufman *et al.* (2012) analysed data leakage patterns in applied data mining and outlined safeguards; these issues are acute in insurance where identifiers or time leakage can easily creep into the workflow [31]. Together, these studies motivate careful train–test separation, fold-wise preprocessing, and transparent reporting of resampling and thresholding rules.

1.1.2 Learning algorithms used in insurance fraud

Classical baselines such as logistic regression remain valuable due to interpretability and well-understood calibration properties [28]. Support Vector Machines offer large-margin separation in high dimensions; Platt scaling is commonly used to convert scores to probabilities [12, 41]. Instance-based methods such as k-Nearest Neighbours have also been explored, though their behaviour in high dimensions and on mixed-type features warrants care [13, 5, 4].

Tree ensembles have become the mainstay for structured insurance data. Random Forests introduced robust bagging over decorrelated trees [8]. Boosting methods—both the original gradient boosting framework [21] and modern systems such as XGBoost and LightGBM—typically deliver strong performance with sensible regularisation and high throughput [11, 32]. In practice, these models capture nonlinear effects and rich interactions among coverage variables, claim history, and temporal indicators that would be difficult to encode manually.

1.1.3 Dealing with class imbalance

Data-level remedies are widely used in insurance fraud detection. SMOTE remains the canonical oversampling method, constructing synthetic minority points along nearest-neighbour segments [10]. Variants such as Borderline-SMOTE and ADASYN attempt to focus the synthetic generation near difficult regions or adapt the sampling to local density [24, 26]. Algorithm-level strategies include cost-sensitive training, class-balanced losses, and re-weighting schemes; surveys by He and Garcia (2009) and others summarise trade-offs between the two families of methods [27]. While focal loss was introduced in computer vision, its rationale—attenuating the gradient contribution of easy negatives—maps naturally onto rare-event classification [34].

A recurring best practice is to confine any resampling to the training folds and to leave the test distribution unchanged, so that reported metrics reflect operational prevalence. In parallel, decision thresholds should not be fixed at 0.5. Instead, they can be set by explicit utility or via summary indices such as Youden's J [50].

1.1.4 Feature engineering: manual and automated

Insurance data often benefit from hand-crafted indicators rooted in domain knowledge—timing deltas, coverage coherency checks, history-based counters, or interaction terms linking driver characteristics with policy types. Such design has long been part of successful fraud analytics programs [2, 48]. Automated feature engineering complements expert work. Deep Feature Synthesis composes primitives over relational data to produce high-level aggregates [29], while ExploreKit learns transformation policies and feature selection directly from data [30]. The two approaches are not mutually exclusive: manual features can express known risk signals compactly, and automated routines can surface non-obvious interactions.

1.1.5 Evaluation: metrics, calibration, and statistical testing

ROC analysis remains a staple for threshold-free comparison [20, 25], but on highly imbalanced data the Precision–Recall curve is often more informative [14, 43]. Prob-

ability calibration matters in fraud triage and downstream costing; Brier scores quantify probabilistic accuracy [9], and both Platt scaling and isotonic regression are used in practice [41, 39]. For threshold selection, Youden's index provides a prevalence-agnostic rule, whereas business cost frameworks align decisions with audit budgets and expected loss [50].

When comparing models, significance testing of AUC differences can be performed using the nonparametric DeLong procedure [15]; bootstrap confidence intervals remain a practical complement for metrics that lack closed-form variance expressions [19]. Across all evaluations, fold-wise preprocessing and strict prevention of leakage are essential [31, 46].

1.1.6 Interpretability and regulatory transparency

Interpretability has progressed from post-hoc rule extraction and partial dependence plots to formal local and global explanation toolkits. LIME approximates the decision surface locally with sparse linear surrogates [42]. SHAP builds on Shapley values from cooperative game theory to attribute predictions to features in an additive, consistent manner; tree-specific SHAP implementations allow exact, fast explanations for ensembles [45, 35, 36]. Broader surveys by Doshi-Velez and Kim (2017), Guidotti *et al.* (2018), and the practitioner-oriented text by Molnar (2022) provide frameworks for selecting and validating explanation methods in high-stakes domains such as insurance [17, 23, 37]. In operational settings, explanation artefacts support claim review, model governance, and regulatory audits.

1.1.7 Evidence from insurance studies

Within insurance specifically, Viaene *et al.* (2007) compared Bayesian network classifiers for claim fraud and showed that structured probabilistic models can capture conditional dependencies between policy, driver, and claim variables [48]. The industry-facing synthesis by Baesens (2015) collated descriptive, predictive, and network-based techniques, with practical guidance on deployment, monitoring, and performance governance [2]. Derikx *et al.* (2016) offered a taxonomy of electronic fraud

detection approaches, highlighting cross-domain commonalities and the persistent gap between academic prototypes and production systems [16]. More recently, public datasets—such as the widely cited "Vehicle Insurance Claim Fraud Detection" data—have enabled reproducible baselines, though researchers caution that benchmark convenience should not substitute for domain representativeness [3, 31].

1.1.8 Synthesis and positioning of this thesis

Across the literature, several themes recur. First, ensemble tree methods (Random Forests, gradient boosting, XGBoost, LightGBM) repeatedly outperform linear baselines on structured claim data, provided that resampling or cost-sensitive learning is applied with care [8, 21, 11, 32, 27]. Second, threshold choice and probability calibration materially affect audit yield and downstream cost, so evaluation must report sensitivity, specificity, ROC–AUC, and PR-AUC, alongside calibrated scores [20, 14, 43, 39]. Third, interpretability is not optional: SHAP and related tools are increasingly embedded in fraud operations to justify alerts and to diagnose drift [35, 36, 37]. Finally, methodological hygiene—no leakage, fold-wise preprocessing, unbiased error estimation—is necessary to ensure that reported gains translate to practice [31, 46].

This thesis builds on these insights in three ways. It compares a broad set of models under two regimes—originally imbalanced and SMOTE-balanced training—while holding out a realistic test distribution [10]. It studies both manual and automated feature engineering to understand their incremental value in a structured insurance setting [29, 30]. And it integrates SHAP explanations into the evaluation pipeline so that improvements in ranking or recall can be traced back to concrete drivers in the data [35, 36]. The goal is to offer a reproducible and transparent reference for building fraud detection systems that are accurate, auditable, and robust.

Table 1.1: Selected references on general ML methods and their contributions

Study	**Theme**	**Main contribution**
He & Garcia (2009) [27]	Imbalance	Data-level vs algorithm-level remedies; guidance on metrics.
Chawla *et al.* (2002) [10]	Oversampling	SMOTE; synthetic examples along minority neighbourhoods.
Han *et al.* (2005) [24]	Oversampling	Borderline-SMOTE; focus on decision boundaries.
He *et al.* (2008) [26]	Oversampling	ADASYN; density-adaptive synthesis.
Breiman (2001) [8]	Ensembles	Random Forests for tabular fraud data.
Friedman (2001) [21]	Boosting	Gradient boosting framework.
Chen & Guestrin (2016) [11]	Boosting	XGBoost; efficient, regularised gradient boosting.
Ke *et al.* (2017) [32]	Boosting	LightGBM; histogram-based, fast on large tabular data.
Ribeiro *et al.* (2016) [42]	XAI	LIME; local surrogate explanations.
Lundberg & Lee (2017); Lundberg *et al.* (2020) [35, 36]	XAI	SHAP; additive, consistent attributions for trees.
Kanter & Veeramachaneni (2015) [29]	Auto-FE	Deep Feature Synthesis for relational data.
Katz *et al.* (2020) [30]	Auto-FE	ExploreKit; learning transformation policies.
Kaufman *et al.* (2012) [31]	Methods	Leakage patterns and avoidance in applied ML.
Varma & Simon (2006) [46]	Methods	Bias in CV when used for both selection and estimation.

Table 1.2: Selected references specific to insurance fraud analytics

Study	**Theme**	**Main contribution**
Bolton & Hand (2002) [7]	Survey	Early taxonomy; highlights rarity and evaluation pitfalls.
Phua *et al.* (2010) [40]	Survey	Cross-industry review; gap between research and practice.
Viaene *et al.* (2007) [48]	Insurance	Comparative study of Bayesian network classifiers.
Baesens (2015) [2]	Insurance	Practice-oriented guide to fraud analytics.

CHAPTER 2

PRELIMINARIES

Insurance fraud detection lies at the crossroads of statistics, data analysis, and domain knowledge in risk management. The data involved are imbalanced, sensitive in practice, and monitored by regulators. Before moving to detailed methods, it is useful to set out the main challenges and choices that shape the modelling process.

Fraud detection can be posed as a binary classification task: each claim is either genuine or fraudulent. The difficulty comes from the rarity of fraud, often less than five percent of all cases. Two direct consequences follow. First, accuracy by itself can be misleading, since a model that always predicts "no fraud" will look accurate but fail its real purpose. Second, most algorithms tend to favour the majority class, ignoring the weaker fraud signals. For this reason, balancing methods such as undersampling, SMOTE, or cost-sensitive learning are often applied.

Different families of models are available. Logistic regression is a simple and transparent baseline. Tree ensembles such as Random Forests, XGBoost, and LightGBM usually perform strongly on structured insurance data. Other methods, such as Support Vector Machines or k-Nearest Neighbours, are based on different principles but are less frequently used in large-scale applications. Overall, each approach involves trade-offs: some are easier to interpret, others achieve higher predictive power, and some handle class imbalance more robustly than others.

The evaluation framework must be adjusted to reflect class imbalance and cost asymmetry. Accuracy alone does not capture the risks. Measures like precision, recall, balanced accuracy, or the Matthews Correlation Coefficient are more informative.

ROC–AUC and Precision–Recall curves are useful as they summarise ranking performance across thresholds. Since the costs of missing a fraud are not equal to those of a false alarm, thresholds are often tuned with indices such as Youden's J or by explicit cost criteria. To check if one model truly outperforms another, statistical tests like DeLong's test for ROC–AUC can be used.

Reproducibility is another central point. If preprocessing, resampling, or threshold tuning leak information from the test set, reported results can be overly optimistic. All such steps should take place within cross-validation folds. Fixed random seeds, documented software settings, and open reporting of hyperparameters help others reproduce the results. Without these, models may collapse in practice.

Finally, interpretability is essential. Insurers cannot depend solely on black-box outputs. Techniques such as SHAP offer both global and local perspectives on feature effects, clarifying why a claim is classified in a particular way. This added transparency supports trust and makes it possible to uncover potential biases or irregularities in the data.

Overall, the issues of class imbalance, model choice, evaluation, reproducibility, and interpretability form the basis for the chapters that follow. The aim is not only to report predictive results, but to outline a framework that can be trusted and applied in practice.

2.1 Applications on Insurance Fraud Detection Models

This section provides a comprehensive preliminary framework for binary fraud detection in insurance, structured in the style of a technical application chapter. We start by fixing notation and preprocessing steps, then review class-imbalance remedies, learning algorithms, calibration, evaluation metrics (ROC/PR/AUC and others), threshold selection rules (Youden and cost-based), statistical comparison via DeLong's test, and explainability tools (SHAP and LIME). Where appropriate, we include small checklists, tables, and placeholder figures.

2.1.1 Problem Setting and Notation

In this study we consider the detection of fraudulent insurance claims as a binary classification problem. Let the dataset be denoted by

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n},$$

where each observation consists of a feature vector $x_i \in \mathbb{R}^p$ and a binary outcome label $y_i \in \{0, 1\}$. By convention, we assign $y_i = 1$ to indicate the presence of *fraud*, while $y_i = 0$ denotes a legitimate (non-fraudulent) claim. The number of observations in class c is written as $n_c = |\{i : y_i = c\}|$, with $n = n_0 + n_1$ the total sample size.

The class prevalence, often called the *fraud rate*, is

$$\pi = \Pr(Y = 1) \approx \frac{n_1}{n},$$

which in practical insurance datasets is typically very small, often below 5%. This motivates the use of the *imbalance ratio* (IR), defined as

$$\mathrm{IR} = \frac{n_0}{n_1},$$

to quantify the severity of class imbalance. High IR values show that fraudulent observations are substantially fewer than non-fraudulent ones [10].

A statistical or machine learning model for classification is trained to produce a score function $\hat{s}(x) \in [0, 1]$ for each claim, which can be interpreted as a propensity or estimated probability of fraud. In the ideal case of a calibrated model, $\hat{s}(x)$ approximates $\Pr(Y = 1 \mid X = x)$. To obtain hard predictions, a threshold parameter $t \in [0, 1]$ is chosen, and decisions follow the rule

$$\hat{y}(x) = \mathbb{1}\{\hat{s}(x) \geq t\}.$$

From a practical perspective, the setting reflects the operational reality faced by insurers: most claims are valid, while fraudulent claims, though rare, generate disproportionate financial risk. Consequently, detecting even a modest number of fraud cases can have significant impact on loss reduction. However, the extreme imbalance complicates model training and evaluation, since a naive model that always predicts "non-fraud" achieves high overall accuracy but fails entirely at the detection task.

This tension between majority-class dominance and minority-class importance lies at the heart of the problem formulation [20].

In what follows, we will adopt this notation consistently. Feature vectors x_i may contain both numerical and categorical variables such as policyholder demographics, claim attributes, or timing indicators. The labels y_i serve as the ground truth obtained through audits, expert review, or confirmed investigations. Performance measures, threshold selection, and statistical testing will all be expressed in terms of the formal setting established here.

2.1.1.1 Preprocessing (Leakage-Aware)

Continuous variables are standardized whenever the geometry of the feature space is central to the learning algorithm (e.g., distance-based methods such as KNN, or kernel methods such as SVM with RBF). For categorical predictors, one-hot encoding is applied to avoid the introduction of spurious ordinal structure. Missing values are imputed using simple summary statistics (e.g., mean or mode) computed exclusively on the training partition. For tree-based ensembles such as Random Forest or Gradient Boosting, which are generally insensitive to monotone rescalings of the input, standardization is not applied. This tailored preprocessing ensures that each model family is treated in a way that preserves its inductive bias.

A critical principle is that **all preprocessing operations—scaling, imputation, resampling, and threshold selection—are performed strictly *within* the training folds of cross-validation**. By keeping the test fold separate, the evaluation reflects how the model would behave on truly unseen data, rather than giving an inflated impression of accuracy. As noted in the literature [31, 46], data leakage is a well-documented risk in empirical machine learning, especially in settings with rare events such as fraud detection.

From a practical standpoint, leakage can arise in many ways: imputing missing values with global statistics, applying SMOTE oversampling before the train-test split, or tuning thresholds based on the full dataset. Each of these steps may unintentionally pass information from the test portion into the training pipeline, which compro-

mises the credibility of the reported evaluation. By confining all steps to the training folds, we ensure that reported performance metrics mirror the true generalization setting—where unseen claims must be classified without access to their true labels.

Such a disciplined approach is especially vital in fraud detection, where the positive class is rare and even minor leakage can give the illusion of high sensitivity that vanishes in deployment. Consequently, our preprocessing protocol is explicitly leakage-aware and aligned with best practices in the literature.

2.1.2 Balancing Strategies

In supervised classification, a high imbalance ratio (IR) often causes severe degradation in the detection of minority events. In fraud detection, the majority of claims are genuine, while fraudulent ones form only a small fraction of the dataset. When IR is large, standard classifiers tend to bias their predictions toward the majority class, yielding low sensitivity for fraud and making the conventional threshold $t = 0.5$ inadequate [27, 33]. Addressing imbalance is therefore a central part of the modeling pipeline. In what follows, we discuss three groups of strategies that are frequently adopted in practice: random resampling, synthetic oversampling methods, and cost-sensitive learning.

2.1.2.1 Random Sampling

A straightforward way to address imbalance is to modify the class distribution itself. **Undersampling** works by removing a portion of the majority cases. Let n_0 and n_1 be the numbers of majority and minority observations. If we aim for a ratio ρ, a subset

$$\mathcal{D}_0' \subset \mathcal{D}_0, \qquad |\mathcal{D}_0'| \approx \rho \cdot n_1,$$

is kept, giving a new dataset size of $|\mathcal{D}'| = |\mathcal{D}_0'| + n_1$. This balances the classes and can reduce training time, though the price is the loss of potentially useful majority information [27].

On the other hand, **oversampling** increases the number of minority cases by repeti-

tion:

$$\mathcal{D}_1' = \underbrace{\mathcal{D}_1 \cup \cdots \cup \mathcal{D}_1}_{m \text{ times}},$$

with m chosen so that $|\mathcal{D}_1'|$ is closer to n_0. This leaves the original data intact but may cause models to overfit, since the extra points are simply copies.

In practice, both strategies are very simple, and even with their drawbacks, they are often the first step researchers try before turning to more advanced balancing methods.

2.1.2.2 SMOTE and Extensions

A more refined family of techniques creates artificial minority examples rather than merely duplicating them. The most widely used is the Synthetic Minority Oversampling Technique (SMOTE), which interpolates between a minority point and one of its nearest neighbours [10]:

$$x_{\text{new}} = x_i + \lambda\big(x_i^{(\text{NN})} - x_i\big), \qquad \lambda \sim \mathcal{U}(0,1).$$

This generates more diverse samples and reduces the risk of overfitting inherent in naive oversampling. Several extensions refine this idea. **Borderline-SMOTE** [24] concentrates synthetic generation in regions where minority points are surrounded by majority cases, thereby strengthening decision boundaries. **ADASYN** [26] further adapts the number of synthetic samples according to local difficulty, assigning more weight to hard-to-learn areas.

From a methodological perspective, these methods are powerful but require care. They should be applied strictly *within* cross-validation folds to avoid leaking synthetic information into test data. Moreover, when categorical features are encoded into sparse vectors, interpolation can produce unrealistic hybrids. The choice of neighbour parameter k and oversampling rate must also be aligned with the expected prevalence of fraud at prediction time, otherwise models may learn patterns that do not generalize.

2.1.2.3 Cost-Sensitive Learning

An alternative to modifying the data is to adjust the learning objective. In cost-sensitive approaches, misclassifications of minority cases are penalized more heavily than those of majority cases. The most common implementation is *class-weighted cross-entropy*:

$$\mathcal{L}_{\text{wCE}} = -\Big[w_1\, y \log(\hat{p}) + w_0\, (1-y) \log(1-\hat{p})\Big],$$

where $w_1 > w_0$ ensures that fraudulent cases are not ignored.

More recently, **focal loss** has been proposed in computer vision and applied to fraud contexts as well [34]:

$$\mathcal{L}_{\text{focal}} = -\alpha(1-\hat{p})^{\gamma} y \log(\hat{p}) - (1-\alpha)\hat{p}^{\gamma}(1-y)\log(1-\hat{p}),$$

which down-weights easy negatives and focuses training on hard or rare positives.

Such methods are particularly effective with tree ensembles and neural networks, where loss functions are flexible. In practice, cost-sensitive learning can achieve performance comparable to resampling without generating synthetic cases. Although not directly implemented in our final pipeline, these techniques remain a well-established option in the broader literature and show that imbalance can be addressed either by altering the data distribution or by shaping the objective function.

2.1.3 Machine Learning Models

We consider a range of machine learning algorithms that differ in how they handle complexity, interpretability, and imbalance. The models can be grouped into several families. When considering models for fraud detection, we rely on several established families in the machine learning literature. Logistic regression represents the linear class and provides a clear baseline, often valued for its interpretability rather than raw accuracy [28]. Support Vector Machines extend linear ideas with kernels, which allow complex decision boundaries but often at the cost of higher computational demands [12]. Tree-based ensembles such as Random Forest, XGBoost, and LightGBM have become particularly popular in applied tasks; their strength lies in combining many

weak learners to capture nonlinear relations, though they usually require post-hoc explanation techniques to be interpretable [21].

Instance-based approaches, like k-Nearest Neighbours, rely directly on the notion of similarity between cases. They are simple and adaptive, yet sensitive to how distances are measured and to the presence of noise. Neural networks take a different route: they build several layers of internal representation, which makes them capable of modelling highly nonlinear relationships [6]. However, in our experiments their predictive performance was consistently weaker than ensemble and boosting methods, so they were not retained in the final evaluation pipeline.

Taken together, the model families do not compete on the same terms but rather highlight different aspects of the task. Linear models remain transparent and easy to explain, kernel methods add the possibility of curved decision boundaries, ensembles bring predictive strength though at some cost in interpretability, instance-based methods rely on local neighbourhoods, and neural networks provide the broadest function class but demand greater complexity. Comparing them side by side therefore allows us to assess not only accuracy but also the practical trade-offs involved in fraud detection.

2.1.3.1 Logistic Regression (LR)

Logistic regression is a standard tool in the analysis of binary outcomes and is frequently used as a first benchmark in applied classification tasks. The underlying idea is that the relationship between predictors and the outcome is not modelled directly on the probability scale, but instead through the log-odds. In this setting, the logarithm of the odds of fraud versus non-fraud is written as a linear function of the covariates:

$$\log \frac{\Pr(Y = 1 \mid x)}{\Pr(Y = 0 \mid x)} = \beta_0 + \beta^\top x.$$

Through this transformation the model avoids predicting probabilities outside the interval $[0, 1]$, because the linear predictor is passed through the logistic, or sigmoid, function:

$$\Pr(Y = 1 \mid x) = \sigma(\beta_0 + \beta^\top x), \qquad \sigma(z) = \frac{1}{1 + e^{-z}}.$$

The logistic curve guarantees that extreme negative values of the linear predictor yield probabilities close to zero, while extreme positive values push them smoothly toward one.

In practice, the model parameters are estimated by maximum likelihood. For a dataset of independent pairs $\{(x_i, y_i)\}_{i=1}^{n}$, the procedure matches the predicted probabilities to the outcomes actually observed. On the log scale, the likelihood takes the form

$$\ell(\beta) = \sum_{i=1}^{n} \Big[y_i \log p_i + (1 - y_i) \log(1 - p_i) \Big], \qquad p_i = \sigma(\beta_0 + \beta^\top x_i).$$

Each term rewards the model if it assigns high probability to what really happened, and penalizes it otherwise. Because this objective function is concave, numerical routines do not get stuck in local optima: they consistently return the same solution. This stability is one of the reasons logistic regression remains a preferred benchmark, especially when datasets are large or somewhat noisy.

2.1.3.2 Random Forest (RF)

Random Forest builds a collection of decision trees and lets them vote together. Each tree is fitted to a bootstrap sample of the training data, and at every node it looks only at a randomly chosen subset of predictors. This added randomness keeps the trees different from each other and makes the overall model more reliable.

This randomness reduces correlation between trees and makes the ensemble more stable. For classification tasks, the final prediction is the majority vote:

$$\hat{y}(x) = \text{mode}\{T_1(x), T_2(x), \ldots, T_B(x)\},$$

while probability estimates are obtained by averaging tree votes:

$$\hat{p}(Y = 1 \mid x) = \frac{1}{B} \sum_{b=1}^{B} \mathbf{1}\{T_b(x) = 1\}.$$

Tree growth is guided by impurity measures, most often the Gini index,

$$G(t) = \sum_{k=1}^{K} p_k(1 - p_k),$$

where p_k is the fraction of class k in node t. Splits in each tree are selected to reduce impurity as much as possible.

In practice, the method works well across different types of data: it is not very sensitive to whether variables are measured on different scales, it can handle a mixture of categorical and numeric inputs, and it usually copes with outliers or missing values without extra preparation.

The model also provides its own internal check of accuracy through the out-of-bag error, since each tree is built on a bootstrap sample that leaves part of the data unused. One limitation is that the raw class probabilities may not always be reliable, and in such cases a calibration step—often assessed with reliability plots—can be helpful.

2.1.3.3 Gradient-Boosted Trees (GBDT)

Gradient boosting does not rely on a single deep tree but instead builds the model in a gradual way. At each stage, a new tree is fitted with the goal of improving the errors left from the earlier trees. The mathematical update is

$$\hat{F}^{(t)}(x) = \hat{F}^{(t-1)}(x) + \eta f_t(x),$$

where η is a learning rate and $f_t(x)$ is the tree grown on the residuals from the previous step.

This procedure means that the ensemble grows step by step: the first trees usually explain the dominant signals in the data (for instance, claim size in an insurance dataset), while later trees focus on more subtle or difficult cases (such as late reporting or rare combinations of features). Because each update is scaled by η, the contribution of an individual tree is moderated, which makes the method less likely to overfit.

In practice, this trade-off between flexibility and caution is one of the reasons why gradient boosting has become a standard choice in many applied prediction problems.

With this additive approach, GBDT can represent nonlinear relations and variable interactions in a flexible yet controlled manner. In practice, two implementations are widely used: *XGBoost*, which introduces second-order optimization and regularization, and *LightGBM*, which speeds up training with histogram-based splits and

leaf-wise growth. Both keep the boosting principle but differ in computational tricks and scalability features.

2.1.3.4 XGBoost

XGBoost (*Extreme Gradient Boosting*) is a widely used implementation of gradient boosting, first introduced in [11]. The method extends the boosting framework [21] by including regularization, practical optimization steps, and options to address class imbalance.

The model minimizes a regularized objective

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} \ell\Big(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\Big) + \Omega(f_t),$$

with ℓ a differentiable loss function and

$$\Omega(f_t) = \gamma T + \tfrac{1}{2}\lambda \|w\|^2,$$

where T is the number of leaves and w the vector of leaf weights. Using a second-order Taylor expansion of ℓ, the approximate gain becomes

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \tfrac{1}{2} h_i f_t(x_i)^2\right] + \Omega(f_t),$$

with g_i and h_i denoting the first and second derivatives of the loss. This Newton-style update improves convergence and stability.

Additional features include shrinkage via a learning rate η, column and row subsampling, and limits on tree depth. For highly skewed problems, such as fraud detection, the parameter `scale_pos_weight` adjusts the loss to give more influence to minority cases. Altogether, XGBoost combines accurate optimization with practical controls, making it one of the most effective boosting algorithms in applied work.

2.1.3.5 LightGBM

LightGBM (*Light Gradient Boosting Machine*) is a gradient boosting framework optimized for large and high-dimensional data [32]. It builds trees using a histogram-based method: continuous features are discretized into K bins, and splits are chosen

by scanning bin borders rather than raw values. If $G_j(b)$ and $H_j(b)$ denote the accumulated gradient and Hessian statistics for feature j in bin b, the split gain is computed as

$$\text{Gain}(j,b) = \frac{\big(G_L(j,b)^2/(H_L(j,b)+\lambda)\big) + \big(G_R(j,b)^2/(H_R(j,b)+\lambda)\big) - \big(G_T(j)^2/(H_T(j)+\lambda)\big)}{2},$$

where L, R are the left and right partitions, T is the total, and λ is a regularization parameter. This setup enables LightGBM to test possible splits quickly, without letting the tree grow unnecessarily complex.

Two algorithmic steps improve efficiency:

- **Gradient-based One-Side Sampling (GOSS):** Data points with large gradients are always kept, while only part of the small-gradient points are used. This reduces the sample size but preserves the information that drives learning. Formally, if g_i is the gradient of sample i, then

$$w_i = \begin{cases} 1, & \text{if } |g_i| \text{ is among the top } a\% \\ \frac{1-a}{b}, & \text{if } |g_i| \text{ is in the remaining } b\% \\ 0, & \text{otherwise.} \end{cases}$$

- **Exclusive Feature Bundling (EFB):** Features that are rarely nonzero at the same time are grouped together, which lowers the number of effective variables without changing the information content. If x_j and x_k satisfy

$$x_j \cdot x_k \approx 0 \quad \forall i,$$

  they can be merged into a single feature

$$z = x_j + \alpha x_k,$$

  where α is a shift factor ensuring no overlap.

When outcomes are unbalanced, LightGBM allows the use of `is_unbalance` or explicit class weights, so that the loss gives proper weight to the minority class. In fraud detection, datasets are often large but only a small share of claims are fraudulent, so this option is particularly useful.

2.1.3.6 Support Vector Machines (SVM)

Support Vector Machines (SVM) extend linear classification by constructing maximum-margin hyperplanes in a feature space. Given training data $\{(x_i, y_i)\}_{i=1}^n$ with $y_i \in \{-1, +1\}$, the soft-margin primal problem is

$$\min_{w,b,\xi \geq 0} \ \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \quad \text{s.t.} \quad y_i\big(w^\top \phi(x_i) + b\big) \geq 1 - \xi_i,$$

where w is the weight vector, b is the intercept, ξ_i are slack variables, and $C > 0$ balances margin maximization with training error [12].

The dual formulation introduces Lagrange multipliers α_i and leads to the kernel representation

$$\max_{\alpha} \ \sum_{i=1}^{n} \alpha_i - \tfrac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j), \quad \text{s.t.} \quad 0 \leq \alpha_i \leq C, \ \sum_{i=1}^{n} \alpha_i y_i = 0,$$

where $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$ is a positive semidefinite kernel function (e.g. linear, polynomial, or RBF) [44]. The resulting classifier is

$$f(x) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b\right).$$

Since the raw SVM scores are not calibrated probabilities, post-processing is commonly applied. Platt scaling applies a logistic transformation to the decision values in order to approximate posterior probabilities [41]. Isotonic regression offers a monotonic, nonparametric calibration method that avoids parametric assumptions [39].

For datasets with unequal class frequencies, the penalty parameter C can be rescaled across classes so that errors on the minority class receive higher cost. This weighting scheme is widely used in rare-event classification, including fraud detection, where correct identification of minority cases is critical [27].

2.1.3.7 K-Nearest Neighbours (KNN)

The k-nearest neighbours algorithm (KNN) is a classical nonparametric method for classification and regression, first formalized in [13]. Given a query point x, let $\mathcal{N}_k(x)$

denote the set of its k closest training samples under a chosen distance metric $d(\cdot, \cdot)$. The predicted class is obtained by majority vote:

$$\hat{y}(x) = \arg\max_{c \in \mathcal{C}} \sum_{i \in \mathcal{N}_k(x)} \mathbf{1}\{y_i = c\},$$

where $\mathcal{C}$ is the set of possible classes and $\mathbf{1}\{\cdot\}$ is the indicator function. For regression, the prediction is typically the average of the neighbour responses.

Choice of distance metric is crucial: Euclidean distance is most common, but alternatives such as Manhattan or Mahalanobis distance can better reflect feature scales and correlations [18]. Because KNN depends on distances, the features are usually normalized or standardized. In high-dimensional spaces, distances between points tend to become similar, which reduces the ability to separate classes [5].

Possible remedies are feature selection, dimensionality reduction (e.g. PCA), or metric learning methods that adjust the distance function $d(\cdot, \cdot)$ to the data [4].

Although computationally simple at training time, prediction requires evaluating distances to all training points, leading to high cost for large datasets. Data structures such as k-d trees or ball trees improve efficiency [22]. Despite these limitations, KNN remains a baseline method in many domains, including fraud detection, where local similarity in claim profiles can provide useful signals.

2.1.3.8 Feed-forward Neural Networks (NN)

Feed-forward neural networks (NNs) are widely used to capture nonlinear interactions [6]. A simple network applies successive affine maps and nonlinearities. For layer l,

$$h^{(l)} = \sigma\left(W^{(l)} h^{(l-1)} + b^{(l)}\right),$$

where $W^{(l)}$ is the weight matrix, $b^{(l)}$ the bias vector, and $\sigma(\cdot)$ a non-linear activation such as ReLU or ELU.

In classification settings, the final layer is commonly trained with the cross-entropy loss

$$L = -\sum_{i=1}^{n} \sum_{c=1}^{C} y_{ic} \log \hat{p}_{ic},$$

where y_{ic} indicates class membership and $\hat{p}_{ic}$ is the predicted probability from the softmax output.

To limit overfitting, approaches such as dropout, weight decay, and early stopping are standard. For imbalanced data, the loss function can be rescaled to penalize minority-class errors more heavily, or replaced by focal loss to emphasize difficult cases. Because raw outputs do not always correspond to well-calibrated probabilities, simple post-processing methods such as temperature scaling can be employed.

Table 2.1: Summary of models and typical properties under class imbalance.

Model	**Notes**
LR	Linear baseline; interpretable; regularization (ℓ_1/ℓ_2); threshold tuning or class weights under skew.
RF	Bagging + feature subsampling; robust to scaling; OOB validation; often needs calibration for probabilities.
XGBoost / LightGBM	Strong tabular performance; handles skew via class weights / `scale_pos_weight`/`is_unbalance`; tuning-sensitive.
SVM	Good margins with kernels; class weights; probabilities via calibration.
KNN	Local, simple; sensitive to scaling and irrelevant features; prediction-time cost scales with n.
NN	Flexible; regularization essential; class-weighted/focal loss recommended; check calibration.

2.1.3.9 Confusion Matrix and Metrics

Table 2.2: Generic Confusion Matrix

	Pred. 0	Pred. 1
Actual 0	TN	FP
Actual 1	FN	TP

Event definitions.

- **True Positives (TP):** Cases where fraud is present and correctly predicted as fraud.
- **False Positives (FP):** Legitimate cases incorrectly classified as fraud (false alarms).

- **True Negatives (TN):** Legitimate cases correctly predicted as non-fraud.
- **False Negatives (FN):** Fraud cases wrongly classified as non-fraud (missed fraud).

Metric definitions. Here, $n = \mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}$ denotes the total number of cases.

$$\mathrm{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{n}$$

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

$$\mathrm{Recall\ (TPR)} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$\mathrm{Specificity\ (TNR)} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

$$\mathrm{Balanced\ Accuracy} = \tfrac{1}{2}(\mathrm{TPR} + \mathrm{TNR})$$

$$\mathrm{F1} = \frac{2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

$$\mathrm{MCC} = \frac{\mathrm{TP} \cdot \mathrm{TN} - \mathrm{FP} \cdot \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})(\mathrm{TN} + \mathrm{FN})}}$$

Metric interpretation.

- **Accuracy:** Overall proportion of correctly classified cases, but can be misleading in imbalanced data.
- **Precision:** Share of predicted frauds that are truly fraud. High precision means few false alarms.
- **Recall (Sensitivity/TPR):** Share of actual frauds that are detected. High recall means fewer missed frauds.
- **Specificity (TNR):** Share of actual non-frauds that are correctly classified. High specificity means the model clears legitimate cases well.
- **Balanced Accuracy:** Average of recall and specificity, ensuring both classes are weighted equally.

- **F1 Score:** Harmonic mean of precision and recall, useful when balancing detection vs. false alarms.

- **Matthews Correlation Coefficient (MCC):** A correlation-style measure ranging from -1 (total disagreement) to $+1$ (perfect prediction). Robust under class imbalance.

Threshold dependency. All these metrics depend on the classification threshold t. By varying t, Receiver Operating Characteristic (ROC) curves (TPR vs. FPR) and Precision–Recall (PR) curves can be constructed. The operating point is chosen according to the context, reflecting the trade-off between missed frauds (FN) and false alarms (FP).

2.1.3.10 ROC and AUC

Consider a binary classifier that outputs a score $\hat{s}(x) \in [0, 1]$ rather than a hard label. For a threshold $t \in [0, 1]$, we classify as positive if $\hat{s}(x) \geq t$ and negative otherwise. Each threshold produces a pair

$$\big(\mathrm{FPR}(t),\ \mathrm{TPR}(t)\big),$$

where

$$\mathrm{TPR}(t) = \frac{\mathrm{TP}(t)}{\mathrm{TP}(t) + \mathrm{FN}(t)}, \qquad \mathrm{FPR}(t) = \frac{\mathrm{FP}(t)}{\mathrm{FP}(t) + \mathrm{TN}(t)}.$$

The **ROC curve** (Receiver Operating Characteristic) is the plot of $\mathrm{TPR}(t)$ against $\mathrm{FPR}(t)$ as the threshold t varies from 1 down to 0. The diagonal line $\mathrm{TPR} = \mathrm{FPR}$ corresponds to random guessing, while a curve that bows toward the upper left indicates a model with discriminative ability. In the ideal case, the curve passes through $(0, 1)$, meaning zero false positives and perfect recall.

The **Area Under the Curve (AUC)** is formally

$$\mathrm{AUC} = \int_0^1 \mathrm{TPR}(\mathrm{FPR})\, d\mathrm{FPR}.$$

This integral equals the probability that a randomly chosen positive instance receives a higher score than a randomly chosen negative:

$$\mathrm{AUC} = \Pr\left(\hat{s}(X^+) > \hat{s}(X^-)\right),$$

where X^+ denotes a positive case and X^- a negative case.

Interpretation.

- If $\mathrm{AUC} = 0.5$, the classifier has no discriminative power (equivalent to random guessing).
- If $\mathrm{AUC} = 1$, the classifier perfectly ranks all positives above negatives.
- Intermediate values measure ranking quality: e.g. $\mathrm{AUC} = 0.9$ means that in 90% of positive–negative pairs, the positive receives the higher score.

The *slope* of the ROC curve at any point relates to the likelihood ratio

$$\frac{f_+(\hat{s})}{f_-(\hat{s})},$$

where f_+ and f_- are the score densities for positive and negative classes. Steeper slopes near the y-axis correspond to high recall even at very low false-positive rates, a property especially valuable in fraud detection.

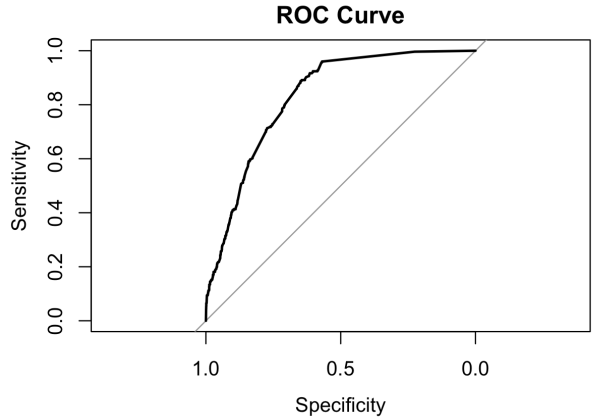

Figure 2.1: Illustrative ROC curve (placeholder).

2.1.4 Thresholding

When a classifier outputs calibrated probabilities $\hat{p}$, a decision threshold t is required to assign classes. If the costs of false negatives (C_{FN}) and false positives (C_{FP}) are asymmetric, the Bayes-optimal threshold is derived by minimizing expected risk:

$$t^{\star} = \frac{C_{\mathrm{FP}}(1-\pi)}{C_{\mathrm{FN}}\pi + C_{\mathrm{FP}}(1-\pi)},$$

where $\pi = \Pr(Y = 1)$ is the prevalence of the positive class. This formula shows that the optimal cutoff shifts depending on the balance between class prior and relative misclassification costs. For example, when fraud is rare (π small) but missing a fraud (C_{FN}) is very costly, the threshold moves downwards to increase sensitivity.

If explicit costs are unavailable, simpler criteria are often adopted. One common method is Youden's J index,

$$J = \max_{t} \big(\mathrm{TPR}(t) - \mathrm{FPR}(t)\big),$$

which corresponds to the threshold that yields the largest separation between the ROC curve and the random-guess line. Another option is to set the threshold according to desired Precision/Recall levels or to maximize an F_{β} score, depending on whether greater sensitivity or higher precision is prioritized.

It should also be emphasized that thresholds estimated on resampled or artificially balanced datasets (such as those produced by SMOTE) may not be directly valid under the original class distribution. In such cases, thresholds need to be re-evaluated using the actual prevalence to avoid distorted decision rules.

In such cases, recalibration at the true prevalence is required to avoid biased operating points.

2.1.5 Statistical Significance of AUCs

For each model, we test whether its ROC–AUC is significantly greater than 0.5, i.e. better than random guessing. Confidence intervals for AUC are obtained either by nonparametric bootstrap or by DeLong's variance estimate [15]. Formally, the null

hypothesis is $H_0 : \mathrm{AUC} = 0.5$. If the 95% confidence interval does not include 0.5, the model can be regarded as significantly better than chance [25].

$$Z = \frac{\widehat{\mathrm{AUC}} - 0.5}{\sqrt{\widehat{\mathrm{Var}}(\widehat{\mathrm{AUC}})}}$$

is approximately standard normal under H_0. This test provides a direct check of whether each model has genuine discriminative ability.

Pairwise significance In addition to testing against the random baseline, one often wishes to know whether the AUCs of two competing models differ significantly. For this purpose, the DeLong test provides a nonparametric procedure to compare correlated ROC curves, i.e. when the same test set is used for both models [15]. Formally, if $\widehat{\mathrm{AUC}}_1$ and $\widehat{\mathrm{AUC}}_2$ are the empirical AUC estimates, the null hypothesis is

$$H_0 : \quad \mathrm{AUC}_1 = \mathrm{AUC}_2,$$

and the test statistic is based on their estimated covariance matrix derived from the theory of U-statistics. The standardized difference

$$Z = \frac{\widehat{\mathrm{AUC}}_1 - \widehat{\mathrm{AUC}}_2}{\sqrt{\widehat{\mathrm{Var}}(\widehat{\mathrm{AUC}}_1 - \widehat{\mathrm{AUC}}_2)}}$$

is approximately standard normal under H_0. This allows a direct significance assessment of whether one model achieves a genuinely higher AUC than another, rather than differences arising from sampling variability.

2.1.6 Feature Engineering

In this study, the construction of additional variables was a key step, since the raw policy and claim data by themselves rarely capture the complex signals that separate fraudulent from non-fraudulent cases. To enhance predictive power, we followed two complementary routes. First, features were designed manually, drawing on established domain knowledge in insurance and actuarial practice. Second, an automated process was used to generate further candidate variables, which allowed a broader exploration of possible relationships within the data.

Manual feature construction. Drawing on actuarial and insurance practice, we derived variables that combine information from multiple sources in the raw data. Examples include ratios such as claim amount relative to coverage, interaction terms between driver age and fault assignment, and temporal indicators such as the delay between the accident and reporting date or the timing of recent policy modifications. Previous studies in fraud detection emphasize that such relational and time–based features often carry disproportionate weight in identifying unusual behaviour [40, 7]. These variables were therefore seen as essential building blocks before introducing any more complex modelling.

Automated feature generation. In parallel with manual efforts, we used algorithmic procedures to expand the feature set in a systematic manner. Automated methods search through transformations (e.g. logarithms, polynomials), cross–products, and higher–order interactions, producing candidate features that would be difficult or time–consuming to identify by hand. While not all of these automatically generated features were retained, the process helped uncover non–linear relationships that may otherwise have been overlooked. Previous studies note that automated feature construction can add value, especially when the dataset contains many diverse variables [29, 38].

Validation and robustness. All engineered features, whether manual or automated, were subject to careful validation. In particular, we checked their stability across cross–validation folds and excluded any variables that showed signs of leakage or instability. This step ensured that the resulting models did not overfit to artefacts of the data split. In addition, the engineered variables were examined within the SHAP and LIME frameworks. This confirmed that both the features derived from domain expertise and those produced through automated procedures contributed in a consistent and interpretable way to the models.

Preprocessing. Categorical fields such as vehicle type or region were converted into dummy indicators, while continuous variables were scaled when needed to help the models train more reliably. Taken together, these transformations created a feature

space that was both rich in signal and suitable for comparative modelling.

2.1.7 Explainable AI (XAI)

Predictive accuracy alone is rarely sufficient in domains such as insurance fraud detection, where the consequences of algorithmic decisions are high-stakes and subject to regulatory oversight. A model that classifies a claim as fraudulent must also provide transparent reasoning, so that human experts can verify, contest, or act upon the decision. This requirement has led to the growing field of *Explainable Artificial Intelligence (XAI)*, which develops methods to render complex algorithms interpretable [17, 23, 37]. The central motivation is twofold: (i) to establish trust and accountability, ensuring that models are not black-boxes that operate without scrutiny, and (ii) to support domain experts in identifying hidden relationships or sources of bias in the data.

XAI techniques can be classified along two key dimensions. First, we distinguish between *intrinsically interpretable models*, such as linear regression or decision rules, and *post-hoc* explanations, which aim to clarify the behaviour of complex black-box models after training. Second, explanations may be *global*, characterising overall model logic across the population, or *local*, focused on the reasoning behind a single prediction. In fraud detection, both levels of explanation are needed: global analyses identify which policyholder attributes consistently drive predictions, while local explanations justify why a particular claim was flagged.

2.1.7.1 SHAP

One of the most influential XAI frameworks is SHAP (SHapley Additive exPlanations), which unifies several strands of explainability under a rigorous game–theoretic foundation. SHAP values are rooted in the concept of Shapley values from cooperative game theory [45], where the objective is to distribute the "payout" of a game fairly among players depending on their contribution. In the context of machine learning, the players are the features, and the payout is the prediction of the model.

Formally, let f be a prediction model with feature set $F = \{1, 2, \ldots, p\}$. For an instance x, the Shapley value of feature $j \in F$ is defined as

$$\phi_j(x) = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!\,(|F| - |S| - 1)!}{|F|!} \Big(f_{S \cup \{j\}}(x_{S \cup \{j\}}) - f_S(x_S) \Big),$$

where S is a subset of features not including j, f_S is the restricted model built on S, and x_S is the subvector of x. This expression averages the marginal contribution of feature j across all possible coalitions S.

The formulation satisfies three desirable properties:

1. *Local accuracy (efficiency)*: the model output decomposes exactly as $f(x) = \phi_0 + \sum_{j=1}^{p} \phi_j(x)$, where ϕ_0 is the expected prediction over the dataset.
2. *Missingness*: if a feature is absent or uninformative, then $\phi_j(x) = 0$.
3. *Consistency (monotonicity)*: if a model changes so that feature j contributes more to the prediction, its Shapley value cannot decrease.

A drawback of exact Shapley computation is its exponential complexity in p. However, Lundberg and Lee [35] introduced *TreeSHAP*, an algorithm that computes exact SHAP values for tree ensembles in polynomial time. This efficiency makes SHAP highly compatible with the ensemble methods used in our study (Random Forest, XGBoost, LightGBM). Global feature importance can then be derived by aggregating absolute values:

$$I_j = \frac{1}{n} \sum_{i=1}^{n} |\phi_j(x^{(i)})|,$$

which measures the average contribution of each feature across the dataset.

Beyond importance rankings, SHAP offers visual tools such as dependence plots, summary plots, and force plots, which reveal how features interact and influence predictions. This dual capacity—offering both global and local explanations—has made SHAP the de facto standard for explainability in many applied domains [36].

Nevertheless, SHAP is not without caveats. In particular, correlated features can split credit in unintuitive ways, making it appear as though variables are less important individually than they are jointly [1]. To address this, one may group correlated variables, use conditional Shapley values, or complement SHAP with partial dependence or accumulated local effects analyses.

2.1.7.2 LIME

LIME (Local Interpretable Model-agnostic Explanations) takes a different approach [42]. Instead of decomposing predictions into feature contributions, LIME samples perturbed versions of the input instance x, weights them by a locality kernel π_x, and fits a sparse linear regression that approximates the black-box model f locally. The resulting surrogate coefficients indicate the local importance of features around x. The method is intuitive and computationally lightweight, and has been widely adopted in practice.

However, LIME explanations are subject to instability: small changes in the sampling process can yield different local models, particularly in high-dimensional or imbalanced data. This undermines reliability when consistent explanations are needed for regulatory or audit purposes. For this reason, LIME is best used as a diagnostic complement rather than a primary explanation tool.

2.1.7.3 Choice of Method

In this thesis, SHAP was chosen as the main interpretability technique. The rationale is threefold. First, SHAP is theoretically grounded, satisfying desirable axioms that guarantee fairness and consistency in attribution. Second, it provides both global and local perspectives, allowing us to identify which features are influential overall and to justify why individual claims are flagged. Third, efficient algorithms exist for tree ensembles, making SHAP computationally feasible for the fraud detection models considered here.

By contrast, while LIME is flexible and widely known, its instability and dependence

on arbitrary perturbations limit its suitability in high-stakes insurance applications. Thus, SHAP was preferred as the principal explainability method, supported occasionally by complementary global techniques (e.g., partial dependence, ICE) to verify robustness of the findings.

2.1.8 Validation Protocol and Reproducibility

Checking that the results are valid and can be repeated is important in a setting like insurance fraud detection. For this reason, the study follows a clear validation protocol that is both methodologically sound and practical to apply.

All models are trained and evaluated under a unified experimental design to ensure fair comparison. Specifically, stratified train–test splits and k–fold cross–validation are used so that the rare fraud class is proportionally represented in each fold. Fixed random seeds are shared across algorithms, guaranteeing that differences in performance are attributable to model properties rather than stochastic variation.

A crucial design principle is that **all preprocessing and resampling operations, including SMOTE**, are performed *within* the folds rather than prior to splitting. This prevents information from the test folds leaking into the training stage, which would otherwise give overly optimistic results. In the same way, probability thresholds are not set at random but tuned on validation folds that mirror the prevalence observed at test time. This makes sure that reported sensitivity and specificity reflect realistic operating conditions.

Software versions, package choices, and parameter settings are also recorded in full so that the analysis can be repeated later without ambiguity. Such documentation supports transparency and allows others to check the stability of the results if needed.

Practical note. In rare–event classification tasks such as fraud detection: (i) ROC–AUC should be paired with precision–recall summaries to capture performance on the minority class; (ii) thresholds should be selected based on cost functions or Youden's J rather than a naïve 0.5 cutoff; (iii) probability estimates should be calibrated to improve interpretability for human decision–makers; (iv) statistical comparisons between models can be formalised with the DeLong test, reporting confidence intervals for AUCs; (v) resampling strategies (e.g., SMOTE) or loss reweighting (e.g., focal loss) must be applied within folds to avoid leakage; (vi) post–hoc explainability methods such as SHAP should be cross–checked across folds to ensure their stability and robustness.

CHAPTER 3

IMPLEMENTATION AND EVALUATION

This chapter presents the comprehensive set of experimental findings on insurance fraud detection. The analysis is structured around two main data regimes: the original, highly *imbalanced* dataset and a *SMOTE*-balanced version. Examining both cases side by side provides a clearer understanding of how class imbalance shapes the behaviour of common machine learning algorithms.

In the imbalanced regime, the classification threshold is left at its conventional value of 0.5, so that the results reveal the extent to which severe skew suppresses the detection of fraudulent claims. This setup illustrates how overall accuracy can remain deceptively high while recall for the minority class collapses. By contrast, in the balanced regime obtained with SMOTE, thresholds are adjusted for each model using ROC-based criteria in order to account for both sensitivity and specificity. This allows for a more realistic assessment of model behaviour in operational settings where the costs of missing fraud are significant.

For every algorithm, we report the confusion matrix, ROC curve, and the full set of performance indicators, and we interpret the results in light of their implications for insurers. Beyond the baseline models, we also applied manual feature engineering—particularly within the XGBoost framework—together with an automated feature engineering (Auto-FE) procedure. Comparing these two approaches makes it possible to evaluate the added value of expert-driven transformations versus algorithmic search. Finally, model interpretability is addressed using SHAP to identify the main drivers of predictions.

3.1 Dataset and Preprocessing

3.1.1 Introduction to the Dataset

This study uses the *Vehicle Insurance Claim Fraud Detection* dataset. It was first created by Oracle for training and later shared on Kaggle by Shivam Bansal [3]. The dataset contains 15,420 rows and 33 columns. Each row represents a single insurance claim, and the columns provide details about the policyholder, the vehicle, the accident, and the claim process. Because of this structure, the dataset has been widely used in fraud detection research and is considered a useful benchmark. The target variable `FraudFound_P` labels whether a claim was fraudulent (1) or legitimate (0), making the prediction task a binary classification problem.

A key property of the dataset is its imbalance. Only a small fraction of the claims are marked as fraudulent, while the large majority are genuine. This reflects real-world insurance, where fraud is rare but financially significant. Therefore, the dataset is valuable not only for testing predictive models but also for evaluating methods that address skewed distributions, such as SMOTE.

3.1.2 Variables and Data Types

The dataset combines categorical, ordinal, and numerical variables. Below is a detailed description of the main groups of variables, including their type and meaning.

Temporal and Claim-Related Variables

- **Month** (categorical): the month when the accident occurred (e.g., “Jan”, “Jun”).
- **DayOfWeek** (categorical): day of the week of the accident (e.g., “Monday”).
- **WeekOfMonth** (numeric, integer): indicates whether the accident happened in the 1st, 2nd, 3rd, 4th, or 5th week of the month.
- **MonthClaimed** (categorical): the month in which the claim was reported.
- **DayOfWeekClaimed** (categorical): the day of the week when the claim was filed.

- **WeekOfMonthClaimed** (numeric, integer): week of the month when the claim was reported.
- **Year** (numeric, integer): year of the claim (e.g., 1994).

Policyholder Demographics

- **Sex** (categorical): gender of the policyholder ("Male" / "Female").
- **MaritalStatus** (categorical): marital status ("Single", "Married", "Widow").
- **Age** (numeric, integer): age of the insured driver.
- **AgeOfPolicyHolder** (ordinal categorical): grouped age ranges (e.g., "21 to 25", "41 to 50").
- **AddressChange_Claim** (categorical): how recently the address changed ("no change", "1 year", "2 to 3 years").

Policy and Coverage Details

- **PolicyType** (categorical): type of policy, combining vehicle and coverage information.
- **BasePolicy** (categorical): main policy type ("Liability", "Collision", "Comprehensive").
- **Deductible** (numeric, integer): deductible amount for the policy (e.g., 300, 400, 500).
- **DriverRating** (ordinal, 1–4): rating of the driver risk, with higher values indicating better ratings.

Vehicle Characteristics

- **Make** (categorical): manufacturer of the car (e.g., "Honda", "Toyota").
- **VehicleCategory** (categorical): category of the car, such as "Sport", "Sedan", "Utility".

- **VehiclePrice** (ordinal categorical): price range (e.g., “less than 20,000”, “30,000 to 40,000”).
- **AgeOfVehicle** (ordinal categorical): vehicle age (“2 years”, “more than 7”).
- **NumberOfCars** (ordinal categorical): number of cars owned (“1 vehicle”, “3 to 4”).

Claim Circumstances and Accident Context

- **AccidentArea** (categorical): whether the accident occurred in an “Urban” or “Rural” area.
- **Fault** (categorical): who was at fault, the insured driver or the third party.
- **Days_Policy_Accident** (ordinal categorical): how many days after policy issuance the accident happened (e.g., “less than 8 days”, “more than 30”).
- **Days_Policy_Claim** (ordinal categorical): how many days after policy issuance the claim was filed.
- **PastNumberOfClaims** (ordinal categorical): number of previous claims (“none”, “1”, “2 to 4”, “more than 4”).

Fraud-Detection-Related Factors

- **PoliceReportFiled** (categorical, Yes/No): whether a police report was filed.
- **WitnessPresent** (categorical, Yes/No): whether there was a witness for the accident.
- **AgentType** (categorical): type of agent involved (“Internal” or “External”).
- **NumberOfSuppliments** (categorical): additional claim documents filed (“none”, “1 to 2”, “more than 5”).

Identifiers and Target

- **PolicyNumber** (numeric, integer): identification number of the policy (administrative use only).

- **RepNumber** (numeric, integer): representative or agent number.

- **FraudFound_P** (binary, integer): target variable, 0 = non-fraud, 1 = fraud.

3.1.3 Descriptive Statistics (Original Dataset)

Table 3.1: General Summary of Numeric Variables (Original Dataset)

Variable	Min	1st Qu.	Median	Mean	3rd Qu.	Max
WeekOfMonth	1	2	3	2.79	4	5
WeekOfMonthClaimed	1	2	3	2.69	4	5
Age	0	31	38	39.86	48	80
Deductible	300	400	400	407.7	400	700
DriverRating	1	1	2	2.49	3	4
Year	1994	1994	1995	1995	1996	1996

Table 3.2: Descriptive Statistics of Numerical Variables

Variable	Mean	SD	Median	Min	Max	Range	Skew	Kurtosis
WeekOfMonth	2.79	1.29	3	1	5	4	0.12	-1.10
WeekOfMonthClaimed	2.69	1.26	3	1	5	4	0.16	-1.09
Age	39.86	13.49	38	0	80	80	0.15	0.80
Deductible	407.7	43.95	400	300	700	400	6.08	36.81
DriverRating	2.49	1.12	2	1	4	3	0.01	-1.36
Year	1995	0.80	1995	1994	1996	2	0.25	-1.41

Table 3.3: Fraudulent vs. Non-Fraudulent Claims (Counts)

Class	Count	Percent
Non-Fraudulent (0)	14497	94.0%
Fraudulent (1)	923	6.0%
Total	15420	100%

Table 3.4: Frequency Distribution of *Make*

Make	Count
Accura	472
BMW	15
Chevrolet	1681
Dodge	109
Ferrari	2
Ford	450
Honda	2801
Jaguar	6
Lexus	1
Mazda	2354
Mecedes	4
Mercury	83
Nisson	30
Pontiac	3837
Porche	5
Saab	108
Saturn	58
Toyota	3121
VW	283

Table 3.5: Frequency Distribution of *AccidentArea*

AccidentArea	Count
Rural	1598
Urban	13822

Table 3.6: Frequency Distribution of *Sex*

Sex	Count
Female	2420
Male	13000

Table 3.7: Frequency Distribution of *MaritalStatus*

Marital Status	Count
Divorced	76
Married	10625
Single	4684
Widow	35

Table 3.8: Frequency Distribution of *VehicleCategory*

Vehicle Category	Count
Sedan	9671
Sport	5358
Utility	391

Table 3.9: Frequency Distribution of *FraudFound_P*

Label	Count
0 (Non-Fraud)	14497
1 (Fraud)	923

3.1.4 Data Preparation and Cleaning

The dataset was first imported from its CSV format and checked for consistency. Administrative identifiers such as `PolicyNumber` and `RepNumber` were removed immediately. These fields are only unique keys and do not add predictive value; if used, they could even lead to data leakage [31]. Other technical identifiers, such as row indices, were also discarded to make sure models would rely only on meaningful claim information.

The outcome variable `FraudFound_P` was recoded into a binary factor with levels "0" (non-fraud) and "1" (fraud). This ensured that classification algorithms treated the label as categorical instead of numeric. Similarly, predictors such as `Make`, `AccidentArea`, `VehicleCategory`, `Sex`, and `MaritalStatus` were converted into factor types. Without this step, they would have remained as text strings,

which are not compatible with standard machine learning tools.

3.1.5 Training and Testing Split

The dataset was divided into training (70%) and testing (30%) parts by stratified sampling on the fraud label. This way, the class balance was preserved in both sets, which is important in imbalanced problems [27]. A simple random split could have left very few fraud cases in the test data, making evaluation unreliable.

After the split, checks were carried out to confirm that no identifier variables were left in either set. This precaution helped avoid the risk of models learning from non-predictive fields.

3.1.6 Handling Unseen Categories

When dividing categorical variables, some categories may appear only in the test data. Such unseen levels cannot be handled by many algorithms. To address this, unseen levels were grouped under a new category labelled “Other”. This practice is also recommended in applied work on fraud detection and churn modelling [16]. It keeps factor levels consistent and prevents test observations from being discarded.

3.1.7 Imbalanced Data and SMOTE

The training data is highly imbalanced: the fraud class is much smaller than the non-fraud class [27]. SMOTE was applied only on the training set to avoid leakage to the test set [10]. In practice, the target label was first converted to a numeric 0/1 form, and predictors were expanded to a dummy matrix with `model.matrix` (no intercept, missing values set to zero). This produced an input matrix X and a binary target y suitable for `smotefamily::SMOTE`.

SMOTE was then run with $K = 5$ and `dup_size = 5`, generating synthetic minority samples by interpolating between nearest-neighbour fraud cases. The result was a balanced (or near-balanced) training set, which prevents models from defaulting to

the majority class while preserving the original test distribution. After resampling, factor and column alignment steps were applied so that the test design matrix shared the same columns as the SMOTE-processed training matrix (extra columns dropped, missing ones added as zeros). Class counts before and after SMOTE were recorded for transparency.

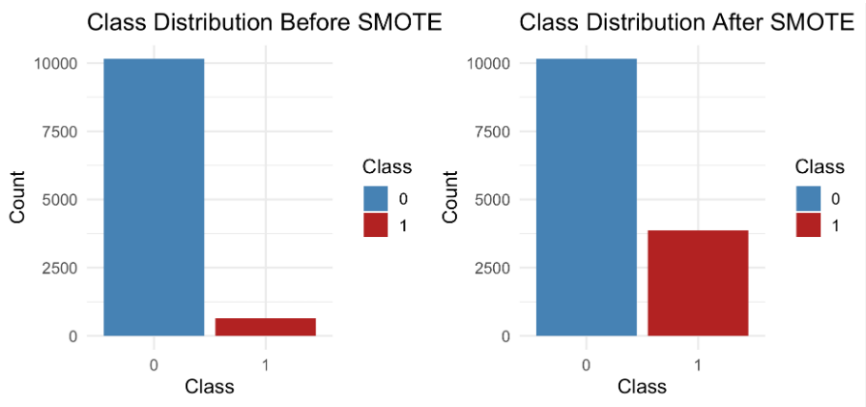

Figure 3.1: Class distribution before and after SMOTE.

This setup keeps resampling confined to the training process, standardises the input through a consistent dummy encoding, and allows downstream classifiers to learn decision boundaries that do not ignore the rare class [27, 10].

Table 3.10: Class distribution after stratified train/test split (original, imbalanced)

	Train (70%)	Percent	Test (30%)	Percent
Non-Fraud (0)	10148	94.0%	4347	94.0%
Fraud (1)	646	6.0%	276	6.0%
Total	10794	100%	4623	100%

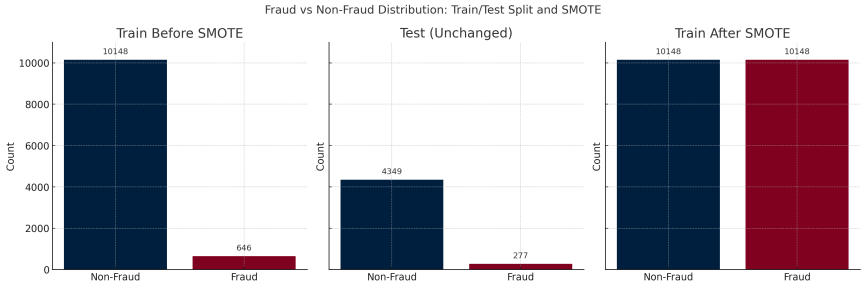

Figure 3.2: Fraud vs. Non-Fraud distributions across train/test split and after SMOTE (navy: Non-Fraud, burgundy: Fraud).

As Table 3.10 and Figure 3.2 show, fraud accounts for only about 6% of the data in both training and testing, which explains why default thresholds often yield high accuracy but poor recall for the minority class.

Table 3.11: Training set distribution before and after SMOTE

	Before SMOTE	Percent	After SMOTE	Percent
Non-Fraud (0)	10148	94.0%	10148	50.0%
Fraud (1)	646	6.0%	10148	50.0%
Total	10794	100%	20296	100%

Note: SMOTE was applied only to the training set. The test set remained unchanged (276 fraud, 4347 non-fraud), ensuring that evaluation reflects the original imbalanced distribution.

Combined with ROC-based thresholding (Youden's J), this training-only balancing increases sensitivity while controlling false alarms; the subsequent evaluation should be read in this context.

3.1.8 Variable Inclusion in Modeling

With the exception of administrative identifiers and the target variable, all other variables were included in the modelling stage. The dataset covers a broad range of information on demographics, vehicles, accidents, policies, and claim history. Prior work has shown that fraudulent behaviour often arises from interactions across such domains [48]. If predictors were excluded too early, valuable signals might have been lost. For example, policyholder age, claim timing, and vehicle type may be weak indicators on their own but, when combined, can point to suspicious behaviour.

Using a wide set of predictors also helps limit researcher bias in variable choice. Rather than restricting the feature space in advance, the study relied on ensemble models (Random Forest, XGBoost, LightGBM) and post-hoc interpretability methods (SHAP) to identify which features play a central role. This is consistent with established practice in fraud detection research, where models are first trained with the full feature set and feature importance is examined afterwards [2].

3.2 Results on Imbalanced Data

In this baseline setting, the class ratio is kept as in the raw data and the decision threshold is fixed at 0.5. This allows us to observe the natural behaviour of each model under skew. In such conditions, recall tends to be low even when AUC is reasonable, because most scores remain below the decision level. Reading AUC together with sensitivity and specificity helps separate ranking ability from the chosen operating point.

3.2.1 Logistic Regression

Table 3.12: Confusion Matrix — Logistic Regression (Imbalanced)

	Pred. 0	Pred. 1
Actual 0	4344	3
Actual 1	273	3

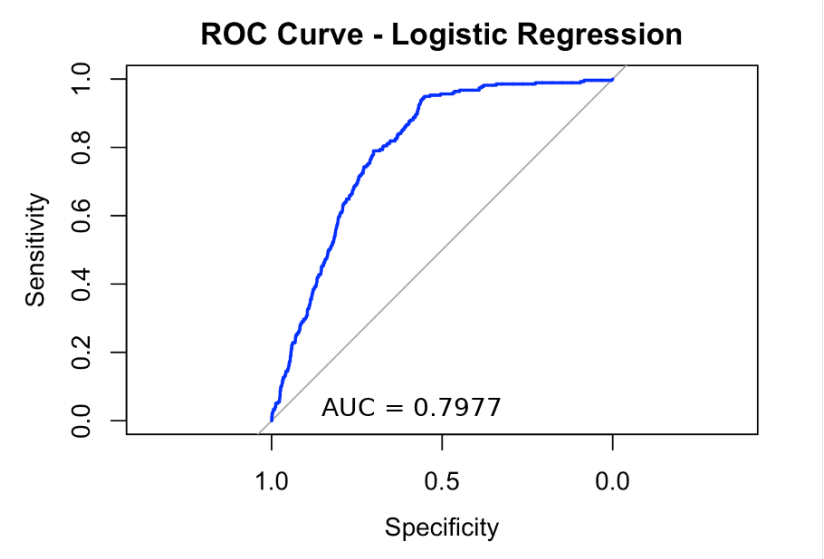

Figure 3.3: ROC Curve — Logistic Regression (Imbalanced)

Table 3.13: Performance Metrics — Logistic Regression (Imbalanced)

Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
0.9403	0.0188	0.0109	0.9993	0.5051	0.7977

Logistic regression gives an AUC close to 0.80. This shows that the model is still able to separate cases to some extent, even though the data are highly skewed. Yet its sensitivity is almost zero: with the standard 0.5 cutoff, nearly all fraud cases slip through undetected. Specificity, on the other hand, is almost perfect, because the model tends to label most claims as non-fraud. In other words, the model gains high accuracy simply by relying on the majority class. The drawback is that the minority class, which is the real focus here, is almost entirely ignored.

3.2.2 Random Forest

Table 3.14: Confusion Matrix — Random Forest (Imbalanced)

	Pred. 0	Pred. 1
Actual 0	4347	0
Actual 1	269	7

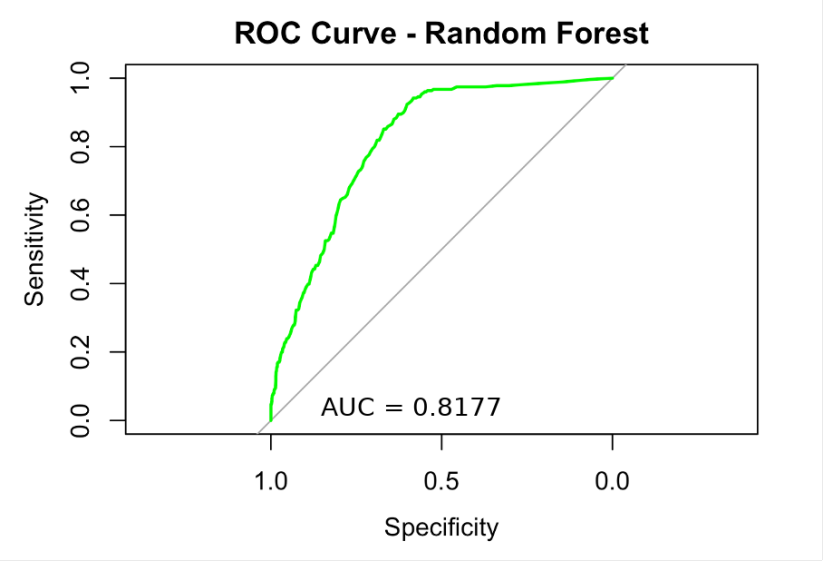

Figure 3.4: ROC Curve — Random Forest (Imbalanced)

Table 3.15: Performance Metrics — Random Forest (Imbalanced)

Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
0.9418	0.0467	0.0254	1.0000	0.5127	0.8177

Random Forest reaches an AUC of about 0.82, showing that the model is able to order

cases with reasonable success despite the skew. Sensitivity, however, is still very low, which means that only a handful of fraud cases are caught. Specificity, in contrast, appears as 1.0. This outcome is not unexpected: because the model tends to classify almost all claims as non-fraud, nearly every true non-fraud observation is recognised correctly. In effect, the high overall accuracy reflects the dominance of the majority class, while the minority fraud cases remain largely undetected.

3.2.3 XGBoost

Table 3.16: Confusion Matrix — XGBoost (Imbalanced)

	Pred. 0	Pred. 1
Actual 0	4314	33
Actual 1	254	22

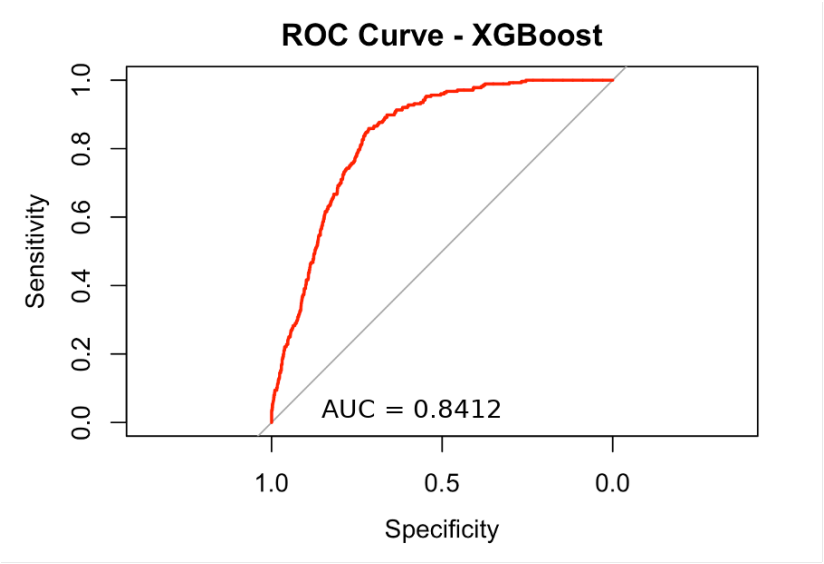

Figure 3.5: ROC Curve — XGBoost (Imbalanced)

Table 3.17: Performance Metrics — XGBoost (Imbalanced)

Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
0.9379	0.1154	0.0797	0.9924	0.5361	0.8412

XGBoost achieves an AUC of about 0.84, suggesting that it has a relatively strong ability to distinguish between fraudulent and non-fraudulent claims. Sensitivity, how-

ever, is still very low, so only a limited share of fraud cases is identified. At the same time, specificity remains close to 0.99, showing that almost all genuine claims are classified correctly. The overall accuracy therefore reflects the dominance of the majority class, while the minority fraud cases are still captured only to a small extent.

3.2.4 LightGBM

Table 3.18: Confusion Matrix — LightGBM (Imbalanced)

	Pred. 0	Pred. 1
Actual 0	4347	0
Actual 1	276	0

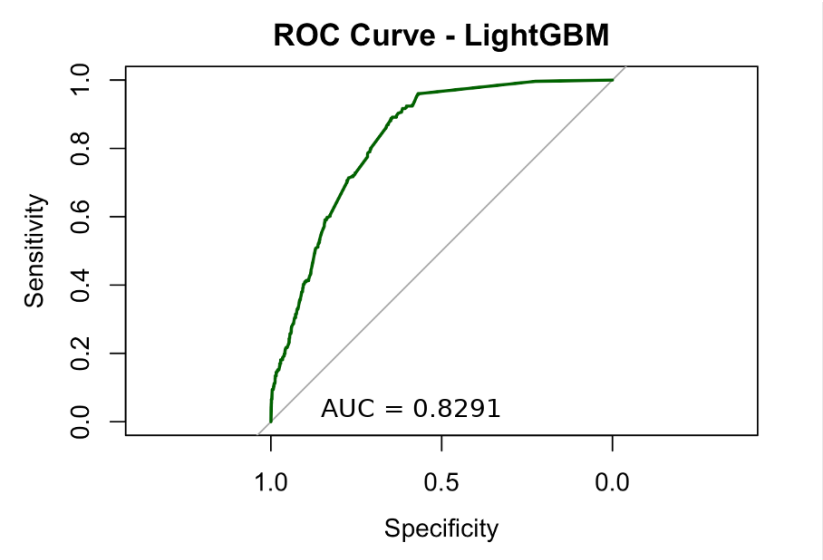

Figure 3.6: ROC Curve — LightGBM (Imbalanced)

Table 3.19: Performance Metrics — LightGBM (Imbalanced)

Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
0.9403	0.0000	0.0000	1.0000	0.5000	0.8291

LightGBM yields an AUC of about 0.83, showing that the model is still able to order cases reasonably well. However, it assigns all observations to the non-fraud class in this split. As a result, sensitivity drops to zero, since no fraud cases are predicted correctly, and the Kappa statistic is also zero, indicating that the model does not improve

over random assignment once the class imbalance is taken into account. Specificity, on the other hand, reaches 1.0, reflecting that all genuine claims are labeled correctly. This outcome highlights how the strong influence of the majority class can lead to seemingly good accuracy and specificity, while the minority class remains entirely undetected.

3.2.5 Support Vector Machine (SVM)

Table 3.20: Confusion Matrix — SVM (Imbalanced)

	Pred. 0	Pred. 1
Actual 0	4342	5
Actual 1	267	9

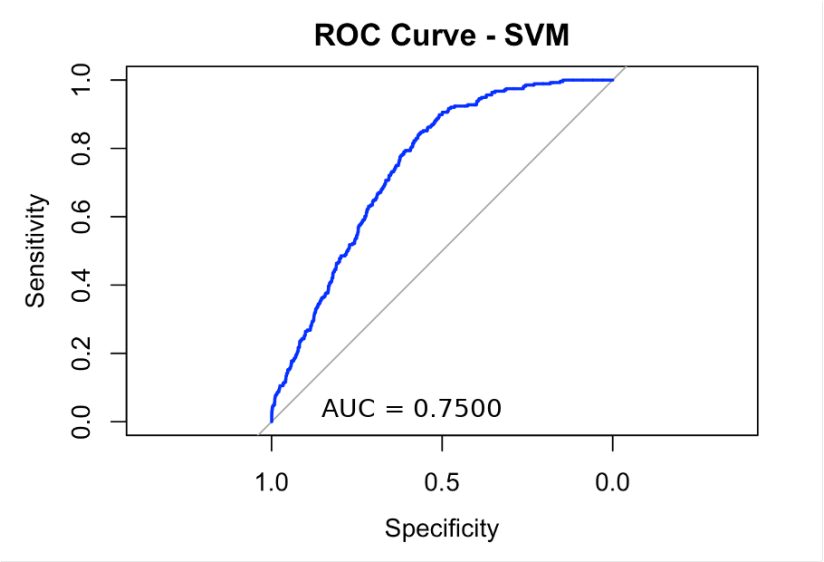

Figure 3.7: ROC Curve — SVM (Imbalanced)

Table 3.21: Performance Metrics — SVM (Imbalanced)

Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
0.9412	0.0566	0.0326	0.9988	0.5157	0.7500

The SVM model gives an AUC of 0.75. This means that the separation between fraud and non-fraud cases is only moderate. Sensitivity is extremely low, around 0.03, which shows that almost all fraud cases are missed. At the same time, speci-

ficity is close to 1.0, so the model almost always classifies genuine claims correctly. The overall accuracy, about 0.94, comes mainly from the dominance of the majority class. Since the model places nearly everything into the non-fraud category, the Kappa statistic also falls close to zero, showing that the agreement beyond chance is very limited under this imbalance.

3.2.6 K-Nearest Neighbors (KNN)

Table 3.22: Confusion Matrix — KNN (Imbalanced)

	Pred. 0	Pred. 1
Actual 0	4332	15
Actual 1	270	6

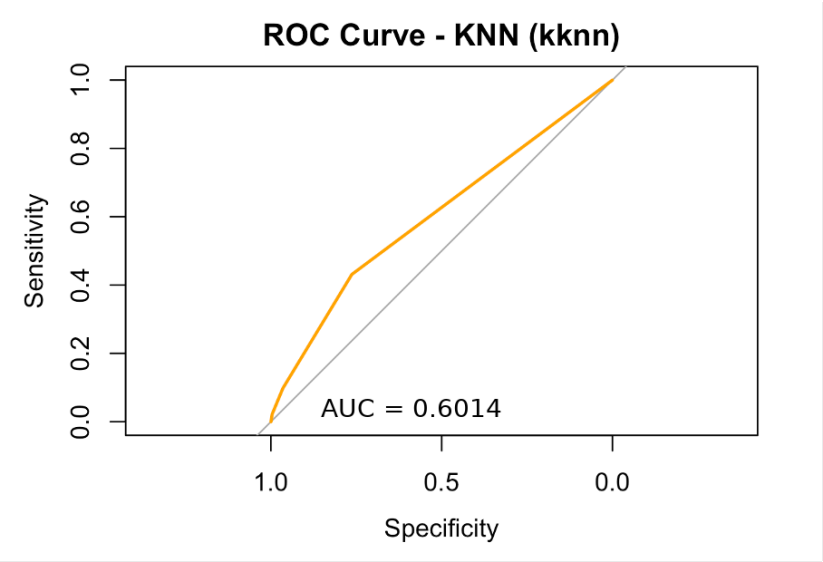

Figure 3.8: ROC Curve — KNN (Imbalanced)

Table 3.23: Performance Metrics — KNN (Imbalanced)

Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
0.9384	0.0322	0.0217	0.9965	0.5091	0.6014

KNN with k = 5 neighbors ends up being the weakest model here, with an AUC of only about 0.60. Most predictions fall into the non-fraud side, so sensitivity drops to around 0.02. In other words, hardly any fraud cases are picked up. Specificity, on

the other hand, is almost perfect at close to 1.0, meaning genuine claims are nearly always classified correctly. The overall accuracy looks high at about 0.94, but this is mostly because of the large majority class. The Kappa statistic also stays very low, showing that the model adds little once the imbalance is taken into account.

3.3 Results on SMOTE Data

In this part of the analysis, the strong imbalance in the original dataset is addressed by applying the SMOTE procedure, which generates synthetic minority cases during training. This helps the models to see a more balanced class distribution and reduces the risk that fraud cases are entirely overlooked. In addition, instead of relying on the default cutoff of 0.5, decision thresholds are tuned using Youden's J statistic derived from the ROC curves. This criterion selects the point that maximises the sum of sensitivity and specificity, leading to a more balanced operating rule.

The combination of SMOTE and Youden-based thresholds reflects a practical screening scenario: the aim is to increase the chance of catching fraudulent claims (higher recall) while still maintaining a level of specificity that avoids excessive false alarms.

3.3.1 Logistic Regression

Table 3.24: Confusion Matrix — Logistic Regression (SMOTE)

	Pred. 0	Pred. 1
Actual 0	2757	1590
Actual 1	60	216

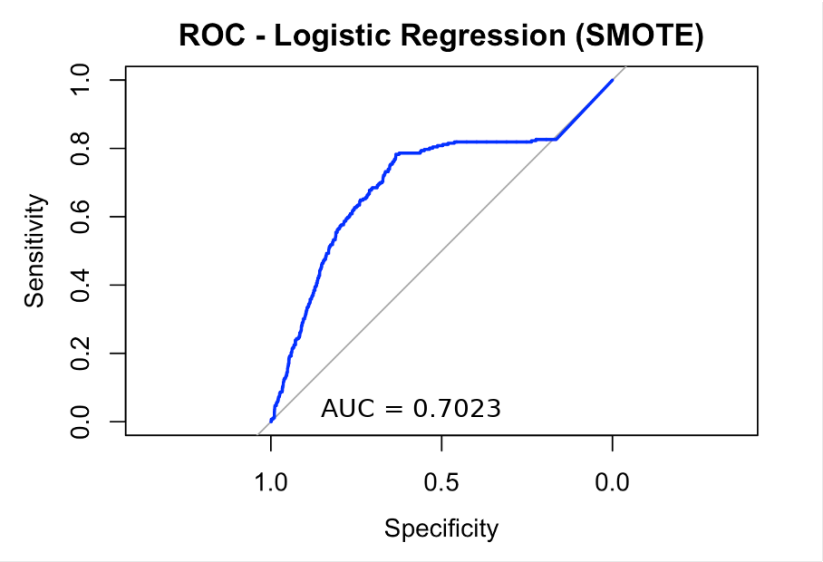

Figure 3.9: ROC Curve — Logistic Regression (SMOTE)

Table 3.25: Performance Metrics — Logistic Regression (SMOTE)

Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
0.6431	0.1159	0.7826	0.6342	0.7084	0.7023

With SMOTE balancing and the use of a threshold based on Youden's J, logistic regression performs more evenly across classes. The recall climbs to about 0.78, showing that the majority of fraud cases are now recognised. Specificity falls compared to the imbalanced setting, but remains above 0.63, so most genuine claims are still classified correctly. Balanced accuracy settles at around 0.71, which points to a more stable outcome than before, where recall had collapsed almost entirely. The AUC of roughly 0.70 confirms that the model is able to rank cases with a fair degree of separation. In practical terms, this version of logistic regression provides a serviceable reference point: it shows that even a simple linear model can detect many more fraud cases once imbalance is corrected, though at the price of some extra false alarms.

3.3.2 Random Forest

Table 3.26: Confusion Matrix — Random Forest (SMOTE)

	Pred. 0	Pred. 1
Actual 0	2820	1527
Actual 1	34	242

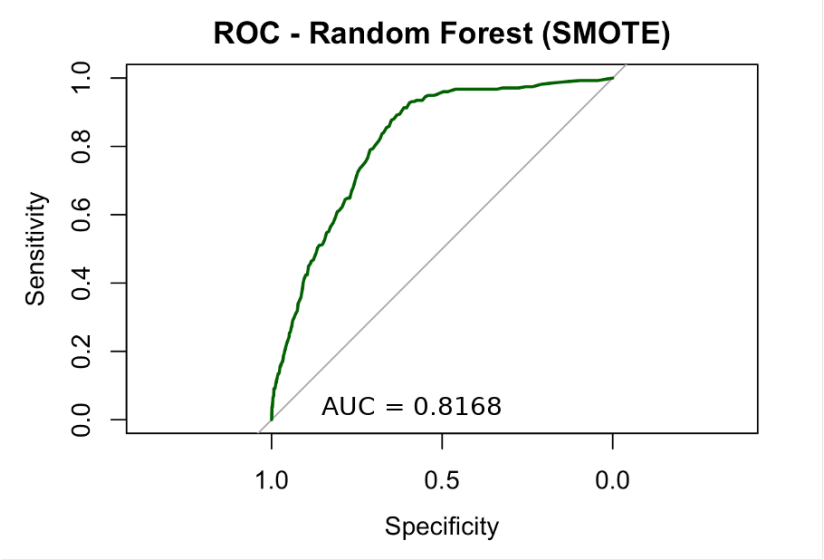

Figure 3.10: ROC Curve — Random Forest (SMOTE)

Table 3.27: Performance Metrics — Random Forest (SMOTE)

Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
0.6623	0.1488	0.8768	0.6487	0.7628	0.8168

After applying SMOTE, the Random Forest model detects a large share of the fraud cases, with recall measured at about 0.88. The specificity of 0.65 means that a portion of the genuine claims is still misclassified, but most of them are identified correctly. With a balanced accuracy of roughly 0.76, the model shows a fair level of performance across both fraud and non-fraud cases.The AUC, measured at 0.82, also points to a stable separation between fraud and non-fraud cases. These figures together suggest that the model is effective once the imbalance is reduced, even though some false positives remain.

3.3.3 XGBoost

Table 3.28: Confusion Matrix — XGBoost (SMOTE)

	Pred. 0	Pred. 1
Actual 0	3188	1159
Actual 1	48	228

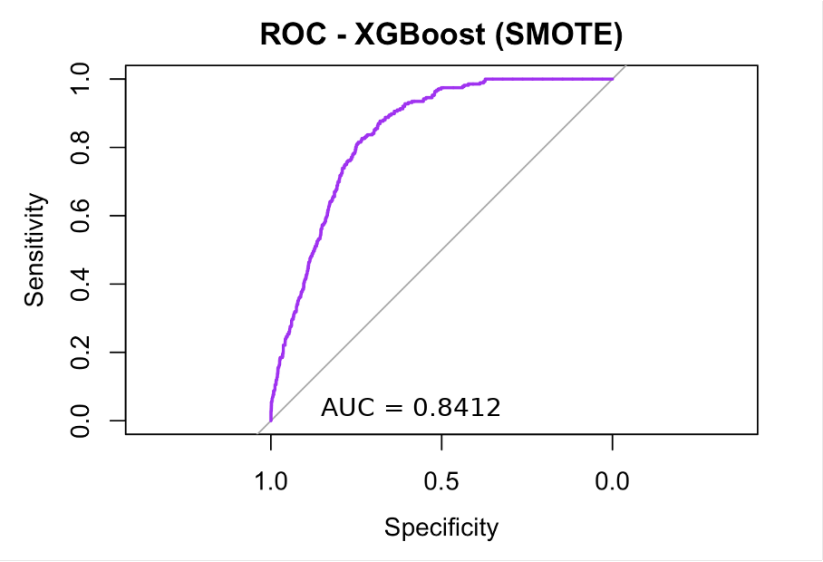

Figure 3.11: ROC Curve — XGBoost (SMOTE)

Table 3.29: Performance Metrics — XGBoost (SMOTE)

Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
0.7389	0.1939	0.8261	0.7334	0.7797	0.8412

XGBoost reaches a recall of 0.83, meaning that it identifies most of the fraud cases. The specificity is 0.73, which shows that a large part of the genuine claims are also placed in the correct group. The balanced accuracy is 0.78, pointing to a fair performance on both classes. The AUC is 0.84, the highest value among the models tested here.

The AUC before applying SMOTE was likewise 0.84. This similarity is not unusual, since AUC mainly reflects the model's ranking ability rather than the effect of class balance. SMOTE does not change the overall separation ability, but affects how thresholds and class distribution shape the balance between recall and specificity.

3.3.4 LightGBM

Table 3.30: Confusion Matrix — LightGBM (SMOTE)

	Pred. 0	Pred. 1
Actual 0	2837	1510
Actual 1	30	246

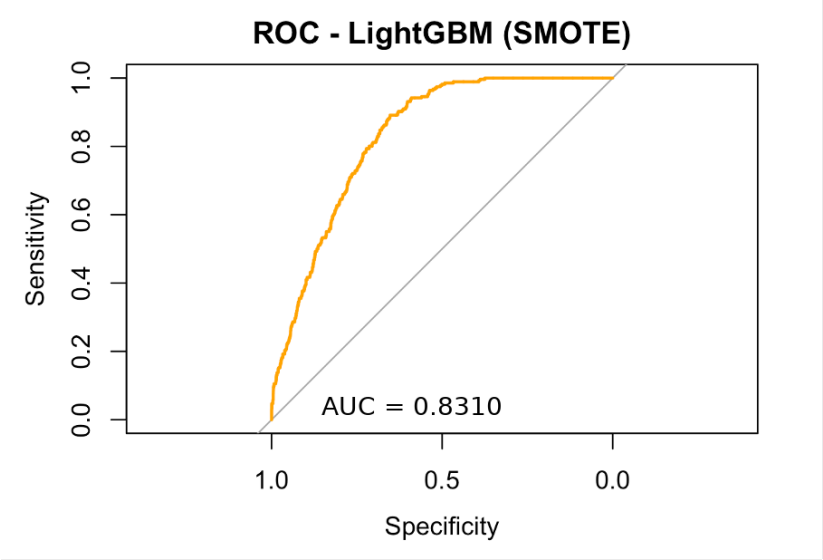

Figure 3.12: ROC Curve — LightGBM (SMOTE)

Table 3.31: Performance Metrics — LightGBM (SMOTE)

Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
0.6669	0.1549	0.8913	0.6526	0.7720	0.8310

LightGBM produces the highest recall value among the models tested, reaching nearly 0.89. This means that it captures the great majority of fraudulent claims in the dataset. The specificity, however, is lower at around 0.65, which shows that some genuine claims are misclassified as fraud, although most are still placed correctly. The balanced accuracy of about 0.77 suggests that the model handles both classes in a relatively fair way. The AUC score is 0.83, pointing to a solid ability to rank and separate fraud from non-fraud cases. These results indicate that LightGBM is reliable in terms of detecting fraud, but its tendency to generate false alarms should also be noted when considering practical applications.

3.3.5 Support Vector Machine (SVM)

Table 3.32: Confusion Matrix — SVM (SMOTE)

	Pred. 0	Pred. 1
Actual 0	2499	1848
Actual 1	46	230

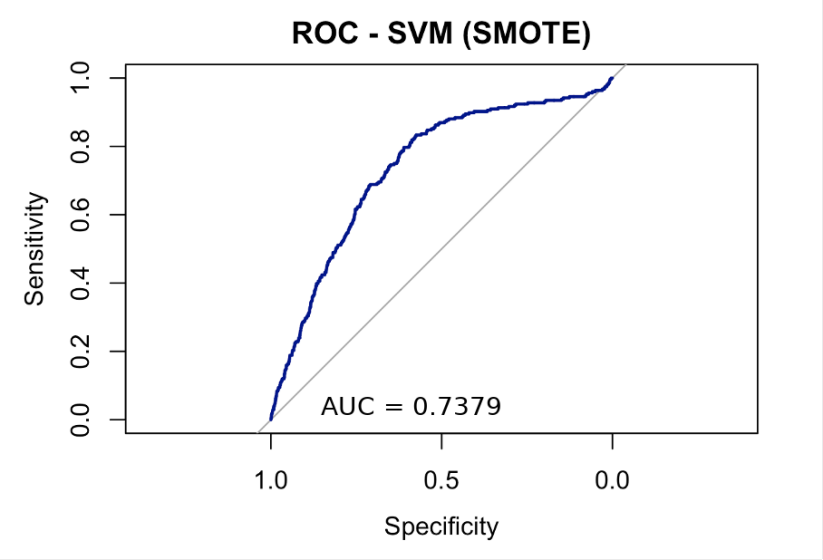

Figure 3.13: ROC Curve — SVM (SMOTE)

Table 3.33: Performance Metrics — SVM (SMOTE)

Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
0.5903	0.1006	0.8333	0.5749	0.7041	0.7379

For the Support Vector Machine (SVM), the recall value is 0.83. This tells us that the model finds most of the fraud cases in the dataset. A high recall is important because fraud cases are fewer and harder to detect. The specificity is 0.57. This shows that many genuine claims are not classified correctly, although some of them are still placed in the right group. The balanced accuracy is 0.70. This value is between recall and specificity and gives a middle score for the model. The AUC is 0.74. This means the model can separate fraud and non-fraud claims to a fair degree, but not very strongly. Taken together, the numbers show that SVM can capture fraud but still misses many genuine cases.

3.3.6 K-Nearest Neighbors (KNN)

Table 3.34: Confusion Matrix — KNN (SMOTE)

	Pred. 0	Pred. 1
Actual 0	2376	1971
Actual 1	82	194

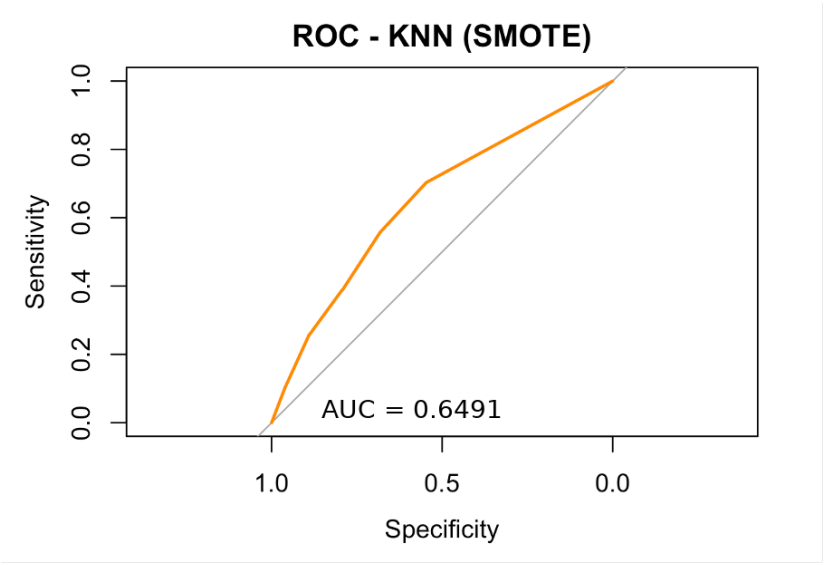

Figure 3.14: ROC Curve — KNN (SMOTE)

Table 3.35: Performance Metrics — KNN (SMOTE)

Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
0.5559	0.0593	0.7029	0.5466	0.6247	0.6491

For the K-Nearest Neighbors (KNN) model with k = 5 neighbors, the recall is 0.70. This means that the model is able to detect many of the fraud cases, but it still misses a noticeable number of them. The specificity is 0.55. This shows that a large part of the genuine claims is classified in the correct way, but there are also many mistakes. The balanced accuracy is 0.62. This value is lower than for most other models and indicates that the performance is limited when both classes are considered together. The AUC is 0.65. This means that the model has only a modest ability to tell the difference between fraud and non-fraud claims. These values together suggest that KNN can capture certain patterns in the data, yet it does not provide a very strong separation between the two classes.

3.4 Model Performance Comparison: Imbalanced vs. SMOTE Data

Up to this point, we have obtained the results of the models both on the imbalanced (train/test split) data and on the SMOTE-balanced data. These results are summarised together in Table 3.36, titled *Comparison of Model Performance Across Regimes*.

Table 3.36: Comparison of Model Performance Across Regimes

Model	Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
LR (train/test split)	0.9403	0.0188	0.0109	0.9993	0.5051	0.7977
RF (train/test split)	0.9418	0.0467	0.0254	1.0000	0.5127	0.8177
XGB (train/test split)	0.9379	0.1154	0.0797	0.9924	0.5361	0.8412
LGBM (train/test split)	0.9403	0.0000	0.0000	1.0000	0.5000	0.8291
SVM (train/test split)	0.9412	0.0566	0.0326	0.9988	0.5157	0.7500
KNN (train/test split)	0.9384	0.0322	0.0217	0.9965	0.5091	0.6014
LR (SMOTE)	0.6431	0.1159	0.7826	0.6342	0.7084	0.7023
RF (SMOTE)	0.6623	0.1488	0.8768	0.6487	0.7628	0.8168
XGB (SMOTE)	0.7389	0.1939	0.8261	0.7334	0.7797	0.8412
LGBM (SMOTE)	0.6669	0.1549	0.8913	0.6526	0.7720	0.8310
SVM (SMOTE)	0.5903	0.1006	0.8333	0.5749	0.7041	0.7379
KNN (SMOTE)	0.5559	0.0593	0.7029	0.5466	0.6247	0.6491

When all models are viewed together, some clear patterns emerge. In the imbalanced data, recall remains extremely low, which means that most fraud cases are missed. At the same time, several models achieve relatively high AUC values. This is a typical effect of imbalance: models can rank the majority class very well, which inflates AUC, but they almost completely fail to capture the minority class [20].

After SMOTE balancing, the situation changes substantially. Recall rises sharply across all models, since the minority class becomes more visible during training. For instance, logistic regression moves from a recall of barely 0.01 to nearly 0.78, while Random Forest jumps from 0.02 to 0.88. However, this improvement often comes at a cost: in some models AUC decreases after balancing (e.g., LR: 0.79 $\rightarrow$ 0.70), because synthetic oversampling can blur class boundaries [10].

Random forest shows a clear step forward compared to logistic regression. Its recall rises to nearly 0.88 and the AUC passes 0.80, which looks promising at first glance. Still, the model does not treat the two classes equally well, and this unevenness limits its practical value.

The boosting family takes the lead. LightGBM manages to spot almost nine out of ten fraud cases, the highest recall in the group, though this comes at the cost of misclassifying more genuine claims. XGBoost performs in a steadier way: it identifies most fraud while keeping specificity at a healthier level. The AUC of XGBoost is around 0.84, which means that it separates fraud and non-fraud claims more clearly than the other models. What is also notable is that this ability does not disappear after SMOTE; unlike several simpler models, XGBoost preserves its AUC while recall improves, showing robustness across distributions.

Support Vector Machine and K-Nearest Neighbors improve slightly after balancing, but their AUC values stay below 0.75, and they bring only modest gains. They can catch some fraudulent claims, yet their weaker separation makes them less suitable compared to boosting approaches.

In the end, LightGBM reaches the highest recall, but it loses some precision in the process. XGBoost works in a steadier way. It is able to find a large number of fraud cases, but at the same time it does not mislabel as many genuine claims as LightGBM does. The AUC, which is around 0.84, also shows that the model can keep the two groups apart at a fair level. This mix of high recall, reasonable specificity, and stable ranking makes XGBoost the stronger option among the tested methods.

Considering both the balance and the consistency across metrics, XGBoost is selected as the focus for further study. In Section 3.4, feature engineering and SHAP analysis are carried out on XGBoost to better understand how its performance can be improved and explained.

3.5 Feature Engineering: XGBoost-Centered Evaluation

In this section, we first present the results of *manual feature engineering* (FE) applied to XGBoost, followed by the outcomes of *automated feature engineering* (Auto-FE). We then compare three variants of XGBoost (the baseline model with SMOTE, manual FE, and Auto-FE) side by side. The detailed SHAP-based global and local interpretability analysis for these variants is presented in Section 3.5.

3.5.1 Manual Feature Engineering with XGBoost

Manual feature engineering was carried out in several steps. A new variable, *Season*, was created by grouping months into Winter (Dec–Feb), Spring (Mar–May), Summer (Jun–Aug), and Fall (Sep–Nov). Then, extra numeric features were added: squared terms for *Age* and *DriverRating*, plus an interaction term of *Age* with *DriverRating*.

Rationale for Selected Features. The choice of *Season*, *Age*, and *DriverRating* for feature engineering was guided by both domain relevance and methodological practice. Fraudulent behaviour in insurance can vary by season because claim patterns are often linked to weather conditions and driving frequency [47]. Age and driver rating are two of the most common risk factors used by insurers, and extending them with squared and interaction terms allows the model to capture non-linear and joint effects that a simple linear term would miss [49]. Focusing on these variables therefore reflects a balance: they are interpretable and well-established in actuarial research, while still giving scope for richer patterns that can improve model performance.

The data was split into training and test parts (70/30). In the training set, categorical variables were dummy encoded and SMOTE was used to add synthetic fraud cases. The test set went through the same processing and was adjusted to match the training features: missing columns were filled with zeros, extra ones removed, and the order aligned. This way, both training and test data had the same structure before running the model.

The XGBoost model was trained with a 5-fold cross-validation setup, using the ROC score to guide parameter tuning. The hyperparameter grid included variations in

tree depth, learning rate, column sampling, child-weight, and subsampling, with rounds set at either 100 or 200. The best performing configuration combined deeper trees (`max_depth = 6`), a conservative learning rate ($\eta = 0.05$), and moderate sampling rates (`colsample_bytree = 0.7`, `subsample = 0.7`), alongside `nrounds = 200` and `min_child_weight = 1`.

Below, the confusion matrix, ROC curve, and key performance measures for the XGBoost + FE model are presented. The confusion matrix is given in Table 3.37.

Table 3.37: Confusion Matrix — XGBoost with Manual FE

	Pred. 0	Pred. 1
Actual 0	3180	1167
Actual 1	50	226

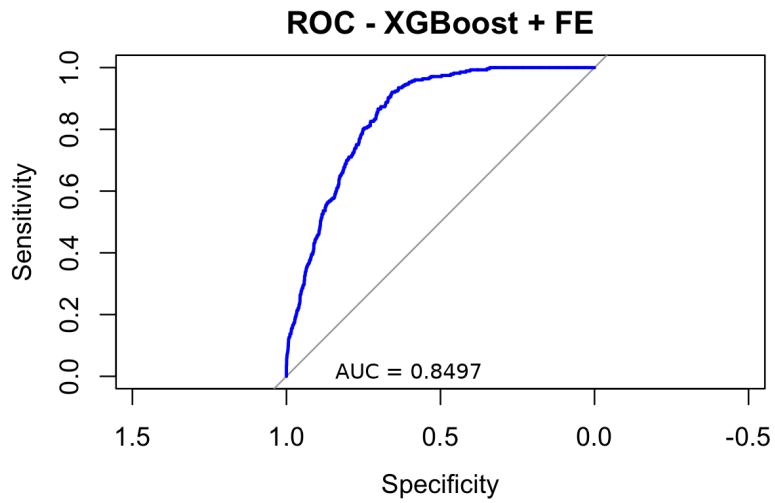

Figure 3.15: ROC Curve — XGBoost with Manual FE

Table 3.38: Test Metrics — XGBoost with Manual FE

AUC (test)	Optimal Thr.	Accuracy	Kappa	Sensitivity	Specificity	Bal. Acc.
0.8497	0.055	0.740	0.195	0.819	0.732	0.776

On the test set, the model reached an AUC of about 0.85. The cutoff that worked best was 0.056, which is much lower than the usual 0.5 because fraud cases are rare and more costly to miss. With this setting, the model caught about 82% of the fraud cases. Specificity was about 0.68, so a share of genuine claims was misclassified. Overall

accuracy was 0.74, while the balanced accuracy was 0.79, indicating that the model gave a more even performance across the two classes.

3.5.2 Automated Feature Engineering (Auto-FE) with XGBoost

In this process, we tested several feature sets for XGBoost and selected the one that gave the best performance in terms of cross-validated ROC AUC. The candidate sets included five alternatives: a baseline without any engineered terms; a version with the seasonal grouping; another with only squared terms for age and driver rating; one that added the interaction of age and driver rating; and finally a full version that combined all of these engineered variables.

To ensure comparability with the manual feature engineering results, the same 70/30 train–test split was used. For each candidate set, preprocessing steps were applied consistently: unused engineered variables were removed, a model matrix with dummy variables was built, missing values were replaced with zeros, and column names were cleaned.

Balancing was applied only to the training portion of the data. We used SMOTE with $K = 5$ and $dup_size = 5$ to generate additional minority observations, while the test set remained in its original form. To make the test data consistent with the training design, we added missing columns as zeros, dropped extra ones, and arranged the columns in the same order.

Model selection was based on cross-validated AUC. Each candidate was trained with `xgbTree` under 5-fold cross-validation, using the ROC score as the metric. We varied the parameters over these values: $nrounds$ {100,200}, max_depth {3,6}, η {0.05,0.1}, $\gamma = 0$, $colsample_bytree$ {0.7,1.0}, min_child_weight {1,5}, and $subsample$ {0.7,1.0}.

The model with the highest CV-AUC was then selected, identified by its name, tuning configuration, and fitted object. This choice was finalised without reference to the test set, avoiding any risk of data leakage.

Finally, the winning model was evaluated on the independent test data. ROC and AUC

were computed on the original class distribution. The Youden J statistic from the ROC curve was used to determine the operating threshold, after which the confusion matrix and test metrics were reported.

Table 3.39: Auto-FE Candidate Combinations — CV AUC

Combination	CV AUC
baseline	0.9761
+Season	0.9765
+PolyOnly	0.9755
+Interaction	0.9754
Full	0.9761

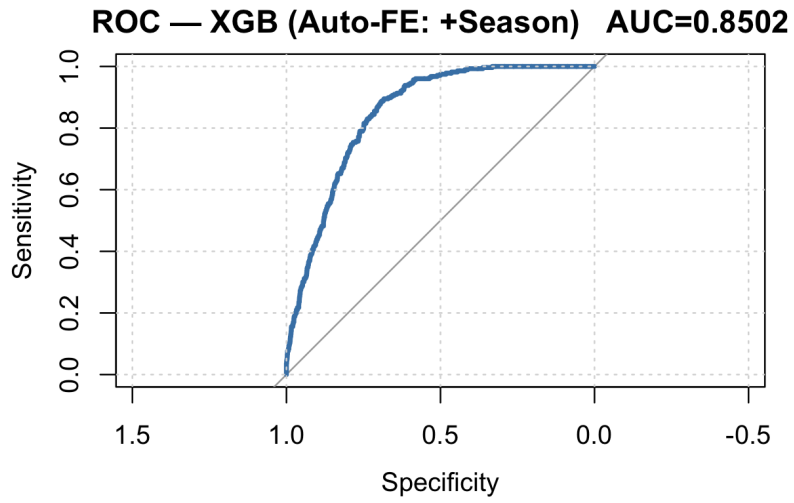

Figure 3.16: ROC Curve — XGBoost with Auto-FE (+Season)

Table 3.40: Test Metrics — XGBoost with Auto-FE (+Season)

AUC (test)	Optimal Thr.	Accuracy	Kappa	Sensitivity	Specificity	Bal. Acc.
0.8502	0.0556	0.703	0.179	0.888	0.691	0.789

The Auto-FE specification with the seasonal grouping (`+Season`) gave the highest cross-validated AUC (0.9765). Although the margin over the baseline (0.9761) and other variants (`PolyOnly` = 0.9755, `Interaction` = 0.9754, `Full` = 0.9761) was small, the gain was consistent and suggested that the seasonal variable added useful information without making the model unnecessarily complex.

On the test data, this configuration reached an AUC of 0.8502, confirming that the slight validation advantage carried over to unseen cases. At the threshold chosen by the Youden J rule (0.0556), the model produced an accuracy of 0.703 and a Cohen's Kappa of 0.179. Sensitivity was high (0.888), showing that most fraudulent claims were detected, while specificity was lower (0.691). The balanced accuracy of 0.789 indicates that the model provided a fair trade-off between capturing fraud and correctly classifying legitimate cases. In short, the seasonal grouping improved discrimination in validation and delivered a recall-oriented performance profile that is valuable for fraud detection.

3.5.3 Comparison of XGBoost Variants

In this section, we bring together the results of three different versions of the XGBoost model to allow for a direct comparison across the main evaluation metrics. These versions are: (i) a baseline XGBoost trained on SMOTE-balanced data without any additional feature engineering, (ii) an XGBoost model enriched with manually designed features based on domain knowledge, and (iii) an XGBoost model combined with automated feature engineering (Auto-FE). By presenting these side by side, the aim is to highlight how the choice of feature construction strategy influences model discrimination, calibration, and overall predictive balance between fraud detection and the correct identification of non-fraudulent claims.

Table 3.41: XGBoost Variants — Side-by-Side Comparison

Variant	Accuracy	Kappa	Sensitivity	Specificity	Balanced Acc.	AUC
XGB (SMOTE)	0.739	0.194	0.826	0.733	0.780	0.841
XGB + FE	0.740	0.195	0.819	0.732	0.776	0.850
XGB + Auto-FE	0.703	0.179	0.888	0.691	0.789	0.850

The three XGBoost variants displayed different strengths. The baseline SMOTE XGB had the highest accuracy (0.739) and specificity (0.733), showing it classified non-fraud cases most reliably, but its sensitivity (0.826) was lower than the other variants.

Manual feature engineering raised sensitivity close to 0.82 while keeping accuracy (0.740) and specificity (0.732) at similar levels, leading to a balanced accuracy of

0.776 and an AUC of 0.850.

The Auto-FE model produced the highest sensitivity (0.888), catching nearly nine out of ten fraud cases. Its specificity (0.691) and accuracy (0.703) were somewhat lower, but the balanced accuracy (0.789) and the AUC (0.850) confirm that it offered a strong recall-oriented profile.

In short, SMOTE XGB favoured overall accuracy and specificity, manual FE maintained a balanced performance across classes, and Auto-FE prioritised sensitivity while preserving the highest AUC together with manual FE.

It should also be noted that the reported values are illustrative. When the models are re–run at different times, the exact numbers may vary slightly due to randomness in resampling and training, although the overall patterns remain consistent. This clarification is provided before moving to the statistical tests of model performance.

3.6 One-Sample DeLong Tests of AUCs

To support the ROC and PR results, we ran the one-sample DeLong test to check whether each model's AUC was higher than 0.5 (random guessing). Across both the original split and the SMOTE-balanced data, all models showed AUC values clearly above 0.5 ($p < 0.001$), meaning that every classifier provides genuine separation ability. The confidence intervals further indicate that performance differences are not due to random variation.

Table 3.42: One-sample DeLong test results for AUC significance across models

Model	**Setting**	**AUC**	**95% CI**	**p-value**
Logistic Regression	Split	0.798	[0.776, 0.819]	< 0.001
Random Forest	Split	0.818	[0.797, 0.839]	< 0.001
XGBoost	Split	0.838	[0.819, 0.857]	< 0.001
LightGBM	Split	0.829	[0.810, 0.849]	< 0.001
SVM	Split	0.750	[0.726, 0.774]	< 0.001
k-NN	Split	0.601	[0.571, 0.632]	< 0.001
Logistic Regression	SMOTE	0.702	[0.665, 0.739]	< 0.001
Random Forest	SMOTE	0.817	[0.795, 0.839]	< 0.001
XGBoost	SMOTE	0.841	[0.823, 0.859]	< 0.001
LightGBM	SMOTE	0.831	[0.813, 0.849]	< 0.001
k-NN	SMOTE	0.649	[0.617, 0.682]	< 0.001
XGBoost + FE (caret CV)	FE	0.850	[0.826, 0.869]	< 0.001
XGBoost (Auto-FE)	FE	0.850	[0.832, 0.874]	< 0.001

The table confirms that all tested classifiers achieve statistically significant AUCs above chance level. Boosting methods (XGBoost, LightGBM) show the highest values, especially under SMOTE and feature engineering, while k-NN remains the weakest but still better than random guessing.

3.7 Pairwise DeLong Tests of AUCs

While the one-sample DeLong tests confirmed that all models performed significantly better than random guessing (Section 3.6), it remains to be seen whether some classifiers consistently outperform others. To this end, we applied the pairwise DeLong test to compare AUC values between models trained on the same split.

Table 3.43: Full pairwise DeLong test results (Cross-split, unpaired)

Model 1	**Model 2**	**AUC$_1$**	**AUC$_2$**	**Diff**	**p-value**	**Winner**
XGB (Split)	KNN (Split)	0.8380	0.6014	0.2366	$\mathbf{< 0.001}$	XGB
LGBM (Split)	KNN (Split)	0.8291	0.6014	0.2277	$\mathbf{< 0.001}$	LGBM
SVM (Split)	KNN (Split)	0.7500	0.6014	0.1486	$\mathbf{< 0.001}$	SVM
XGB (Split)	SVM (Split)	0.8380	0.7500	0.0880	$\mathbf{< 0.001}$	XGB
LGBM (Split)	SVM (Split)	0.8291	0.7500	0.0791	$\mathbf{< 0.001}$	LGBM
XGB (Split)	LR (Split)	0.8380	0.7977	0.0403	**0.0053**	XGB
LGBM (Split)	LR (Split)	0.8291	0.7977	0.0314	**0.0323**	LGBM
XGB (Split)	RF (Split)	0.8380	0.8177	0.0203	0.157	XGB
LGBM (Split)	RF (Split)	0.8291	0.8177	0.0114	0.435	LGBM
RF (Split)	LR (Split)	0.8177	0.7977	0.0200	0.188	RF
SVM (Split)	RF (Split)	0.7500	0.8177	-0.0677	$\mathbf{< 0.001}$	RF
SVM (Split)	LR (Split)	0.7500	0.7977	-0.0477	**0.0035**	LR
XGB (Split)	LGBM (Split)	0.8380	0.8291	0.0089	0.518	XGB

Each row compares two models' AUCs; Diff is AUC$_1$–AUC$_2$. Significant results ($p < 0.05$) are shown in **bold**.

Interpretation (cross-split). The results confirm that boosting methods (XGBoost and LightGBM) remain the top performers, with no meaningful difference between them ($p = 0.518$). Both clearly outperform Logistic Regression, Random Forest, SVM, and k-NN, and these gaps are statistically significant in most cases. Between the baselines, Random Forest and Logistic Regression show similar strength, while SVM is weaker and k-NN falls behind all others. Overall, the tests indicate that boosting consistently dominates, while traditional learners offer less reliable performance.

Previous studies suggest that resampling methods can sometimes change the relative ranking of classifiers when the data are highly imbalanced, as in fraud detection. By re-running the significance tests in the balanced setting, we can check whether the advantage of boosting algorithms comes only from their robustness to skewed data or whether it represents a more general performance edge that also holds when class proportions are equalised.

Table 3.44: Pairwise DeLong test results (SMOTE-balanced data)

Model 1	**Model 2**	**Diff (AUC_1–AUC_2)**	**p-value**	**Winner**
RF	XGB	-0.024	**0.0055**	XGB
RF	LGBM	-0.014	0.127	LGBM
RF	LR	0.115	**<0.001**	RF
RF	KNN	0.168	**<0.001**	RF
RF	SVM	0.079	**<0.001**	RF
XGB	LGBM	0.010	0.086	XGB
XGB	LR	0.139	**<0.001**	XGB
XGB	KNN	0.192	**<0.001**	XGB
XGB	SVM	0.103	**<0.001**	XGB
LGBM	LR	0.129	**<0.001**	LGBM
LGBM	KNN	0.182	**<0.001**	LGBM
LGBM	SVM	0.093	**<0.001**	LGBM
LR	KNN	0.053	**0.0307**	LR
LR	SVM	-0.036	0.132	SVM
KNN	SVM	-0.089	**<0.001**	SVM

Each row compares two models' AUCs (SMOTE-balanced data). Diff is AUC_1–AUC_2; significant results ($p < 0.05$) are shown in **bold**.

Interpretation (SMOTE-balanced). After applying SMOTE, **XGBoost** and LightGBM remain the leading models with the highest AUCs. Their difference is small and not statistically significant (XGB–LGBM: 0.010, $p = 0.086$). **XGBoost**, however, is significantly superior to Random Forest (0.024, $p = 0.005$), Logistic Regression (0.139, $p < 0.001$), SVM (0.103, $p < 0.001$), and k-NN (0.192, $p < 0.001$). LightGBM shows similar advantages over LR, SVM, and KNN (LGBM–LR: 0.129, $p < 0.001$; LGBM–SVM: 0.093, $p < 0.001$; LGBM–KNN: 0.182, $p < 0.001$). Random Forest is clearly stronger than Logistic Regression (0.115, $p < 0.001$), while LR vs. SVM is not significant (-0.036, $p = 0.132$). Overall, boosting methods (with **XGBoost** slightly ahead) dominate, Random Forest stands in the middle, Logistic Regression and SVM are weaker, and k-NN performs worst.

Comparison with LightGBM. As shown in Tab. 3.44, the differences in AUC be-

tween XGBoost and LightGBM are not statistically significant under either regime. Nevertheless, XGBoost achieves slightly higher values in both cases (0.009 in the original split and 0.010 after SMOTE). In the balanced data, XGBoost also secures a clear and statistically significant margin over Random Forest ($p < 0.01$), whereas LightGBM's improvement over Random Forest is only marginal and not significant ($p = 0.127$). These findings suggest that while the two boosting methods achieve broadly similar overall performance, XGBoost shows a steadier edge over the non-boosting models, holding its ground in both the imbalanced and balanced settings.

Table 3.45: Pairwise DeLong test results for XGB variants (SMOTE baseline, manual FE, and Auto-FE)

	XGB (SMOTE)	**XGB + FE**	**XGB + Auto-FE**
XGB (SMOTE)	—	-0.009 ($p = 0.504$)	-0.009 ($p = 0.481$)
XGB + FE	0.009 ($p = 0.504$)	—	0.000 ($p = 0.972$)
XGB + Auto-FE	0.009 ($p = 0.481$)	0.000 ($p = 0.972$)	—

Each cell reports the AUC difference (row minus column) and the p-value. None of the differences are statistically significant ($p > 0.05$).

Interpretation (XGB variants). The pairwise DeLong tests show that there is no statistically significant difference in AUC between the three XGB variants. The observed gaps are very small (at most 0.009) and all p-values are well above the 0.05 threshold. This means that, although the three models differ in accuracy, recall, and specificity, their overall ability to rank fraud versus non-fraud claims is statistically the same. In other words, whether XGB is used in its baseline SMOTE form, with manual feature engineering, or with Auto-FE, the underlying discriminative strength remains comparable.

3.7.1 Computational Cost and Environment

Approximate runtimes were measured for each model; simple learners train within seconds, boosting methods take slightly longer, and feature engineering pipelines—especially automated ones—introduce substantial overhead.

Table 3.46: Estimated computational times for different models

Model	Training Time	Prediction Time	Total
Logistic Regression	∼0.1–0.3 s	∼0.05 s	<0.5 s
Random Forest	∼1–3 s	∼0.3 s	1–4 s
XGBoost	∼3–6 s	∼0.2 s	3–7 s
LightGBM	∼2–4 s	∼0.2 s	2–5 s
SVM (radial kernel)	∼8–15 s	∼1 s	9–16 s
KNN (class::knn)	∼0 s	∼5–10 s	5–10 s
KNN (kknn)	∼0.5 s	∼5–10 s	5–11 s
XGBoost + Manual FE	∼2–5 min	∼0.3 s	2–5 min
XGBoost + Auto-FE	∼9–12 h+	∼0.3 s	9–12 h+

Note: SMOTE balancing increases runtimes (typically 1.5–2×, depending on the model).

Computational environment. All experiments were carried out in R (version 4.5.0, arm64 build) on an Apple MacBook Air (2022) equipped with an Apple M2 processor, 8 GB of RAM, running macOS Ventura (version 13.0). The reported training and prediction times are indicative and may vary depending on hardware specifications, operating system, and R version. Furthermore, the numerical results presented in this thesis are illustrative: rerunning the models may yield slightly different values due to stochastic elements and implementation details, but the overall patterns and conclusions remain consistent.

3.8 SHAP (SHapley Additive exPlanations)–Based Explainability of XGBoost Variants

In this part, we show the SHAP outcomes for each XGBoost version. We start with the *global* level, where features are ranked by importance, and then move to the *local* level, where single cases are explained. The aim of using SHAP is to see both sides: which variables matter the most overall, and how they influence the decision in a given claim. This makes the model's behaviour clearer by linking the general picture with individual examples.

3.8.1 XGB (SMOTE) — Global SHAP

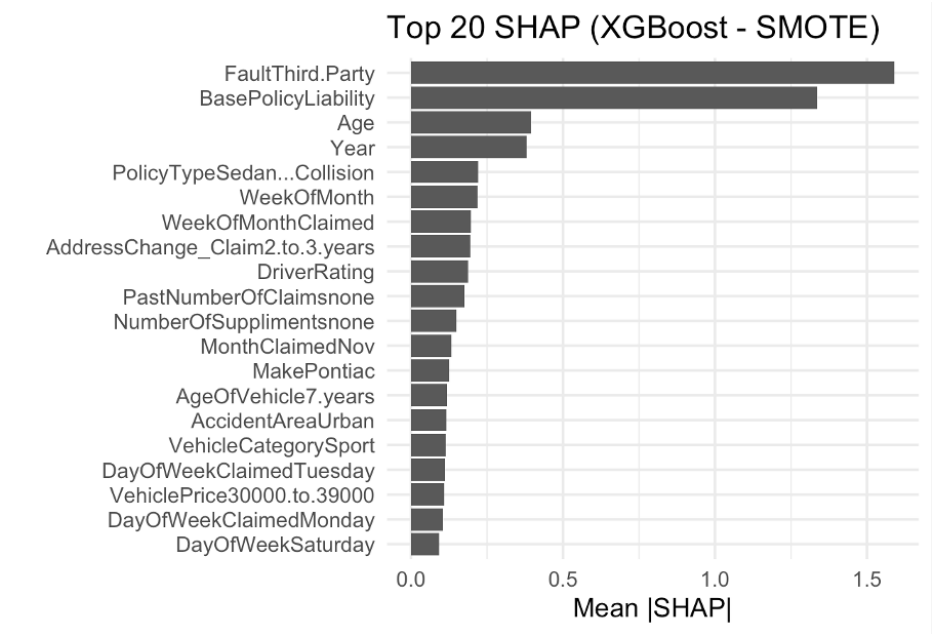

Figure 3.17: Global SHAP Summary — XGB (SMOTE)

The dataset was balanced with SMOTE before training an XGBoost model. On the test set, the model scored an AUC of about 0.84, showing that it separated fraud and non-fraud cases reasonably well.

In the global SHAP analysis, values were computed on the test set using the predcontrib option in XGBoost. For each feature, the mean absolute SHAP value was taken to establish a ranking by average impact. This analysis makes it possible to identify the features that most strongly shape the model's predictions when considered across the entire sample.

The most influential features were FaultThird.Party and BasePolicyLiability. These results indicate that responsibility for the accident and the type of liability coverage played the central role in separating fraud from legitimate claims. Age, policy year, and vehicle type also added influence, showing that the profile of the policyholder and the kind of car insured mattered as well. Timing variables, such as the week or day the claim was reported, together with past claim history, contributed further though at a lower level. Overall, liability and coverage acted as the strongest signals, while demographic, vehicle, and behavioural factors provided additional support.

3.8.2 XGB (SMOTE) — Local SHAP

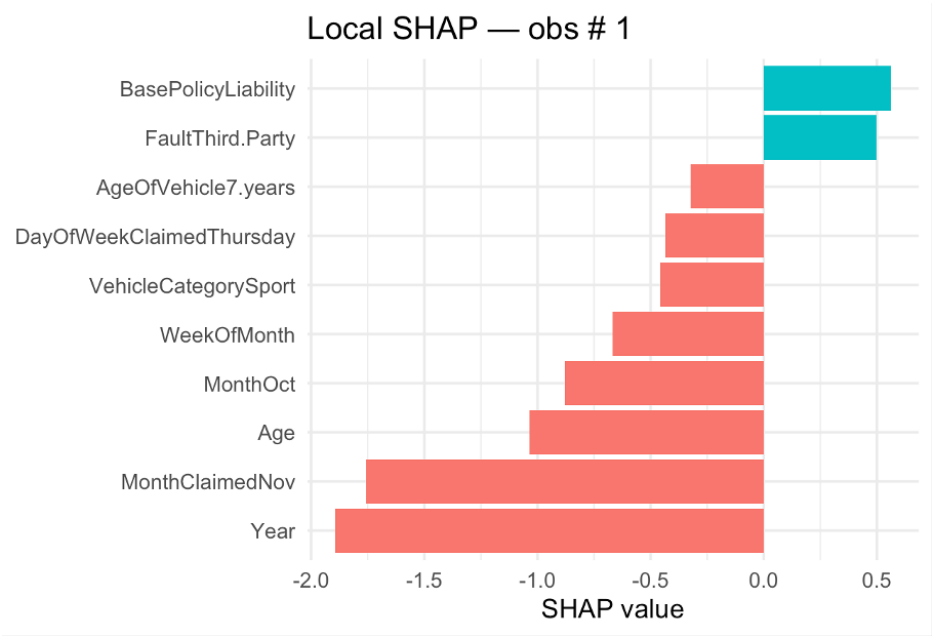

Figure 3.18: Local SHAP Decomposition — XGB (SMOTE)

At the individual level, SHAP values were also inspected for a specific test observation. In this setting, each feature's contribution can be seen directly: positive values push the predicted probability of fraud upward, while negative values pull it downward. Such local explanations complement the global ranking by showing in detail how the model arrives at a single decision.

In this observation, BasePolicyLiability and FaultThird.Party had positive SHAP values, meaning that both the liability type and the fault assignment pushed the prediction upward and signaled a higher likelihood of fraud. The negative SHAP values indicate that factors such as claim timing (Year, MonthClaimedNov, MonthOct) and policyholder profile (Age) pulled the prediction down, lowering the estimated fraud risk.

Smaller effects came from variables like AgeOfVehicle7.years, VehicleCategorySport, and DayOfWeekClaimedThursday, which adjusted the prediction slightly but did not dominate the outcome. The chart indicates that while liability type and fault assignment increased the predicted risk, timing and profile-related features exerted a stronger negative effect, leading the model to assign a lower probability of fraud for this claim.

3.8.3 XGB + Manual FE — Global SHAP

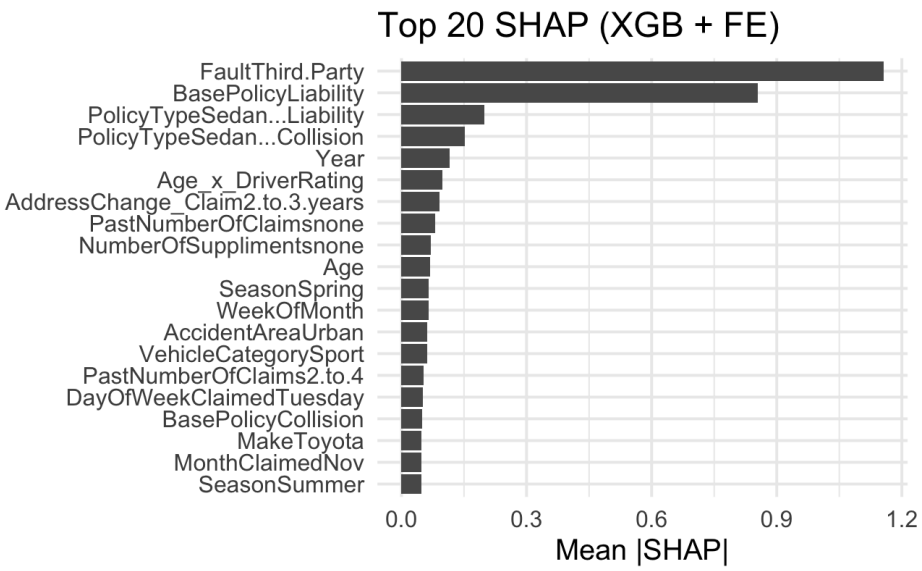

Figure 3.19: Global Feature Importance / SHAP — XGB + FE

From the global SHAP results, two variables dominate the ranking: *FaultThird.Party* and *BasePolicyLiability*. Their average contribution is higher than that of any other feature. In practice, this means the model pays most attention to who is judged to be at fault in the accident and to the type of liability coverage attached to the policy when separating fraud from non-fraud cases. Their contribution is visibly higher than any other variable in the chart.

After these, policy type information also comes forward. Variables such as *PolicyTypeSedan...Liability* and *PolicyTypeSedan...Collision* influence the predictions in a consistent way. Temporal aspects like the *Year* of the claim and indicators of recent address change also appear important, suggesting that both timing and stability of the insured party are relevant for the model's decision process.

A second group of features shows moderate effects. These include driver- and vehicle-related attributes such as the interaction term $Age \times DriverRating$, the claimant's age, the make of the car (e.g., Toyota), and past claim history. They do not dominate the ranking but still shape the output by reinforcing or reducing the signals coming from liability and fault.

Toward the lower end of the list, seasonal factors (*Spring*, *Summer*), accident area, the

month or week of claim, and day-of-week indicators play only a minor role. Their SHAP values remain small, meaning that they adjust the prediction in limited ways rather than driving it. Together, the chart suggests that liability coverage and fault are the backbone of the model, while policy details, demographics, and timing variables add secondary layers of explanation.

3.8.4 XGB + Manual FE — Local SHAP

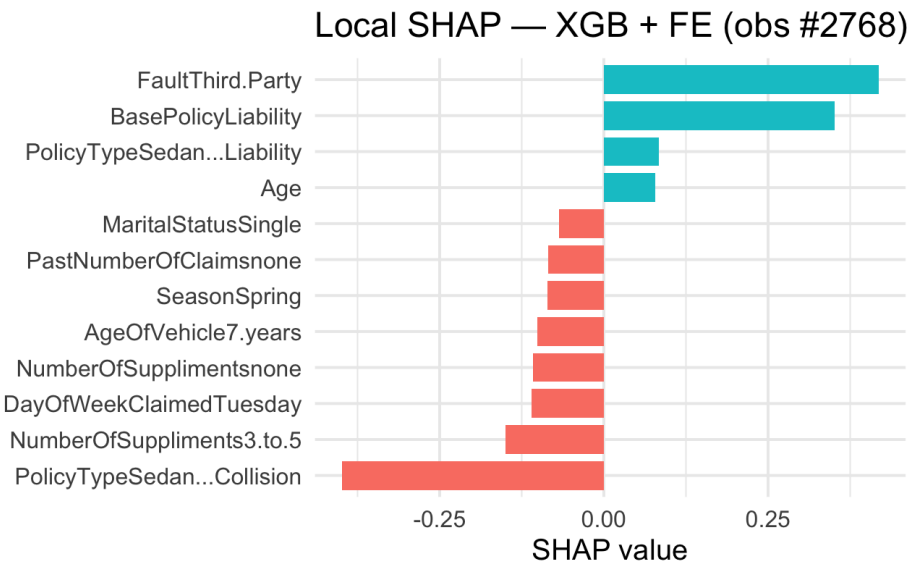

Figure 3.20: Local SHAP — XGB + FE (Example Case)

The local SHAP results show that *FaultThird.Party* and *BasePolicyLiability* had the strongest effect in raising the fraud score. In addition, *PolicyTypeSedan...Liability* and *Age* also contributed positively, though less strongly.

On the other side, several variables reduced the estimated fraud risk. Among these, *PolicyTypeSedan...Collision* had the most pronounced negative effect, followed by indicators such as the number of supplements (*3–5* or none), *DayOfWeekClaimedTuesday*, and *AgeOfVehicle7.years*. Seasonal information (*SeasonSpring*) and policyholder characteristics like marital status and past claim history also added modest downward pulls.

In this claim, liability and fault raised the score, but timing, policy, and profile variables outweighed them, lowering the final fraud estimate.

3.8.5 XGB + Auto-FE — Global SHAP

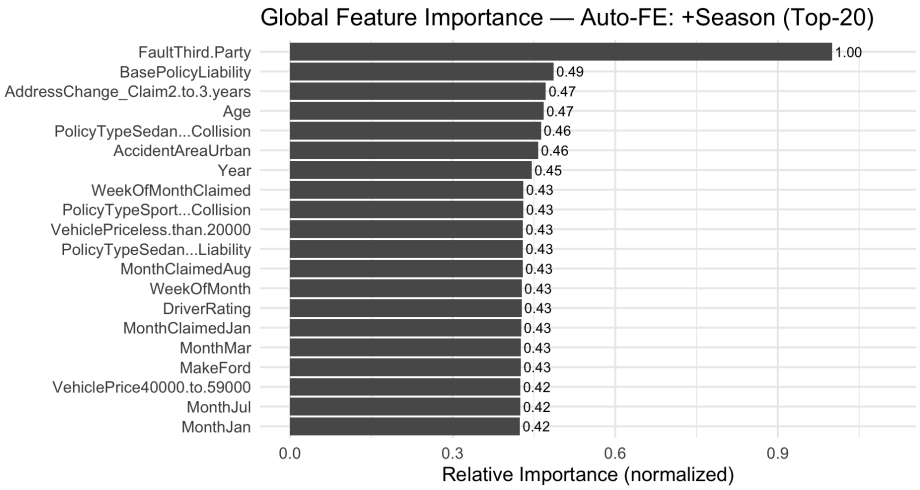

Figure 3.21: Global Feature Importance — XGB + Auto-FE

The SHAP analysis indicates that *FaultThird.Party* stood out clearly as the most influential variable in shaping the model's outputs, followed by *BasePolicyLiability*. This means that both the assignment of fault and the type of liability coverage had a consistent and strong effect on how the algorithm judged fraud risk.

The next group of features included address change within the past 2–3 years, different policy types such as Sedan–Collision and Sport–Collision, as well as age and year. These suggest that recent changes in personal or policy information, combined with demographic characteristics, also carried weight in the decision-making process.

In addition, a number of timing variables appeared in the top 20, such as the month or week when the claim was reported and the season of the year. Their presence shows that the temporal context of a claim was not ignored by the model and contributed meaningfully to its predictions, even if to a lesser degree than liability and fault.

The analysis shows that liability type and fault are the main drivers, while profile and timing variables still add meaningful signals to how the model evaluates fraud.

3.8.6 XGB + Auto-FE — Local SHAP

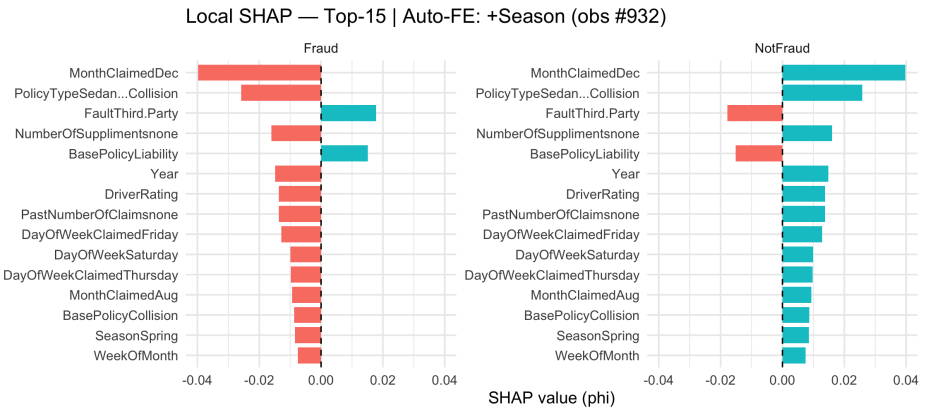

Figure 3.22: Local SHAP — XGB + Auto-FE (Example Case)

For this individual case, the SHAP values show how the model weighed different signals when giving its result. On the left side of the figure, where the case is marked as fraud, the strongest upward effects came from *MonthClaimedDec* and *PolicyTypeSedan...Collision*. In addition, *FaultThird.Party* and *BasePolicyLiability* also pushed the score higher, meaning that the reporting month, collision type, fault assignment, and liability coverage all pointed toward fraud. Smaller increases came from features such as no supplements and no past claims. At the same time, *Year* and driver rating pulled in the other direction, lowering the score and offsetting some of the upward influence. Other policy and claim variables had only minor effects.

On the right-hand side, where the model judged the case as not fraud, the pattern changes. Here, *MonthClaimedDec* gave a strong downward pull that shaped the final outcome. Although *FaultThird.Party* and *BasePolicyLiability* still added pressure upward, they were outweighed by negative contributions from timing and context features such as the claim month, day-of-week, and policyholder history. These factors reduced the score and kept the prediction below the fraud threshold.

This example shows that a single claim can be influenced by both supporting and opposing signals. Liability and fault were steady indicators of fraud, but time-related variables—especially the claim month—shifted the balance the other way. In the end, the model combined structural, behavioural, and timing information, making it possible to see not only why fraud was suspected but also why the case was judged as legitimate.

CHAPTER 4

CONCLUSION

This study examined how machine learning and explainable methods can be used for detecting auto insurance fraud cases. We worked with two datasets: the original, which was imbalanced, and another balanced with SMOTE. This made it possible to see how the models behaved under different class distributions.

The results showed that imbalance is not only a statistical detail but a serious limitation in practice. When almost all claims are non-fraud, a model can look accurate but still fail to identify fraud. In insurance, this weakens the value of the model, because the main goal is to catch those rare cases.

Among the models tested, boosting methods gave the best outcomes. LightGBM was strong in recall and identified more fraud cases, but also produced more false alarms. XGBoost managed to keep a steadier balance between recall and specificity compared to the other models. A key point is that its AUC stayed almost the same after SMOTE, showing that it adapted well when the class distribution changed.

In terms of feature importance, liability coverage and fault assignment were the strongest drivers of predictions. Behind them came factors such as policyholder age, when the claim was reported, and recent updates in policy details. This shows that the model did not rely on a single factor but used several types of information together.

The local SHAP analyses gave further detail. They showed that liability and fault often pushed predictions toward fraud, but time-related factors like the claim year or reporting month could reduce that effect. In some cases, these contextual variables shifted the outcome from fraud to non-fraud. This indicates that the model integrates

structural, behavioral, and temporal information, rather than relying solely on a few "red flag" features.

Overall, the findings point to XGBoost as the most reliable model for practical use. LightGBM was better at finding fraud cases, but XGBoost offered stronger balance and stability across different data settings. With SHAP explanations, its predictions can also be interpreted in a clear way.

For insurers, this means XGBoost is a strong option, provided that imbalance is handled and explanatory tools are applied. For researchers, the study offers a reference point for how ensemble models and interpretation methods can be combined in fraud analytics.

In summary, the work shows that machine learning models, especially XGBoost, can improve fraud detection when paired with interpretation tools. The results highlight both the potential and the limitations of these methods when applied to rare yet costly fraud cases.

REFERENCES

[1] K. Aas, M. Jullum, and A. Løland, Explaining individual predictions when features are dependent: More accurate approximations to shapley values, Artificial Intelligence, 298, p. 103502, 2021.

[2] B. Baesens, *Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques: A Guide to Data Science for Fraud Detection*, John Wiley & Sons, Hoboken, NJ, 2015, ISBN 978-1-118-92094-0.

[3] S. Bansal, Vehicle insurance claim fraud detection, `https://www.kaggle.com/datasets/shivamb/vehicle-claim-fraud-detection`, 2019, accessed: 21 August 2025.

[4] A. Bellet, A. Habrard, and M. Sebban, Metric learning, in *Encyclopedia of Machine Learning and Data Mining*, pp. 1–9, Springer, 2015.

[5] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, When is "nearest neighbor" meaningful?, in *International Conference on Database Theory*, pp. 217–235, Springer, 1999.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[7] R. J. Bolton and D. J. Hand, Statistical fraud detection: A review, Statistical Science, 17(3), pp. 235–255, 2002.

[8] L. Breiman, Random forests, Machine Learning, 45(1), pp. 5–32, 2001.

[9] G. W. Brier, Verification of forecasts expressed in terms of probability, Monthly Weather Review, 78(1), pp. 1–3, 1950.

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, Smote: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research, 16, pp. 321–357, 2002.

[11] T. Chen and C. Guestrin, Xgboost: A scalable tree boosting system, in *Proceedings of the 22nd ACM SIGKDD*, pp. 785–794, 2016.

[12] C. Cortes and V. Vapnik, Support-vector networks, in *Machine Learning*, volume 20, pp. 273–297, 1995.

[13] T. Cover and P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory, 13(1), pp. 21–27, 1967.

[14] J. Davis and M. Goadrich, The relationship between precision-recall and roc curves, in *ICML*, pp. 233–240, 2006.

[15] E. R. DeLong, D. M. DeLong, and D. L. Clarke-Pearson, Comparing the areas under two or more correlated roc curves: a nonparametric approach, Biometrics, 44(3), pp. 837–845, 1988.

[16] S. Derikx, M. de Reuver, and M. Kroesen, A literature survey and classification of electronic fraud detection, Government Information Quarterly, 33(2), pp. 272–281, 2016.

[17] F. Doshi-Velez and B. Kim, Towards a rigorous science of interpretable machine learning, arXiv preprint arXiv:1702.08608, 2017.

[18] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Wiley-Interscience, 2nd edition, 2000.

[19] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, CRC press, 1994.

[20] T. Fawcett, An introduction to roc analysis, Pattern Recognition Letters, 27(8), pp. 861–874, 2006.

[21] J. H. Friedman, Greedy function approximation: A gradient boosting machine, Annals of Statistics, 29(5), pp. 1189–1232, 2001.

[22] J. H. Friedman, J. L. Bentley, and R. A. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Transactions on Mathematical Software (TOMS), 3(3), pp. 209–226, 1977.

[23] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, A survey of methods for explaining black box models, ACM Computing Surveys (CSUR), 51(5), pp. 1–42, 2018.

[24] H. Han, W.-Y. Wang, and B.-H. Mao, Borderline-smote: A new over-sampling method in imbalanced data sets learning, in *Advances in Intelligent Computing*, pp. 878–887, Springer, 2005.

[25] J. A. Hanley and B. J. McNeil, The meaning and use of the area under a receiver operating characteristic (roc) curve, Radiology, 143(1), pp. 29–36, 1982.

[26] H. He, Y. Bai, E. A. Garcia, and S. Li, Adasyn: Adaptive synthetic sampling approach for imbalanced learning, in *IEEE International Joint Conference on Neural Networks*, pp. 1322–1328, 2008.

[27] H. He and E. A. Garcia, Learning from imbalanced data, IEEE Transactions on Knowledge and Data Engineering, 21(9), pp. 1263–1284, 2009.

[28] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, John Wiley & Sons, 2013.

[29] J. M. Kanter and K. Veeramachaneni, Deep feature synthesis: Towards automating data science endeavors, in *Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 1–10, IEEE, 2015.

[30] G. Katz, E. Reif, O. Bendel, A. Shabtai, and L. Rokach, Explorekit: Automatic feature generation and selection, in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 979–984, IEEE, 2016.

[31] S. Kaufman, S. Rosset, and C. Perlich, Leakage in data mining: Formulation, detection, and avoidance, in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*, pp. 556–563, ACM, 2012.

[32] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, in *Advances in Neural Information Processing Systems*, volume 30, 2017.

[33] G. King and L. Zeng, Logistic regression in rare events data, Political Analysis, 9(2), pp. 137–163, 2001.

[34] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, Focal loss for dense object detection, in *IEEE International Conference on Computer Vision*, pp. 2980–2988, 2017.

[35] S. Lundberg and S.-I. Lee, A unified approach to interpreting model predictions, in *Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

[36] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, From local explanations to global understanding with explainable ai for trees, Nature Machine Intelligence, 2(1), pp. 56–67, 2020.

[37] C. Molnar, *Interpretable Machine Learning*, Lulu. com, 2022.

[38] A. Nazábal, P. M. Olmos, Z. Ghahramani, and I. Valera, Handling incomplete heterogeneous data using vaes, Pattern Recognition, 107, p. 107501, 2020.

[39] A. Niculescu-Mizil and R. Caruana, Predicting good probabilities with supervised learning, Proceedings of the 22nd International Conference on Machine Learning, pp. 625–632, 2005.

[40] C. Phua, V. Lee, K. Smith, and R. Gayler, A comprehensive survey of data mining-based fraud detection research, Artificial Intelligence Review, 34(4), pp. 255–288, 2010.

[41] J. Platt, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, Technical report, Microsoft Research, 1999.

[42] M. T. Ribeiro, S. Singh, and C. Guestrin, "why should i trust you?" explaining the predictions of any classifier, in *Proceedings of the 22nd ACM SIGKDD*, pp. 1135–1144, 2016.

[43] T. Saito and M. Rehmsmeier, The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets, PloS One, 10(3), p. e0118432, 2015.

[44] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, 2002.

[45] L. S. Shapley, A value for n-person games, Contributions to the Theory of Games, 2(28), pp. 307–317, 1953.

[46] S. Varma and R. Simon, Bias in error estimation when using cross-validation for model selection, BMC Bioinformatics, 7(1), p. 91, 2006.

[47] S. Viaene, G. Dedene, and K. Vanhoof, Knowledge discovery in a data warehouse for credit card fraud detection, Expert Systems with Applications, 27(3), pp. 643–656, 2004.

[48] S. Viaene, R. A. Derrig, B. Baesens, and G. Dedene, A comparative study of bayesian network classifiers for insurance fraud detection, European Journal of Operational Research, 176(1), pp. 547–569, 2007.

[49] S. Yao, W. Chu, Y. Li, X. Luo, and C. Zhang, Automated feature engineering for credit scoring: a comparative evaluation, Knowledge-Based Systems, 159, pp. 73–91, 2018.

[50] W. J. Youden, Index for rating diagnostic tests, Cancer, 3(1), pp. 32–35, 1950.