Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Statistics

Master's Thesis

Explainability in Machine Learning Models for Churn Prediction

Kenan Arslanbay

© 2025 CZU Prague

Declaration

I declare that I have worked on my master's thesis titled "Explainability in Machine Learning Models for Churn Prediction" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the master's thesis, I declare that the thesis does not break any copyrights.

In Prague on date of submission _____31.03.2025_____

Acknowledgement

I would like to thank Ing. Tomáš Hlavsa, Ph.D. and all other persons for their advice and support during my thesis process.

Explainability in Machine Learning Models for Churn Prediction

Abstract

With the increasing trend in machine learning based AI applications nowadays people have a lack of confidence for the output of AI systems because they do not understand how these systems make decision. Thus, research on explainable AI has become very popular. This thesis introduces an approach to enhance the explainability of black box machine learning models, aiming to clarify how these models utilize features to make predictions. The study begins with a comprehensive review of existing explainability methods, leading to the selection of a suitable approach, such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-Agnostic Explanations) machine learning explainability methods. The model is built using publicly available tabular data, and both local and global explanations are generated to interpret the model's decisions. This thesis enhances the explainability of black-box models by applying SHAP and LIME to interpret feature importance in a predictive model using publicly available tabular data. The thesis highlights how combining high-performing models with robust explainability techniques supports **trustworthy and interpretable AI**, particularly in sensitive decision-making contexts.

Keywords: Model Explainability, Machine Learning, Churn Prediction, Model Interpretation, SHAP Values

Table of content

1) Introduction

In today's rapidly evolving AI landscape, the ability to understand and trust the decisions made by AI systems has become increasingly important. As machine learning models are increasingly integrated into critical domains like finance, healthcare, and law, the need for transparency in these models become crucial. Individuals and organizations must be able to comprehend the reasoning behind AI decisions to ensure fairness, accountability, and reliability.

Explainable Artificial Intelligence (XAI) has emerged as a crucial field of research, addressing the challenge of making complex machine learning models more transparent and understandable. XAI aims to provide insights into the decision-making processes of models, allowing users to see not just the outcome, but also the underlying reason. This transparency is essential not only for building trust but also for improving model performance through error analysis and understanding causal relationships.

Machine learning models can generally be categorized into two types: transparent models, such as linear regression and decision trees, where the decision-making process is easily interpretable; and black-box models, such as artificial neural networks and random forests, where the complexity of the model makes its decisions difficult to understand. Generally, there is a trade-off between model performance and explainability. Transparent models offer higher explainability but may lack in performance, while black-box models often deliver better performance at the cost of interpretability.

This thesis explores the explainability of machine learning models used for churn prediction, specifically focusing on understanding how these models utilize features to make decisions. Churn prediction, which involves prediction whether a customer will leave the bank, is a critical business problem that demands not only accurate predictions but also a clear understanding of

the factors influencing those predictions. This understanding is essential for developing strategies to retain customers and increase income.

The study begins with a comprehensive review of existing explainability methods, leading to the selection of SHapley Additive exPlanations (SHAP) as the most suitable method for detailed analysis. SHAP values offer a unified measure of feature importance that can be used to explain individual predictions (local explanations) as well as overall model behavior (global explanations). By applying SHAP to a binary classification model developed using publicly available tabular data, this research aims to generate both local and global level explanations to interpret the model's decisions.

The focus of this thesis is to evaluate the effectiveness of chosen explainability methond in providing clear and actionable insights into the model's decision-making process. This study not only enhances the transparency of churn prediction models but also contributes to the broader goal of creating more interpretable and trustworthy machine learning systems.

2) Objectives and Methodology

2.1 Objectives

The primary objective is to understand mechanism behind machine learning model predictions by evaluating the features has been used. Thus, the reasoning of black box model will be demystified in human understandable way.

Research of Explainability Methods

Available explainability methods will be explored based on its explainability power, specifically in terms of interpretability, transparency, and trust. In this research, model explainability methods will be explored and applied to improve interpretability, transparency, and trust in machine learning predictions. Specifically, SHAP and LIME two widely-used Python libraries for model interpretation will be implemented and evaluated

Developing Churn Model

A machine learning model will be developed using a publicly available tabular dataset to predict the likelihood of customer churn. This model will assess the probability that each customer will leave the bank, based on various features such as demographics, transaction history, and engagement metrics. The model's predictions will be crucial for identifying at-risk customers.

Evaluation of Effectiveness of Explainability Methods

The effectiveness of each explainability method will be evaluated by examining its ability to provide clear, actionable insights into the model’s predictions. This evaluation will focus on several key criteria, including interpretability (how easily the explanations can be understood), transparency (how well the method reveals the model’s decision-making process), and trustworthiness (the degree to which the explanations inspire confidence in the model’s predictions). Both local and global explanations will be assessed to determine how well each method clarifies the model's behavior at the individual and overall levels.

2.2 Methodology

The objective of this consisting of two parts. Firstly, build robust churn prediction models using logistic regression, decision tree, random forest, and XGBoost; and interpret these models with two state-of-the-art explainability techniques SHAP and LIME to decide which method provides more consistent, intuitive, and actionable insights for business decision-making.

Model Development

Each predictive model is developed not only to estimate churn likelihood but also to provide a basis for comparative explainability. The following sections describe each model briefly.

Logistic Regression

Logistic regression is a statistical model used to examine the relationship between a **qualitative dependent variable** and one or more **independent variables**.

The model computes a **weighted linear combination** of the input features and passes the result through a **logistic (sigmoid) function** to produce a probability value between 0 and 1. This probability represents the likelihood that a given observation belongs to the **positive class**.

The LR model estimates the likelihood of an occurring event based on a set of predictors, using the logistic function:

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n)}}$$

Equation 1

Here, β_0, β_1,, β_n are the model coefficients and x_1,…, x_n are the input features. The logistic function $\sigma(z) = \frac{1}{1+e^{-z}}$ transforms the linear predictor into a probability.

To assess the significance of individual coefficients, **hypothesis tests** are performed for each parameter:

- **Null Hypothesis (H_0):** $[H_0: \beta_i = 0]$
- **Alternative Hypothesis (H_1)**: $[H_1: \beta_i \neq 0]$

The test typically involves:

- Computing the **standard error (SE)** of each coefficient.
- Calculating the **Wald statistic**:

$$z = \frac{\widehat{\beta_\iota}}{SE(\widehat{\beta_\iota})}$$

Equation 2

Finding the **p-value** associated with the zzz-value to determine statistical significance.

If the p-value is less than a chosen significance level (e.g., 0.05), we reject H0H_0H0 and conclude that the predictor has a statistically significant effect on the outcome.

Decision Tree

A decision tree is a supervised learning algorithm that partitions data based on feature values, forming a tree-like structure of decision rules. At each node, the algorithm selects the best split using a measure of impurity or information gain. Common splitting criteria include **Gini impurity**, **entropy** (used in information gain), **information gain ratio**, **classification error** (for classification), and **variance reduction** or **mean squared error (MSE)** (for regression)

$$\text{Gini Impurity} = 1 - \sum_{i=1}^{C} p_i^2$$

Equation 3

$$\text{Entropy} = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

Equation 4

pi is the probability of class i at a given node, and C is the total number of classes. The decision tree algorithm selects splits that minimize impurity (or entropy), resulting in nodes that are as pure as possible.

Random Forest

Random Forest is an **ensemble learning method** that builds upon **decision trees**. Specifically, it creates a **collection (or forest) of decision trees**, each trained on a **random subset of the training data** (via bootstrap sampling) and a **random subset of features** at each split. This randomness helps **reduce variance** and prevent overfitting.

Each individual tree makes a prediction, and the final output of the model is determined by **aggregating the predictions**:

- For classification: by **majority voting**
- For regression: by **averaging the outputs**

While Random Forest uses decision trees as base learners, it is **not itself a single decision tree model**, but rather an **ensemble** that combines the results of many trees to improve generalization performance.

$$\widehat{y_{\mathrm{RF}}} = \frac{1}{T}\sum_{t=1}^{T} \hat{y}_t$$

Equation 5

Formula represents the random forest prediction $\widehat{\boldsymbol{y_{\mathrm{RF}}}}$ as the average of individual tree predictions $\hat{\boldsymbol{y}}_{\boldsymbol{t}}$ in the ensemble.

- $\widehat{\boldsymbol{y_{\mathrm{RF}}}}$: The final prediction made by the random forest model.
- T: The total number of trees in the random forest.
- $\hat{\boldsymbol{y}}_{\boldsymbol{t}}$: The prediction made by an individual tree ttt.

- $\sum_{t=1}^{T} \hat{y}_t$ The sum of predictions from all T trees.
- $\widehat{y_{\mathrm{RF}}} = \frac{1}{T}\sum_{t=1}^{T} \hat{y}_t$ The average of all tree predictions, which serves as the final output of the random forest.

eXtreme Gradient Boosting

XGBoost builds an ensemble of weak learners (usually decision trees) sequentially. Each new tree is constructed to correct the errors of the previous trees by minimizing a regularized objective function, thus improving predictive performance while controlling model complexity.

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Equation 6

In this equation, $l(y_i, \hat{y}_i)$ is the loss function that measures the error between the true value y_i and the prediction $\hat{y}_i$ and $\Omega(f_k)$ is a regularization term for the $(k) - th$ tree f_k. The summation over n data points and k trees ensures that the model minimizes error while avoiding overfitting.

Shap (Shapley Additive Explanations)

SHAP is based on **Shapley values**, a concept from cooperative game theory that fairly distributes contributions among players (features in ML model). The goal is to explain the impact of each feature on model predictions.

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(|N| - |S| - 1)!}{|N|!} [f(S \cup \{i\}) - f(S)]$$

Equation 7

where:

- $\boldsymbol{\phi_i}$ is the Shapley value for each feature,
- **N** is the set of all features,
- **S** is the subset of features excluding i
- $\boldsymbol{f(S)}$ is the model's prediction using only features in S.

LIME (Local Interpretable Model-Agnostic Explanations)

LIME approximates a complex model locally with a simple interpretable model, such as a linear regression, around a specific instance.

LIME selects an explanation by solving the following optimization problem:

$$\xi(x) = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

Equation 8

Where:

- $\boldsymbol{\xi(x)}$ is the local explanation for instance x
- **G** is the set of interpretable models (e.g., linear models, decision trees)
- $\boldsymbol{\mathcal{L}(f, g, \pi_x)}$ is is the loss function measuring how well approximates in the local neighborhood π_x
- $\boldsymbol{\pi_x}$ is a proximity function defining the locality around
- $\boldsymbol{\Omega(g)}$ is a complexity function ensuring the interpretability of g

The Performance Metrics of Classification Models

To determine the effectiveness of machine learning classification models various performance metrics must be analyzed. These metrics help assess the efficiency of the classification approach used and enable a comprehensive evaluation of different models. Relying on a single metric as the sole measure of success would be misleading, making it essential to consider multiple evaluation criteria.

In churn classification, the dataset is usually split into three subsets: training, validation, and test sets. During the training phase, the model learns patterns and relationships from the training data. A separate validation set is then used to fine-tune model parameters and perform hyperparameter tuning, ensuring that the model generalizes well without overfitting. Finally, the test set—comprising data the model has never seen, provides a realistic assessment of its predictive performance. Predictions generated from the test set are compared against the actual

churn outcomes using a confusion matrix, which summarizes true positives, true negatives, false positives, and false negatives. This process helps evaluate the model's accuracy, reliability, and overall performance in predicting customer churn.

Table 1: Confusion Matrix

Confusion Matrix		**Actual Values**	
		1	**0**
Predicted Values	**1**	True Positive TP	False Positive FP
	0	False Negative FN	True Negative TN

Table 1 (Guliyev & Yerdelen, 2021) shows the confusion matrix is explained as follows for a two-category classification model:

True Positive (TP): Represents instances where observations that belong to class 1 are correctly predicted as 1.

True Negative (TN): Refers to cases where observations that belong to class 0 are accurately classified as 0.

False Negative (FN): Occurs when observations that belong to class 1 are mistakenly predicted as 0.

False Positive (FP): Happens when observations that belong to class 0 are incorrectly classified as 1.

The accuracy rate (ACC) is determined by calculating the ratio of correctly classified observations to the total number of samples. This metric evaluates how well the classification

model predicts class 1 when the actual class is 1 and class 0 when the actual class is 0. The accuracy rate can be computed using the following formula:

$$ACC = \frac{TP + TN}{FN + FP + TP + TN}$$

Equation 9

Using a confusion matrix, we can also calculate **sensitivity** and **specificity** rates.

- **Sensitivity** (also known as recall) measures the proportion of correctly classified positive cases (TP) relative to the total actual positive cases (TP + FN). It is calculated as:

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

Equation 10

- **Specificity** represents the proportion of correctly classified negative cases (TN) relative to the total actual negative cases (TN + FP). It is computed as:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Equation 11

Both **sensitivity** and **specificity**, along with **accuracy (ACC),** have values ranging from 0 to 1. A value closer to 1 indicates better model performance. Additionally, there is an inverse relationship between sensitivity and specificity—an increase in sensitivity often leads to a decrease in specificity, and vice versa (Lambert & Lipkovich, 2008)

Another key evaluation metric is **AUC (Area Under the Curve)**, which measures the total area under the **Receiver Operating Characteristic (ROC) curve**. It is widely used to assess model performance alongside the ROC curve (Ben Jabeur et al., 2023) The **AUC** value ranges from 0

to 1, with values closer to 1 indicating a highly accurate model. When the AUC is large, it signifies minimal overlap between the distributions of TN and TP, meaning the model effectively distinguishes between different classes (Mai et al., 2025)

Imbalanced Classification Problems

In customer churn analysis, researchers have consistently observed an imbalanced class distribution in customer datasets. Typically, the number of churned customers is significantly smaller than that of non-churned customers. This imbalance can lead to a situation where overall classification accuracy appears high, but the model performs poorly in predicting churned (minority class) customers.

This issue arises because standard classification algorithms are usually biased toward the majority class (often referred to as the "negative" class), resulting in a higher misclassification rate for the minority class (the "positive" class) (Hoens & Chawla, 2013)

To address this problem, one common approach is resampling the training dataset to balance class distributions before model training. Two popular resampling techniques are:

- Random Undersampling (RUS): This method reduces the size of the majority class by randomly removing its instances, aiming to balance the dataset.
- Random Oversampling (ROS): This technique increases the number of minority class instances by duplicating them to achieve class balance.

While both methods have their advantages, they also come with drawbacks:

- RUS can result in the loss of potentially important data, which might affect the model's ability to generalize. To mitigate this, some RUS methods incorporate heuristics for smarter instance removal.
- ROS may lead to overfitting, as it duplicates existing minority class examples, which can cause the model (especially symbolic classifiers) to create overly specific rules that only apply to the repeated instances.

Another effective solution is the use of ensemble learning methods, particularly tree-based models. These techniques improve performance by combining multiple classifiers and aggregating their predictions to make final decisions (Kuncheva, 2014).

Cost-sensitive ensemble methods go a step further by incorporating misclassification costs into the learning process, focusing on reducing cost rather than altering the base classifiers. Popular ensemble models include Random Forest, AdaBoost, and XGBoost. These models are widely recognized in machine learning for their strong performance across various applications, making them a promising option for handling class imbalance issues (Woźniak et al., 2014)

3) Literature Review

The growing complexity of machine learning (ML) models has underscored the critical need for interpretability, particularly in curicial domains such as healthcare, finance, and criminal justice. This demand has given rise to the field of **Explainable Artificial Intelligence (XAI)**, which seeks to make ML models more understandable, trustworthy, and accountable.

This section provides a comprehensive overview of the literature concerning interpretability and explainability in ML, organized around foundational definitions, evaluation methods, taxonomies, model types, and associated challenges. Additionally, the role of classical data mining models is discussed to contextualize interpretability within traditional and modern modeling paradigms.

Foundational Definitions

Early influential works clarified what *interpretability* and *explainability* mean in (Lipton, 2017) examined the "mythos" of interpretability, dissecting the diverse motivations and notions in the literature.

He identified competing concepts (e.g. **transparency** of models vs. **post-hoc explanations**) and questioned common assumptions (e.g. that linear models are inherently interpretable while complex models are not)

Doshi-Velez and Kim called for a more rigorous science of interpretable ML, advocating formal definitions and evaluation frameworks for explanations– a stance that has guided much subsequent research. (Doshi-Velez & Kim, 2017)

Human-Centered Explanation Theory

Miller bridged XAI with social science, arguing that AI explanations should be informed by how humans *naturally explain* things. (T. Miller, 2018)He reviewed findings from philosophy and cognitive psychology, noting that most XAI research had relied on researchers' intuitions of a "good" explanation rather than human-centered theories. By leveraging concepts like cognitive biases and social context, this work provides a theoretical foundation for designing explanations that align with human expectations.

Interpretability in the Context of Classical Data Mining Models

Before the advent of highly complex machine learning architectures, traditional **data mining** provided the foundation for extracting knowledge from data. Data mining models were designed not only for predictive accuracy but also for **transparency**, **actionability**, and **human-understandable insights**. As the field has evolved, the interpretability strengths of classical models have come into sharper contrast with the opacity of modern black-box systems, prompting the resurgence of interest in explainability.

The Classical Data Mining Process

The data mining pipeline typically follows a structured process, often described using the **CRISP-DM** (Cross-Industry Standard Process for Data Mining) framework:

1. **Business Understanding**
2. **Data Understanding**
3. **Data Preparation**
4. **Modeling**
5. **Evaluation**
6. **Deployment**

Each of these phases carries implications for interpretability:

1. Business Understanding & Data Understanding

At this stage, interpretability is primarily concerned with **feature relevance**, **data quality**, and **descriptive statistics**. Analysts aim to understand the domain context and identify which variables are likely to influence outcomes. Exploratory Data Analysis (EDA) and visualizations play a key role here, offering intuitive explanations even before models are built.

2. Data Preparation

In classical data mining, preprocessing steps such as normalization, encoding, imputation, and feature selection are crucial—and **interpretable by design**. Feature engineering decisions are typically made manually, giving analysts a direct understanding of how inputs are transformed. This contrasts with deep learning models where raw data may be transformed internally and non-transparently via embeddings or learned representations. In modern XAI, recovering interpretability often involves reverse-engineering these transformations.

3. Modeling: Classical Models and Inherent Interpretability

The **modeling phase** is central to discussions on interpretability. Classical data mining relied heavily on **white-box models** that were both powerful and understandable:

- **Decision Trees**: Offer flowchart-like explanations. Each path from root to leaf corresponds to a decision rule.
- **Linear and Logistic Regression**: Provide coefficients that quantify the effect of input features.
- **Naïve Bayes**: Though probabilistic, this model provides interpretable conditional probabilities.
- **Association Rule Mining (e.g., Apriori, FP-Growth)**: Generates clear "if-then" rules useful for recommendation and market analysis.
- **Clustering (e.g., K-means)**: Offers groupings of similar instances, often visualized in low-dimensional space.

These models provide **global interpretability**, where the entire model logic is accessible, rather than relying on local post-hoc explanations.

4. Evaluation

Interpretability in evaluation goes beyond performance metrics like accuracy or AUC. Classical data mining models enable the analyst to **trace predictions back to model components**, providing justifications.

For example:

- A decision tree may show that a prediction was made because *"income > $50K and age < 40"*.
- In linear regression, the prediction for a target variable can be decomposed into additive contributions of each feature.

Modern interpretability tools like **SHAP** and **LIME** attempt to replicate this evaluation transparency for black-box models.

5. Deployment

In deployment, interpretability affects:

- **Trust and accountability** (can stakeholders justify decisions to users or regulators?)
- **Maintenance** (can developers debug unexpected predictions?)
- **Monitoring** (can shifts in model behavior be understood over time?)

Traditional models are easier to maintain in production due to their predictable and transparent nature. In contrast, deep learning systems may require ongoing interpretability audits using post-hoc tools.

Limitations of Classical Data Mining Models

While classical models are inherently interpretable, they often struggle with:

- Capturing **nonlinear relationships**
- Handling **high-dimensional** or **unstructured data** (e.g., images, text)
- Achieving **state-of-the-art performance** on large-scale tasks

This trade-off between **accuracy** and **interpretability** motivated the development of more complex models. However, the loss of transparency has necessitated a reintroduction of interpretability via **XAI techniques**.

Bridging the Gap: Hybrid Approaches

Modern approaches increasingly seek to combine the interpretability of classical models with the performance of black-box models:

- **Surrogate Models**: Using decision trees or linear models to approximate the behavior of black-box models locally or globally.
- **Hybrid Pipelines**: Using interpretable models for feature selection or preprocessing before passing data to a complex model.
- **Rule Extraction**: Deriving symbolic rules from trained neural networks or ensembles.

This emerging synthesis suggests that the lessons from classical data mining are still deeply relevant and can guide the design of transparent, accountable, and effective AI systems.

Explainability and Interpretability

The concepts of interpretability and explainability in machine learning have been the focus of significant research, especially as machine learning models become increasingly complex and opaque. While these terms are often used interchangeably, there are subtle distinctions between them that researchers have sought to clarify.

Interpretability generally refers to the ability to understand the cause of a decision made by a model, often in intuitive or human-understandable terms. Interpretability as "the ability to explain or to present in understandable terms to a human" (Doshi-Velez & Kim, 2017) while Miller describes it as "the degree to which a human can understand the cause of a decision" (C. A. Miller, 2021)However, both definitions, while intuitive, lack the mathematical rigor that is often desired in technical fields

On the other hand, explainability relates more closely to the internal workings of a model how the model processes inputs to generate outputs. A model can be interpretable without being fully explainable; that is, one might understand the output without comprehending the underlying processes.

As Gilpin says We take the stance that interpretability alone is insufficient. In order for humans to trust black-box methods, we need *explainability* – models that are able to summarize the reasons for neural network behavior, gain the trust of users, or produce insights about the causes of their decisions. While interpetabil-ity is a substantial first step, these mechanisms need to *also* be complete, with the capacity to defend their actions, provide relevant responses to questions, and be audited. Although interpretability and explainability have been used interchangeably, we argue there are important reasons to distinguish between them. (Gilpin et al., 2018)

Evaluation of Interpretability

Doshi-Velez and Kim categorize the evaluation methods for interpretability into three types: application-grounded, human-grounded, and functionally grounded evaluations. Application-grounded evaluation focuses on how interpretations affect specific tasks or applications, such as error identification or reducing discrimination in outputs. (Doshi-Velez & Kim, 2017)

Human-grounded evaluation involves non-expert users and assesses the general quality of interpretations, independent of specific applications. Functionally grounded evaluation, the most formal of the three, does not involve human subjects but relies on mathematical definitions to evaluate interpretability. This method is particularly useful when ethical concerns prevent human testing or when methods are not mature enough for practical deployment.

Taxonomies

Several attempts have been made to classify and define interpretability and explainability methods in machine learning. Gilpin proposed a taxonomy focusing on deep learning, categorizing methods based on their approach to processing data, explaining internal representations, and the self-explanatory nature of certain networks. (Gilpin et al., 2018) A dadi and Berrada worked by analyzing a broad range of literature, stressing the need for more formalism and human-machine interaction in the field of explainable AI. They identified a trend

where explainability is often considered only in terms of modeling, advocating for a broader exploration that includes other aspects of machine learning. (Adadi & Berrada, 2018)

Guidotti further refined the taxonomy by categorizing methods based on the type of problem they address explaining, inspecting, or creating transparent black-box models. The work also highlighted the ongoing challenges in developing formal metrics for evaluating interpretability and the need for better methods in fields like recommender systems. (Guidotti, 2024)

Murdoch introduced the Predictive, Descriptive, Relevant (PDR) framework, proposing three types of metrics—predictive accuracy, descriptive accuracy, and relevancy to evaluate interpretability methods. Their work emphasized the potential of post-hoc interpretability to improve predictive accuracy, suggesting that a combination of transparent models and post-hoc methods might be ideal in some cases. (Murdoch et al., 2019)

Arrieta provided a comprehensive taxonomy distinguishing between transparent models and post-hoc interpretability methods, particularly for deep learning. Their study also introduced the concept of Responsible Artificial Intelligence, proposing criteria for implementing AI in organizations to ensure ethical and transparent practices. (Barredo Arrieta et al., 2020)

Different Scopes of Machine Learning Interpretability: A Taxonomy of Methods

There are various perspectives when examining the evolving landscape of machine learning interpretability methods—such as the type of data they handle or whether they offer local or global explanations. The classification of these techniques should not be approached from a single viewpoint. Instead, multiple dimensions exist that can be used to distinguish and further categorize interpretability methods. To select the most suitable method for a given problem, practitioners must consider all relevant aspects of each approach.

One particularly important distinction is based on the type of algorithms a method can be applied to. If a method is limited to a specific family of models, it is referred to as *model-*

specific. Conversely, methods that can be applied across a wide range of algorithms are known as *model-agnostic*. Another key dimension involves the *scale of interpretation*: methods that explain individual predictions are considered *local*, while those that provide insight into the entire model are labeled *global*. Furthermore, the *type of data* a method supports is a critical consideration. While most interpretability methods are designed for tabular or image data, there are also techniques tailored to text data.

Figure 1 illustrates a mind map that summarizes these various dimensions, helping practitioners navigate the taxonomy of interpretability methods. Taking all of these factors into account is essential to selecting the most appropriate method based on the specific needs of a given task.

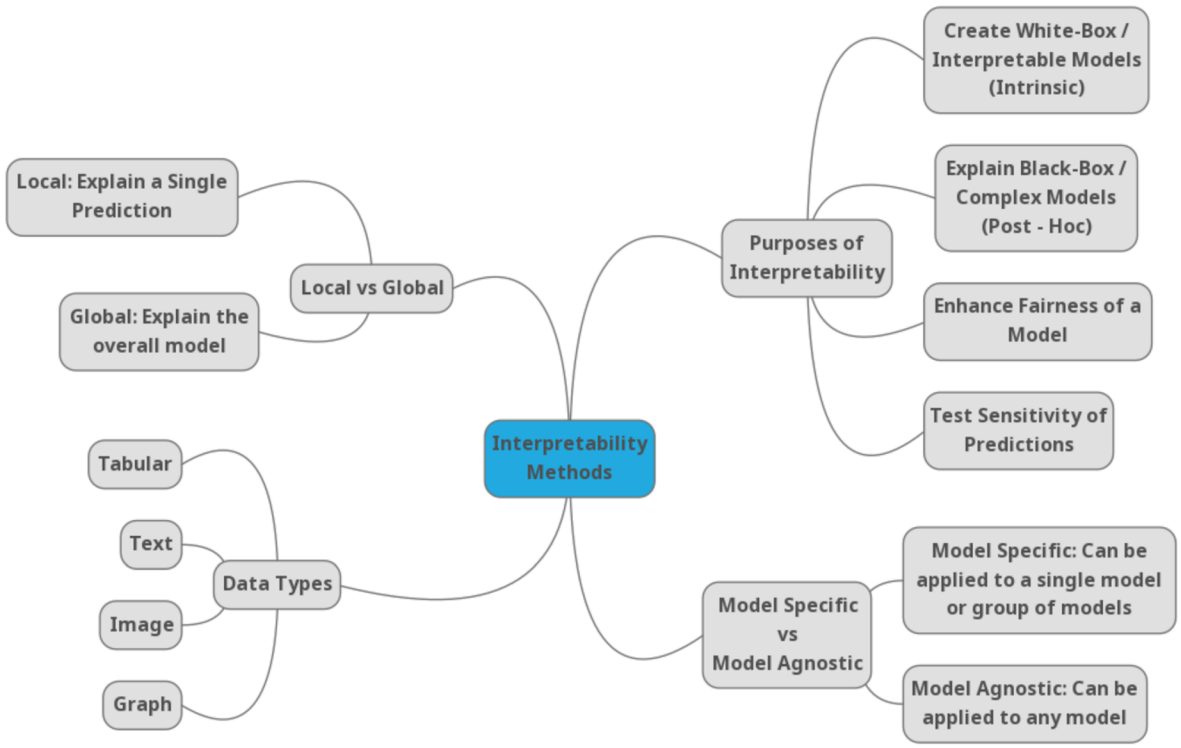

Figure 1: mind-map of Machine Learning Interpretability Techniques.(Lambert & Lipkovich, 2008)

Data Type Considerations

Tabular data:

Many interpretability techniques (Hassija et al., 2023) have been developed with structured, tabular data in mind. Features are usually well-defined, and methods like feature importance scores or rule extraction can be directly applied. The clarity of feature representation in tabular data often makes it easier to derive meaningful interpretations.

Image data:

For image-based models, interpretability methods often focus on visual cues. Techniques such as saliency maps, Grad-CAM, or occlusion-based methods help identify which regions of an image are most influential in driving a particular prediction. These visual explanations are particularly useful for applications in medical imaging, autonomous driving, or any domain where spatial information is critical.

Test data:

Text data presents unique challenges due to its unstructured nature. Interpretability methods for text often involve highlighting key words or phrases that contribute most significantly to a model's decision. Attention mechanisms in transformer models naturally provide a form of interpretability by showing which tokens the model focused on. Additionally, adaptations of model-agnostic methods (like LIME) can be used to perturb text inputs and assess their impact on predictions, thus revealing the most critical textual components.

Interpretability Methods

Interpretability methods (Viswan et al., 2023) can be categorized based on their application to specific algorithms (model-specific) or their general applicability across different models (model-agnostic). They can also be classified by the scale of interpretation, whether local (explaining individual predictions) or global (explaining the entire model). A particularly significant category includes methods designed to interpret complex black-box models, often through post-hoc analyses. Techniques like saliency maps, integrated gradients, and Layer-wise

Relevance Propagation (LRP) have been developed to provide visual or numerical insights into how models, particularly deep neural networks, arrive at specific decisions.

In addition to these, model-agnostic methods like **LIME**, **SHAP** and **Anchors** have gained popularity for their flexibility in explaining predictions from any black-box model. These methods often simulate the behavior of the model around specific instances to generate interpretable explanations. However, they come with limitations, such as instability in generated interpretations or computational inefficiencies in high-dimensional spaces.

Black Box Models

Black-box models refer to machine learning models that are highly complex, making it difficult or impossible for humans to understand the internal workings and decision-making processes. These models often involve intricate structures with numerous parameters and layers, such **as deep neural networks**, ensemble methods like **random forests**, and **support vector machines**. The term "black-box" highlights the opacity of these models: while they can produce highly accurate predictions, the rationale behind these predictions is not readily accessible or understandable.

Examples of Black-Box Models:

Deep Neural Networks (DNNs):

These models (Hassija et al., 2023) are perhaps the quintessential black-box models. DNNs, particularly those with many layers (known as deep learning models), are capable of learning highly complex patterns from large datasets. For example, convolutional neural networks (CNNs) are used extensively in image recognition tasks, where they can identify objects within images with remarkable accuracy. However, understanding why a particular CNN identifies a specific object in an image is challenging, as the model's decisions are based on abstract features extracted across multiple layers.

Random Forest:

A random forest is an ensemble method that combines multiple decision trees to improve predictive performance (Breiman, 2001). While each individual decision tree may be relatively interpretable, the aggregation of many trees into a random forest creates a model that is difficult to interpret. The combined effect of multiple trees, each considering different subsets of features, results in a model where understanding the contribution of each feature to the final decision is non-trivial.

Support Vector Machines (SVMs):

SVMs are powerful models used for classification and regression tasks (Suthaharan, 2016). When kernel functions are used to map input data into high-dimensional spaces, the resulting decision boundaries become highly complex and difficult to interpret. The transformation of data through kernels creates decision surfaces that are abstract and not easily interpretable by humans.

Methods for Explaining Black-Box Models

Black-box models are often difficult to interpret, so a range of techniques have been developed to help explain their behavior after they've been trained and are making predictions (Hassija et al., 2024).

Grad-CAM (Gradient-weighted Class Activation Mapping):

Grad-CAM is a technique used to generate visual explanations for predictions made by CNNs. It does this by using the gradients of any target concept, flowing into the final convolutional layer, to produce a coarse localization map of the important regions in the image. This map highlights the areas of the image that are most relevant to the model's prediction, providing insights into which parts of the image the model focuses on when making decisions.

SHAP (Shapley Additive Explanations):

SHAP values provide a way to explain the output of any machine learning model by distributing the prediction among the input features. It is based on cooperative game theory and assigns an importance value to each feature based on its contribution to the prediction. SHAP can be applied to a wide range of models, making it a versatile tool for interpreting black-box models (Winter, 2002).

Shap includes several specialized submodules tailored to different model types and interpretability needs. These submodules apply the same theoretical foundation—Shapley values from cooperative game theory but differ in how they approximate or compute those values, depending on the underlying model. Below is an overview of the key SHAP submodules used in practice.

Tree Shap

It is designed specifically for tree-based machine learning models such as XGBoost, LightGBM, CatBoost, and random forests. It provides exact computation of SHAP values by leveraging the structure of decision trees, resulting in both accurate and computationally efficient explanations. This method can handle large ensembles and provides detailed insight into how each feature contributes to a specific prediction or to overall model behavior. Tree SHAP is especially useful in applications like credit risk modeling or fraud detection, where transparency and speed are essential (Sharma et al., 2020).

Kernel Shap

It is a model-agnostic explainer that can be used with any predictive model, including black-box algorithms like support vector machines or neural networks. It works by approximating SHAP values through a game-theoretic approach using weighted linear regression on a series of perturbed samples. While Kernel SHAP is theoretically grounded and versatile, it is computationally expensive, especially as the number of input features increases. Nonetheless, it is a powerful tool when dealing with models where internal structure is not accessible (Roshan & Zafar, 2022).

Deep Shap

It is a variant of SHAP optimized for deep learning models, particularly those built with TensorFlow or Keras. It combines ideas from SHAP and DeepLIFT, using a modified backpropagation algorithm to estimate feature attributions. Deep SHAP is more computationally efficient than Kernel SHAP and is better suited to deep neural networks with complex architectures. This method is commonly used to interpret models in computer vision or natural language processing by highlighting the most influential input components, such as image regions or words in a sentence(Temenos et al., 2023).

Linear Shap

It is tailored for linear models like logistic regression and linear regression. It computes exact SHAP values by directly analyzing the linear model coefficients, making it both fast and precise. Since linear models are inherently interpretable, SHAP adds value by attributing predictions to individual features in a theoretically consistent way, reinforcing transparency in domains such as healthcare analytics or customer churn prediction(Mosca et al., 2022) .

Partition Explainer

It offers an approximate alternative for tree-based models by using a hierarchical clustering of feature dependencies. This approach simplifies the computation by grouping features and estimating their combined contributions, resulting in faster explanations when dealing with high-dimensional data. While not as exact as Tree SHAP, the Partition Explainer strikes a practical balance between speed and interpretability, especially when approximate insights are acceptable or when Tree SHAP becomes computationally burdensome (Juneja et al., 2024) .

LIME (Local Interpretable Model-agnostic Explanations):

LIME is a model-agnostic technique designed to explain the predictions of complex, black-box machine learning models by approximating them with interpretable models in a local neighborhood around the prediction of interest. The central idea is that while the global behavior of a model may be too complex to interpret directly, its behavior in the vicinity of a single prediction can often be approximated by a simpler, more transparent model. To achieve this, LIME perturbs the original input data by making small changes to the feature values and then observes how these perturbations affect the output of the black-box model. It then uses these synthetic samples and their corresponding predictions to fit a sparse linear model such as a linear regression or decision stump that approximates the black-box model's decision boundary near the specific instance.

This local surrogate model highlights which features contributed most to the prediction and in what direction, offering insight into the model's reasoning. LIME is particularly useful for tabular data, text classification, and image classification tasks. For instance, in a text classification task, LIME can identify specific words in a sentence that most influenced the prediction. One of the key advantages of LIME is its flexibility it can be applied to any model type without requiring access to internal model parameters. However, its explanations are

inherently local and may not generalize across different instances, meaning that multiple explanations are often needed to understand the model more broadly.

Other Notable Tools and Frameworks

Beyond the major libraries above, there are other open-source explainability tools relevant to tree-based models:

Partial Dependence and ICE Plots (scikit-learn/PDPlot):

While not a separate library, scikit-learn provides functions to calculate **partial dependence plots (PDP)** and individual conditional expectation (ICE) for any regressor, with optimized support for tree-based models via the "recursion" method PDPs show the average effect of a feature on the predicted outcome (global explanation), which is useful for understanding feature influence in ensembles. Tools like **PDPbox** (Python library) and built-in sklearn.inspection.plot_partial_dependence make it easy to integrate these plots into model interpretation workflows. They complement per-instance methods like SHAP/LIME by giving a global view of feature behavior. Scalability is good for tree models since the tree structure can speed up calculation.

Visualization is typically a simple line plot (for PDP) or multiple lines (for ICE) showing how the prediction changes as you vary one or two features.

AI Explainability 360 (AIX360):

Developed by IBM, AIX360 is an extensible toolkit that includes a **collection of algorithms** for explainability. It covers a wide range of methods (beyond just feature attributions), such as contrastive explanations, rule-based global explanations, and fairness metrics. For tree ensembles, AIX360 doesn't provide a single specific tool like TreeSHAP, but it includes generic methods and some research-oriented techniques that can be applied. It supports **tabular, text, image, and time-series data**, and even has an **interactive GUI** for step-by-step use cases . AIX360 is more of a "library of many XAI algorithms" – powerful but with a steeper learning curve if you're just looking to explain boosted trees. It is actively maintained by the Trusted-AI community at IBM, though its adoption in mainstream Kaggle-style workflows is lower.

This toolkit is useful if you need advanced or specialized explainers (for example, boolean rule lists or counterfactual examples in addition to feature importance).

OmniXAI:

Introduced in 2022, OmniXAI (from Salesforce research) aims to be a **one-stop explainability library** with a unified interface. It supports multiple data types (tabular, image, text, time-series) and both **model-specific and model-agnostic methods** for explanation. For tree models in particular, OmniXAI can generate feature attributions (likely leveraging SHAP or similar under the hood) and partial dependence, as well as counterfactual explanations. It provides an easy API where, for instance, a TabularExplainer will automatically pick appropriate techniques for your model type. A notable feature of OmniXAI is its **dashboard**: with a few lines of code you can launch a GUI that lets you explore different explanation types side by side. This can be very useful for analyzing tree model predictions with multiple lenses (e.g., feature importance, what-if analysis, and counterfactual examples in one place). Being a newer toolkit, OmniXAI is actively developed but still growing in community adoption (it has fewer stars compared to SHAP or LIME). It appeals to practitioners who want a consolidated tool rather than juggling many libraries.

Shapash:

Shapash is a library focused on **making SHAP values more accessible** and presentation-ready. It acts as a wrapper over SHAP (and LIME) to produce clear, "story-telling" visualizations with **explicit labels and summaries** for each feature's influence.

Shapash works well with tree-based models, simplifying the creation of dashboards or reports that non-data-scientists can understand. For example, it can produce a plot or HTML report saying: *"For this prediction, **Age** had a high impact increasing the score, while **Income** and **Debt** decreased the score"* in plain language. The visual capabilities include decision plots and contribution plots with legends that are easy to interpret. Shapash is actively maintained (originating from an insurance company's data science team) and is used when **explainability needs to be delivered to business stakeholders**. It's slightly less flexible than raw SHAP, but much more user-friendly for communication purposes.

What-If Tool (WIT):

The What-If Tool is an interactive visual tool (by Google PAIR) that can be used in Jupyter or TensorBoard environments to probe model behavior. While not specific to tree models, WIT can load a tree ensemble model and allow users to do **counterfactual what-if analysis** (manually adjusting feature values to see how predictions change), view partial dependence-like plots, and check fairness metrics. It supports integration with SHAP and LIME to highlight feature importance as well

WIT is especially useful for analyzing *individual instances or small datasets in detail* – for example, you can select a mispredicted example and tweak inputs to see how to flip the prediction. Its strength is the interactive UI rather than algorithmic novelty. For developers focusing on explainability in web apps or reports, WIT provides a no-code interface to explore tree model decisions, which can complement programmatic libraries like those above

White Box Models

In contrast to black-box models, white-box models are inherently interpretable and transparent. These models are designed in such a way that their decision-making process is clear and understandable to humans. The internal structure of white-box models allows users to trace how inputs are processed and transformed into outputs, making it easier to understand the logic behind predictions.

Examples of White-Box Models:

Linear Regression: Linear regression is a classic example of a white-box model. It models the relationship between input features and the target variable as a linear equation. The coefficients of the equation represent the weight or importance of each feature in making predictions. This straightforward relationship makes it easy to understand how each input affects the output, ensuring high interpretability.

Decision Trees: Decision trees are another example of white-box models. They break down a decision-making process into a series of binary decisions, where each node represents a feature, and the branches represent the outcomes of decisions based on that feature. The path from the root to a leaf node provides a clear explanation of how the input features lead to a particular decision or prediction. This hierarchical structure allows for easy interpretation and explanation of model predictions.

Rule-Based Systems:

Rule-based systems consist of a set of "if-then" rules that define how inputs should be transformed into outputs. These systems are inherently interpretable because each rule corresponds to a specific decision path, which can be easily understood and communicated. For example, a medical diagnosis system might use rules such as "If the patient has a fever and a cough, then diagnose flu," which can be easily followed and explained.

The following figure illustrates the differences between Black box and White box model.

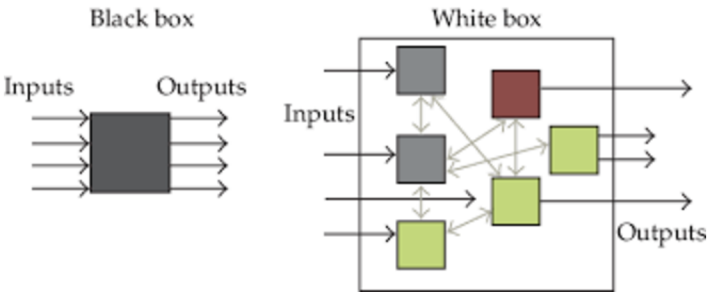

Figure 1: Black box vs White box (Volovoi, 2012)

The Issue of Model "Churn" in Interpretability

In the context of machine learning interpretability, churn refers to the **instability of model explanations over time**. This occurs when feature attributions, local explanations, or global model behaviors change significantly between retrainings or slight modifications of the input data. Churn can undermine trust in AI models, particularly in critical applications like finance, healthcare, and legal decision-making, where consistent and reliable explanations are necessary.

Even if a model maintains high predictive accuracy, fluctuations in explanations can lead to confusion among users, regulatory challenges, and difficulties in debugging model behavior. Without stable interpretability, stakeholders including data scientists, domain experts, and end-users may struggle to rely on model outputs for long-term decision-making.

Challenges and Future Directions

Although, the advances in interpretability and explainability, several challenges remain. The lack of formal, universally accepted definitions and metrics for evaluating interpretability continues to be a significant hurdle. Additionally, while many methods exist, there is no one-size-fits-all solution. Different methods are better suited to different types of models, data, and applications. The field is moving towards combining different interpretability techniques to create more robust and comprehensive explanations.

Moreover, the balance between model complexity and interpretability is a recurring theme in the literature. As models become more complex, achieving transparency without sacrificing performance becomes increasingly difficult. Future research may focus on developing hybrid models that maintain high predictive power while offering sufficient interpretability or on creating new methodologies that integrate multiple interpretability techniques to cover different aspects of a model's decision-making process.

4) Practical Part

4.1 Dataset

The dataset used in this project was sourced from Kaggle. The dataset consists of 10000 rows, a binary outcome class and 13 features, including demographic variables, credit score given, and data of customer's past interaction with the bank. The binary dependent variable is whether the customer has churned (1 for yes, 0 for no) or not.

The data is publicly available on kaggle (Dhakad, 2022)

Variables used in the data summarized in Table 1.

Table 2: Definition of Variables

Feature	Description	Feature Type
Customer ID	A unique identifier for each customer	Numerical
Surname	The customer's surname or lastname	Categorical
Credit Score	A numerical value representing customer's credit score	Numerical
Geography	The country where the customer resides	Categorical
Gender	The customer's gender whether male or female	Categorical
Age	The customer's age in years	Numerical
Tenure	The number of years the customer has been with the bank	Numerical
Balance	Average balance of customer in $	Numerical
NumOfProducts	The number of bank products the customer uses (e.g., savings account, credit card)	Numerical
HasCrCard	Whether the customer has a credit card	Numerical
IsActiveMember	Whether is the customer is an active member	Numerical
EstimatedSalary	The estimated salary of the customer in $	Numerical

Exited	Whether the customer has exited the bank (Target Variable)	Numerical

4.2 Descriptive Statistics of Variables

Table 2 summarizes the descriptive statistics for all variables examined in this study. In terms of geographical distribution, 50.1 % of the customers are from France, while 25.1 % and 24.8% are from Germany and Spain, respectively. Looking at the outcome variable, 20.4 % of customers have exited the bank, compared to 79.6 % who remain. On average, customers use 1.5 products (SD = 0.6), with the number ranging from 1 to 4. The gender distribution is balanced with 54.6 % males and 45.4 % females. Furthermore, the variable indicating possession of a credit card (HasCrCard) shows a mean value of 0.7 (SD = 0.5) on a binary scale (0 or 1). Customers have an average credit score of 650.5 (SD = 96.7), with scores spanning from 350 to 850. The average tenure with the bank is 5 years (SD = 2.9), ranging between 0 and 10 years. In addition, the estimated salary averages $100,090.2 (SD = 57,510.5) with values from about $11.58 up to $199,992.48, and the account balance averages $76,485.9 (SD = 62,397.4) with a range from 0.0 to $250,898.09. These descriptive statistics offer a comprehensive snapshot of the customer profiles and financial behaviors analyzed in this study.

Table 3: Descriptive Statistics of Variables

Variable	Statistics / Frequency	Variable	Statistics / Frequency
Geography	France (5014 | 50.1 %) Germany (2509 | 25.1 %) Spain (2477 | 24.8 %)	NumOfProducts	Mean (SD): 1.5 (0.6) Min ≤ Med ≤ Max: 1 ≤ 1.0 ≤ 4
Gender	Male (5457 | 54.6 %) Female (4543 | 45.4 %)	HasCrCard	Mean (SD): 0.7 (0.5) Min ≤ Med ≤ Max: 0 ≤ 1.0 ≤ 1
CreditScore	Mean (SD): 650.5 (96.7) Min ≤ Med ≤ Max: 350 ≤ 652.0 ≤ 850	IsActiveMember	Mean (SD): 0.5 (0.5) Min ≤ Med ≤ Max: 0 ≤ 1.0 ≤ 1
Tenure	Mean (SD): 5.0 (2.9) Min ≤ Med ≤ Max: 0 ≤ 5.0 ≤ 10"	EstimatedSalary	Mean (SD): 100090.2 (57510.5) Min ≤ Med ≤ Max: 11.58 ≤ 100193.915 ≤ 199992.48

Balance	Mean (SD): 76485.9 (62397.4) Min ≤ Med ≤ Max: 0.0 ≤ 97198.54000000001 ≤ 250898.09"	Exited	Not exited: 79.6 % Exited: 20.4 %

4.3 Correlation Analysis

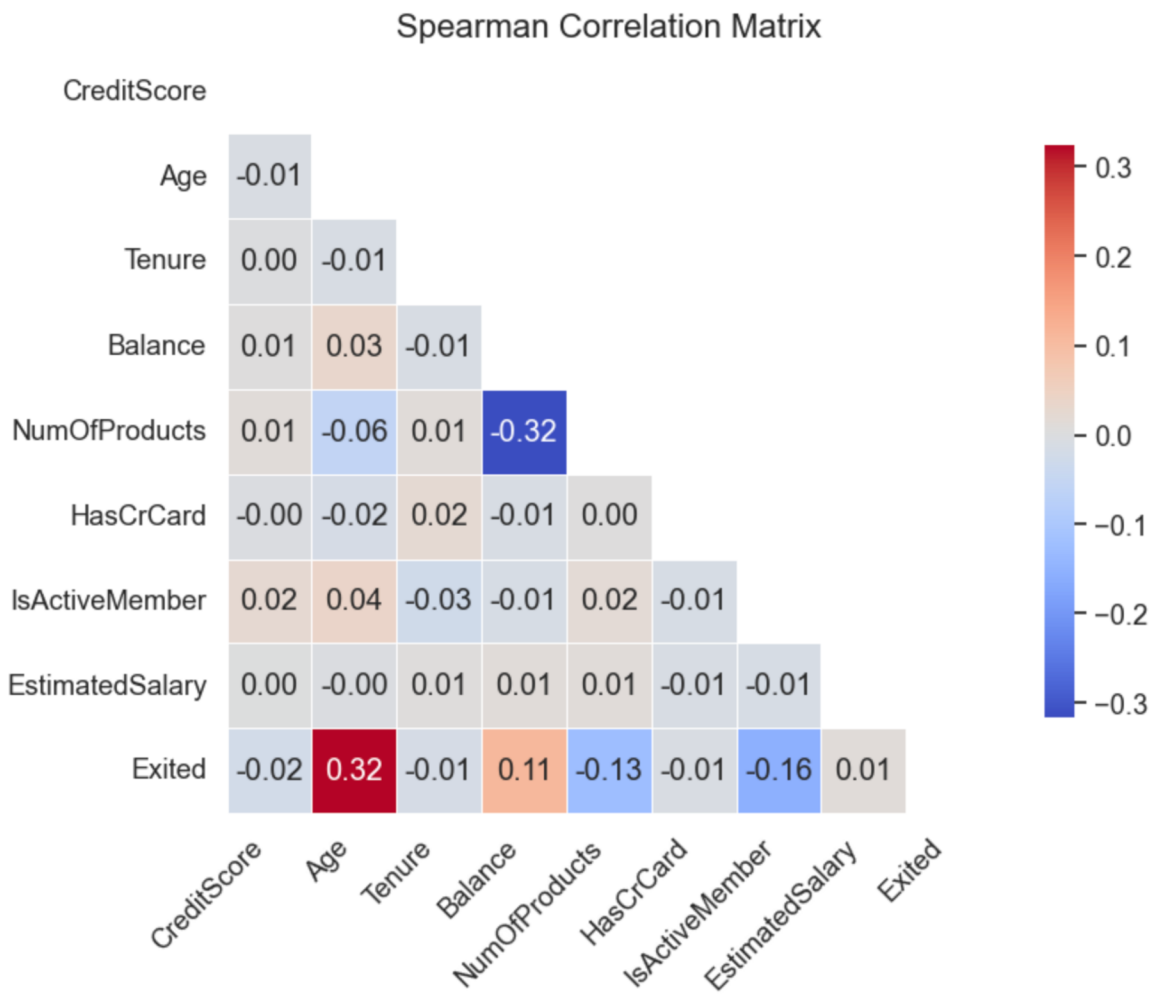

Figure 2: Spearmans's Correlation Matrix

Spearman's correlation is particularly suitable for identifying monotonic relationships in datasets that may include outliers, making it well-suited for customer behavior analysis. The Spearman correlation matrix shown in Figure 2 suggests mostly weak correlations between features and customer churn (Exited).

Among all variables, the highest correlation with **Exited** is with **Age** (0.32), indicating a moderate positive monotonic relationship—older customers may be more likely to churn.

Another moderately positive correlation is observed with **Balance** (0.11), suggesting that higher balances might be linked to higher churn probability.

On the contrary, **IsActiveMember** (-0.16) and **NumOfProducts** (-0.13) both show moderate negative correlations with churn. This implies that active members and those with more products are less likely to leave. **EstimatedSalary**, **Tenure**, and **CreditScore** have very low correlations with churn (near 0), indicating limited monotonic relationships.

Notably, **HasCrCard** (0.00) shows no significant monotonic relationship with churn, suggesting credit card ownership doesn't meaningfully affect churn risk.

Due to the relatively low correlation coefficients across most features, it is advisable to use **nonlinear machine learning models** (e.g., Random Forest, XGBoost, or Neural Networks) to better capture complex interactions and nonlinear dependencies among variables affecting customer churn.

4.4 Dataset Partitioning Strategy

To ensure a robust and generalizable evaluation of the machine learning model, the dataset was split into training and testing subsets using the train_test_split function from the scikit-learn library in python as follows:

```
# Train - Test Split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=42,
                                                    shuffle=True,
                                                    stratify=y)
```

Figure 3: Data Partition Strategy

- **X, y**: Respectively represent the features and target variable.
- **test_size=0.3**: Allocates 30% of the dataset to the test set, leaving 70% for training. This is a standard ratio ensuring a sufficient sample size for both training and evaluation.
- **random_state=42**: Fixes the random seed for reproducibility, ensuring consistent splits across different runs.

- **shuffle=True**: Randomizes the dataset before splitting, which prevents any order-based biases.
- **stratify=y**: Ensures the class distribution is preserved across both training and testing datasets. This is particularly important for imbalanced classification tasks, where class representation must be consistent.

4.4 Model Initialization: XGBoost Classifier

To build a high-performance classification model capable of handling imbalanced data, an **XGBoost (Extreme Gradient Boosting)** classifier was used. The configuration is as follows:

```
# 2. Initialize XGBoost classifier
clf_xgb = xgb.XGBClassifier(
    use_label_encoder=False,
    eval_metric='aucpr',  # better for imbalanced data
    scale_pos_weight=scale_pos_weight,
    random_state=42,
    n_estimators=1000,  # large number, rely on early stopping
    learning_rate=0.05,
    max_depth=6
)
```

Figure 4: Model Parameters

- **use_label_encoder=False**: Disables XGBoost's default label encoder, allowing for more control over data preprocessing.
- **eval_metric='aucpr'**: Uses **Area Under the Precision-Recall Curve** as the evaluation metric, which is more informative for imbalanced dataset.
- **scale_pos_weight=scale_pos_weight**: Assigns a weight to the positive class to compensate for imbalanced class distributions, improving model sensitivity.
- **random_state=42**: Ensures model reproducibility.
- **n_estimators=1000**: Sets a high number of boosting rounds, relying on early stopping to avoid overfitting.

- **learning_rate=0.05**: A smaller learning rate allows the model to converge slowly and improve generalization.
- **max_depth=6**: Limits the maximum tree depth, controlling model complexity and reducing overfitting risk.

4.5 Traininig Dataset Evaluation

Figure 5: Training Dataset Evaluation

Figure 5 summarizes the performance of four classification models on the **training dataset**, evaluated using four key metrics: **Sensitivity**, **Specificity**, **Area Under the Receiver Operating Characteristic Curve (AUC)**, and **Accuracy**. These metrics were selected to capture both class-specific performance (i.e., sensitivity and specificity) and overall model effectiveness (i.e., AUC and accuracy).

It can be observed that the **Decision Tree** and **Random Forest** classifiers achieved perfect scores across all four metrics, with values of 1.00 for sensitivity, specificity, AUC, and accuracy. This outcome suggests that both models were able to fit the training data without error. However, such perfect performance on the training set typically indicates that the models may have **overfitted** the training data. Overfitting occurs when a model captures not only the underlying patterns but also the noise in the dataset, thereby limiting its generalizability to unseen data.

The **Logistic Regression** model demonstrated a notably lower sensitivity (0.17), while achieving high specificity (0.97), an AUC of 0.74, and an accuracy of 0.81. These results suggest that the model was more effective in correctly identifying the negative class but performed poorly in detecting the positive class. This behavior is commonly observed in **imbalanced classification problems**, where the minority class tends to be underrepresented. The lower sensitivity indicates that a substantial proportion of positive instances were misclassified, which could be detrimental in scenarios where the detection of the positive class is critical.

In contrast, the **Gradient Boosting** model produced a more balanced performance. Although its sensitivity (0.46) was lower than ideal, it was significantly higher than that of Logistic Regression. Furthermore, it achieved a specificity of 0.97, an AUC of 0.88, and an accuracy of 0.87. These values indicate that Gradient Boosting was able to learn more complex relationships within the data, capturing meaningful patterns without overfitting to the extent seen in the Decision Tree or Random Forest models.

In summary, while the tree-based models (Decision Tree and Random Forest) exhibited excellent performance on the training set, the risk of overfitting must be considered. Gradient Boosting emerged as a more balanced alternative, offering strong predictive power without perfectly memorizing the training data. These findings highlight the importance of **evaluating model performance on a separate test set**, which is addressed in the following section.

4.6 Test Dataset Evaluation

Figure 6: Test Data Evaluation Results

Figure 6 shows the performance of the same four classification models evaluated previously, this time using the **test dataset**. As with the training set, the models were assessed based on **Sensitivity**, **Specificity**, **Area Under the Curve (AUC)**, and **Accuracy**. These metrics provide insight into each model's ability to generalize to previously unseen data.

Upon inspection of the heatmap, several key observations can be made:

The **Logistic Regression** model achieved the highest **specificity** (0.98), indicating a strong capability to correctly identify negative instances. However, its **sensitivity** remained low at 0.18, similar to its performance on the training set. This indicates that the model consistently struggled to detect positive cases, suggesting that it may not be suitable for tasks where the identification of the positive class is critical. Nonetheless, its overall **accuracy** was 0.82 and **AUC** was 0.78, reflecting a relatively balanced performance in terms of discrimination ability.

The **Decision Tree** classifier, which previously achieved perfect scores on the training set, experienced a substantial decline in test performance. Its **sensitivity** decreased to 0.49 and **AUC** to 0.68, indicating a weaker ability to distinguish between classes. The drop in accuracy (0.78) and specificity (0.86) further confirms that the model had **overfit** the training data, resulting in reduced generalization capability on unseen instances.

The **Random Forest** model maintained high performance across all metrics, achieving 0.48 in sensitivity, 0.96 in specificity, 0.86 in AUC, and 0.86 in accuracy. Although its sensitivity was not markedly high, its performance remained **well-balanced** and did not suffer from the overfitting that was observed in the Decision Tree model. This suggests that Random Forest provides a more stable and generalizable solution in this context.

The **Gradient Boosting** model delivered the most favorable results among all models on the test set. It achieved a **sensitivity** of 0.48, **specificity** of 0.97, **AUC** of 0.88, and **accuracy** of 0.87. These values reflect a **high degree of generalizability** and effectiveness across all key performance measures. Compared to its training performance, Gradient Boosting showed minimal degradation, indicating that it successfully learned from the training data without overfitting.

In summary, while Logistic Regression offered high specificity and simplicity, it underperformed in terms of sensitivity. Decision Tree overfitted the training data and failed to generalize effectively. Random Forest exhibited strong and stable performance, whereas Gradient Boosting demonstrated the **best overall generalization** across all metrics, making it a strong candidate for the further analysis.

4.6 Logistic Regression Estimated Parameters

Feature	Coefficient
CreditScore	-0.0020
Geography	0.1841
Gender	-0.3887
Age	0.0677
Tenure	-0.0131
Balance	0.0000
NumOfProducts	-0.0402
HasCrCard	-0.1294
IsActiveMember	-1.1566
EstimatedSalary	0.0000
Intercept	-2.4546

Figure 7: Logistic Regression Estimated Parameters

The logistic regression model predicts a binary outcome (e.g., customer churn) using coefficients that reflect changes in log odds per unit increase in each feature:

- **IsActiveMember (-1.1566)**: Active members are far less likely to churn, emphasizing engagement's role.
- **Gender (-0.3887)**: Males are less prone to churn than females.
- **Geography (0.1841)**: Certain regions show higher churn risk.
- **Age (0.0677)**: Older age slightly increases churn likelihood.
- **CreditScore, Balance, EstimatedSalary (~0)**: These have negligible impact, suggesting irrelevance or multicollinearity.
- **Intercept (-2.4546)**: Baseline log odds when all features are zero.

4.7 Feature Importances

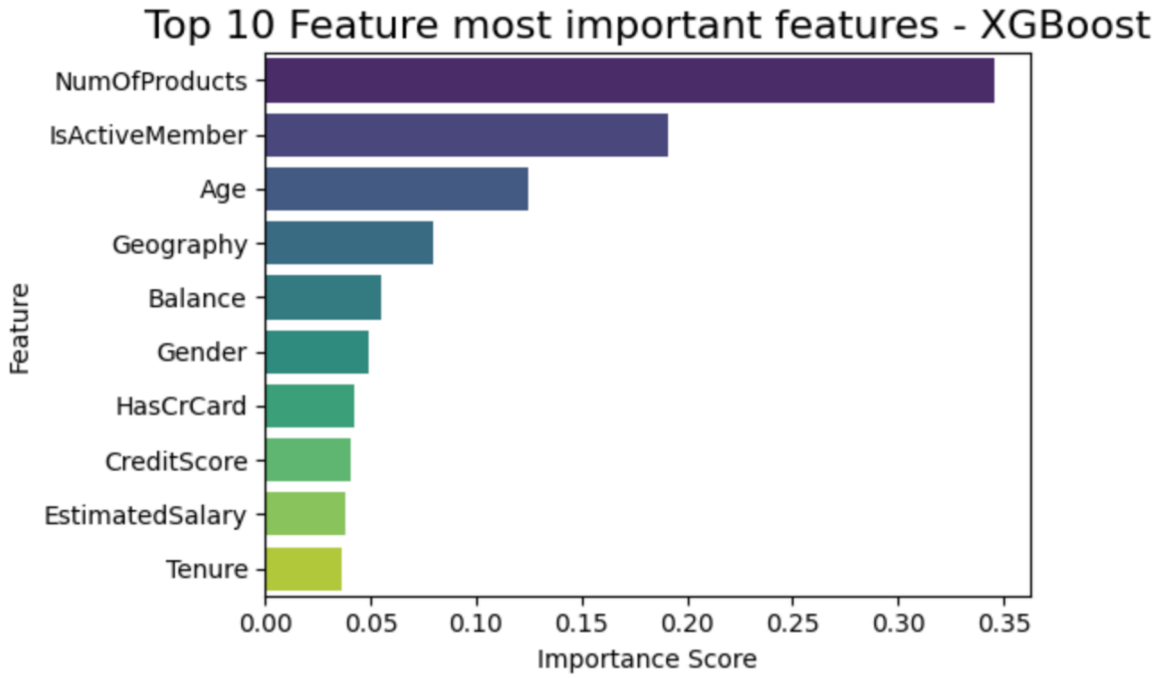

Figure 8: Feature Importances

The Figure 7 shows the top 10 features ranked by their importance in the model. **NumOfProducts**, **IsActiveMember**, and **Age** are the most influential, indicating they play a key role in the model's predictions. Other features like **Geography**, **Balance**, and **Gender** also contribute but to a lesser extent.

These importances are based on the model's internal metrics and serve as a baseline understanding of feature relevance. In the following analysis, these will be further evaluated using **explainability methods** (e.g., SHAP) to gain deeper insight into how each feature affects individual predictions.

4.8 Explainability Implementation

4.8.1 Global Explainability with Shap

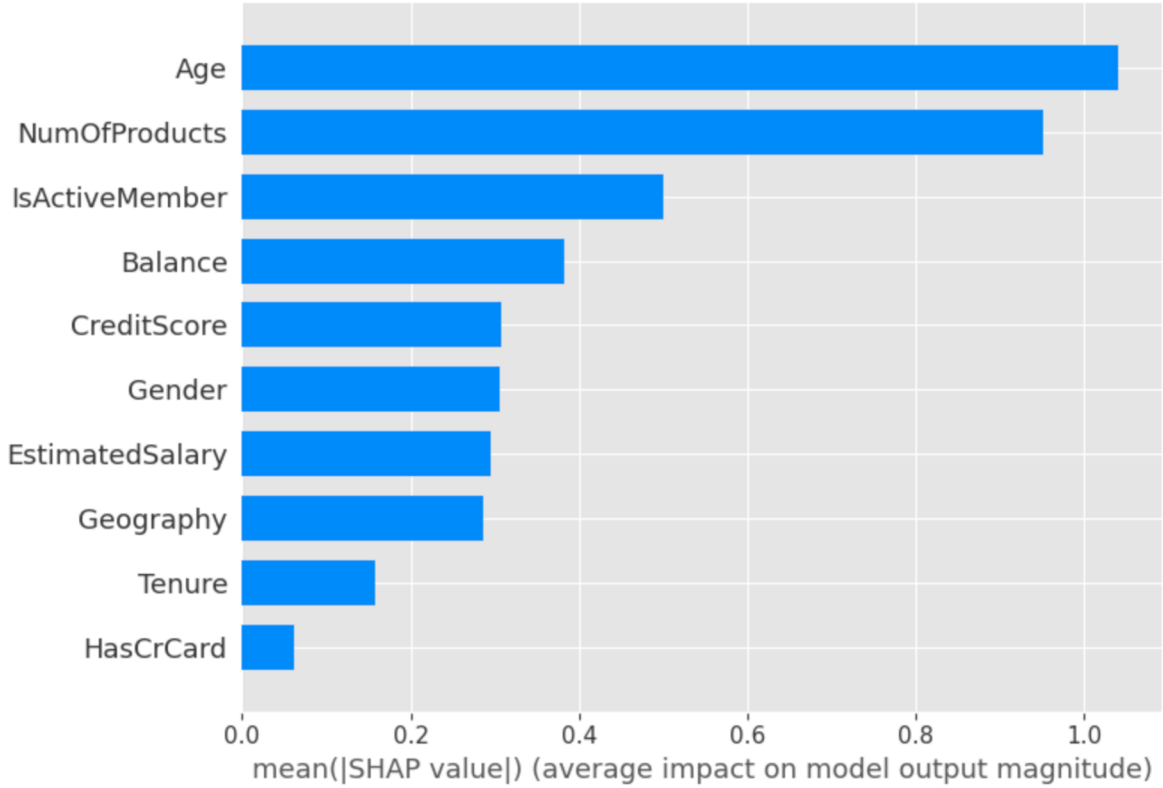

Figure 9: Global Explainability with Shap on XGBoost model

The SHAP summary plot on Figure 8 generated using Shap's **TreeExplainer** method that provides a **global interpretability** on the customer churn prediction model. This analysis highlights the overall impact of each feature across all predictions.

The **mean absolute SHAP values** represent the average contribution of each feature to the model's decision-making, offering insights into the **general trends** influencing customer churn.

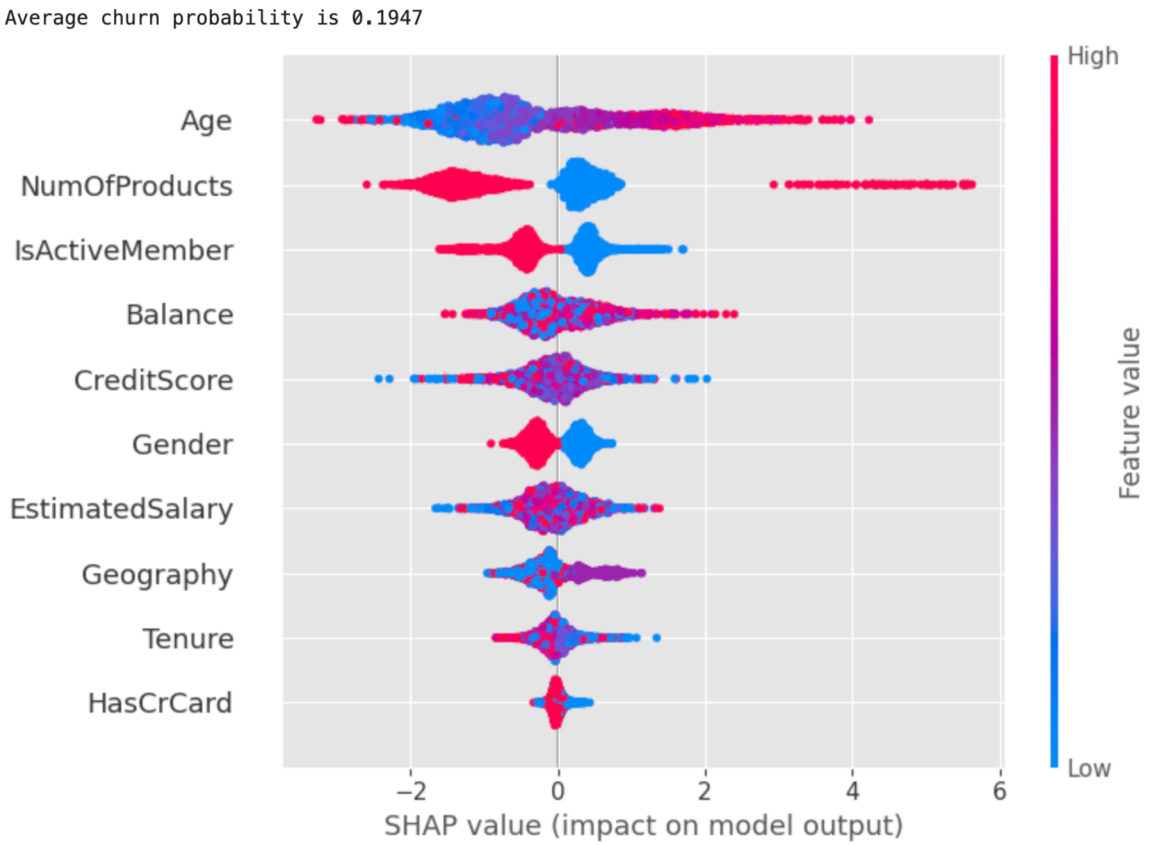

Figure 10: Shap Summary plot for global explainability with XGBoost

The SHAP summary plot presented in the Figure 9 provides a **global explainability analysis** of the customer churn prediction model using **SHAP's TreeExplainer**. It shows different feature values influence the model's output. The **x-axis represents SHAP values**, which indicate the impact of each feature on the model's churn prediction, while the **color gradient represents feature values** (red for high values, blue for low values).

The **average churn probability (0.1947)** represents the **mean predicted likelihood** of customer churn across the test dataset. It serves as a **baseline probability**, indicating that, on average, **19.47% of customers are predicted to churn**.

- Individual predictions fluctuate around this value based on feature contributions.
- Higher SHAP values push predictions above this baseline, increasing churn risk.
- Lower SHAP values reduce churn likelihood, improving retention prospects.

- This metric helps in **understanding overall churn trends** and comparing individual customer risks.

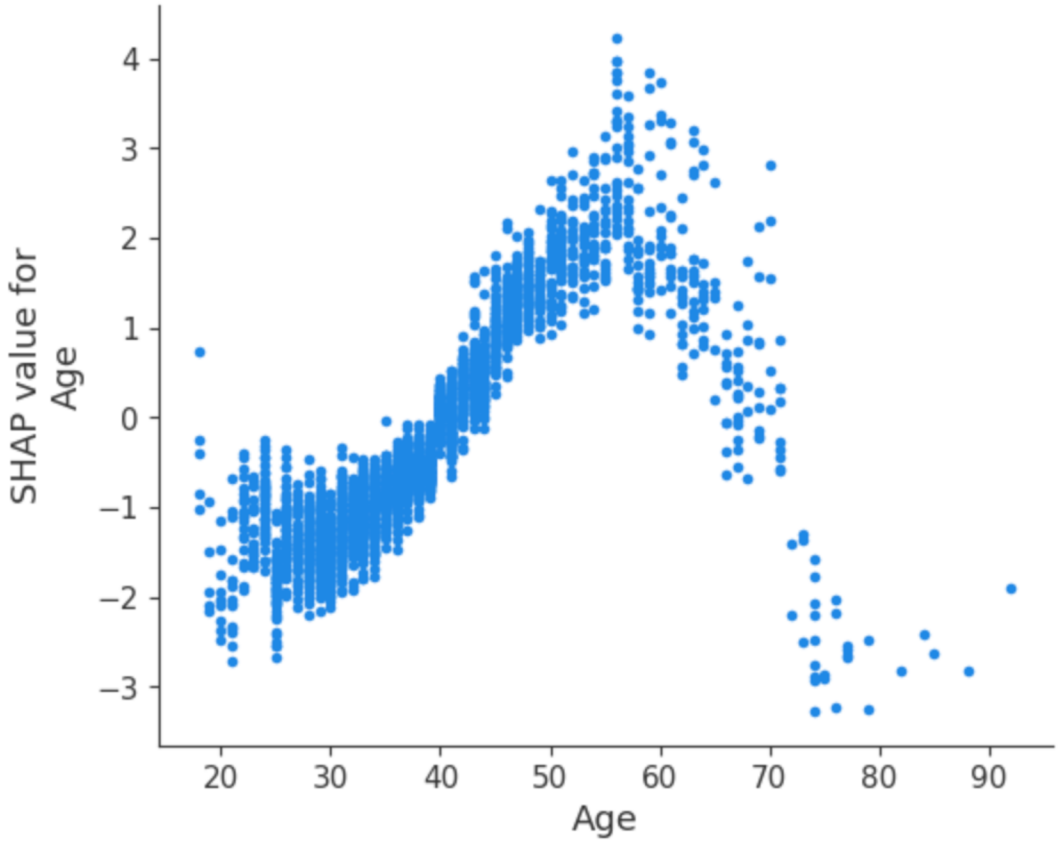

Figure 11: Shap Dependence Plot for Age Variable

A **SHAP dependence plot** from figure 10 shows how a feature's actual value (x-axis) impacts its contribution (y-axis, the SHAP value) to the model's prediction. In our churn model:

- **X-axis (Age):** The customer's age.
- **Y-axis (SHAP value):** How much Age pushes the prediction toward churn (positive) or staying (negative).

For Age:

- **Younger (<30)** and **older (>65)** customers have **negative SHAP values** (lower churn risk).
- **Middle-aged (35–60)** customers have **positive SHAP values** (higher churn risk).

Figure 12: Shap Value Distribution Between Features

Dependence plot from figure 11 shows how shap values for age distributed among the 'IsActiveMember'. Inactive customers (represented in blue) tend to have higher SHAP values across the same age range, indicating that inactivity further amplifies the risk of churn.

Conversely, active customers (represented in red) generally exhibit lower SHAP values, suggesting a mitigating effect on churn risk. Thus, both age and customer activity level are shown to significantly influence the model's churn predictions.

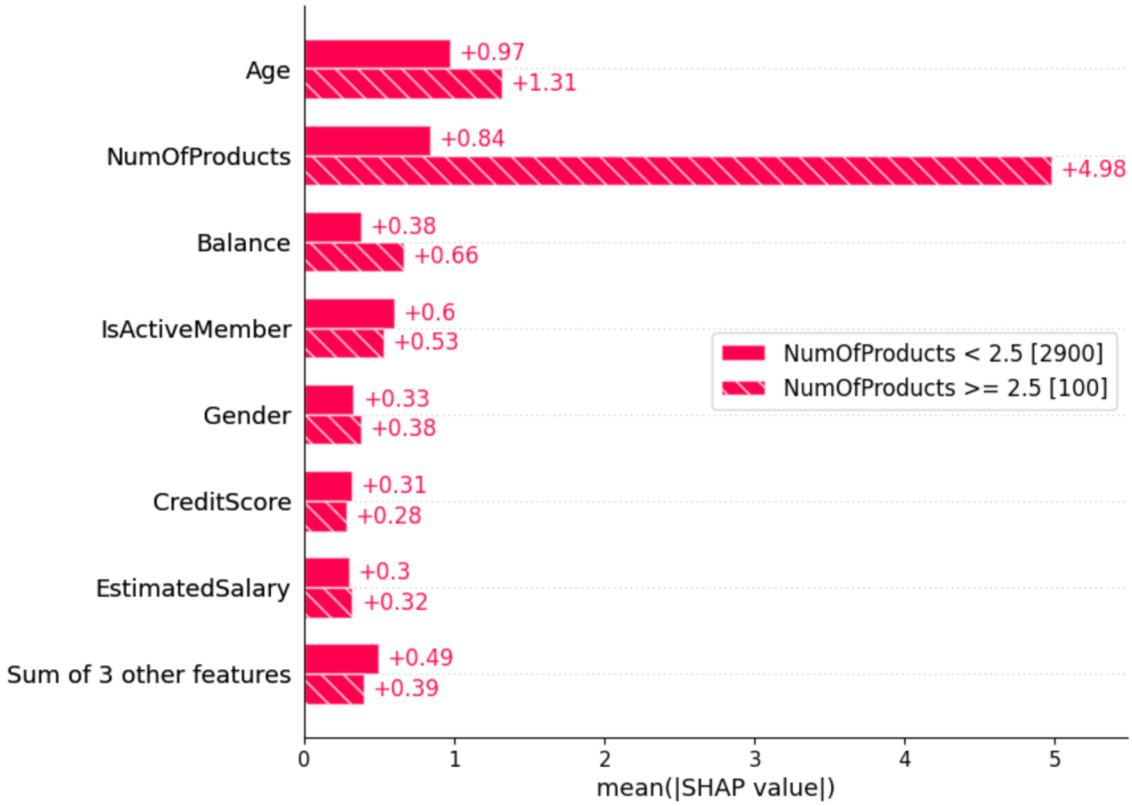

Figure 13: Shap Cohort Analysis

In the SHAP cohort analysis in Figure 12, the dataset was divided based on the feature **NumOfProducts**, creating two groups: customers with fewer than 2.5 products (n = 2,900) and those with 2.5 or more (n = 100). The goal was to compare how different features contribute to churn predictions across these cohorts.

For customers with **fewer products**, churn predictions were influenced by a combination of features, with **Age** being the most important, followed by **NumOfProducts**, **IsActiveMember**, and **Balance**. This indicates a multifactorial explanation for churn in this group.

In contrast, for customers with **three or more products**, **NumOfProducts** overwhelmingly dominated the prediction, with a significantly higher SHAP value than all other features. **Age** also remained relevant, but other features had minimal impact.

These results suggest that churn drivers differ across segments: while multiple factors influence churn among low-product users, churn among high-product users is primarily driven by the number of products held.

4.8.2 Local Explainability with Shap

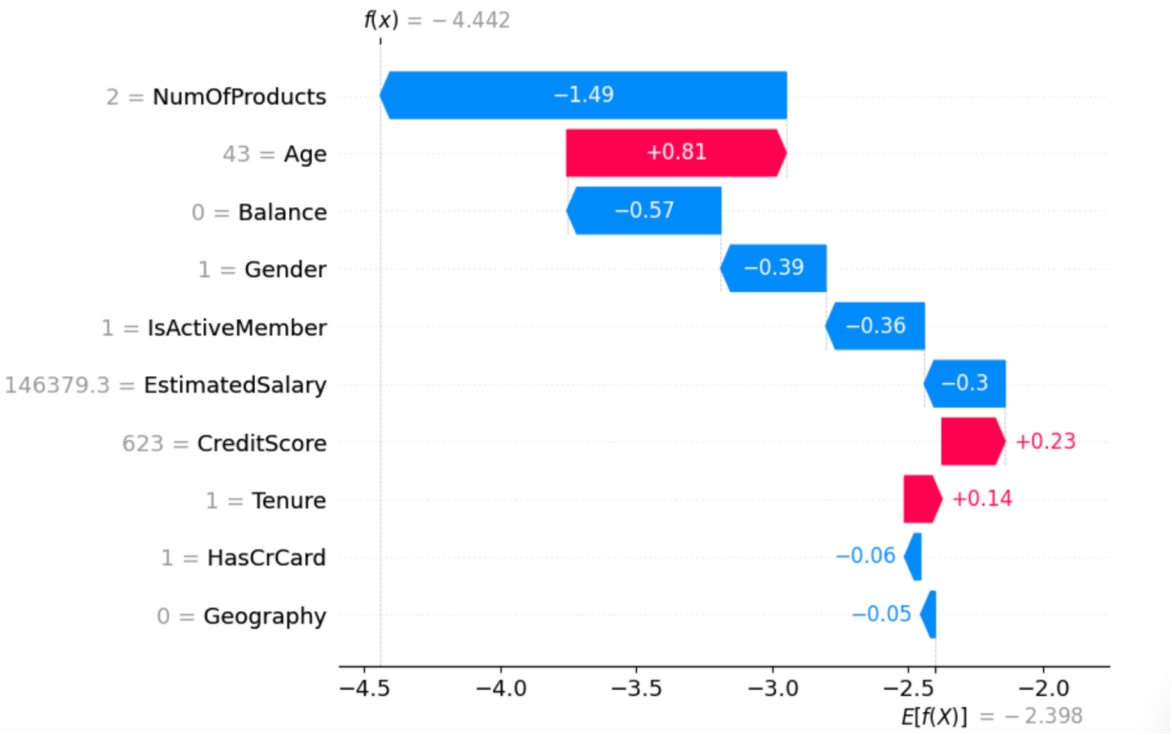

Figure 14: Local Explainability with XGBoost Model

In Figure 13, the SHAP waterfall plot illustrates the contribution of individual features to a specific model prediction. The base value, representing the model's average output across all observations (−2.398), is sequentially adjusted by the SHAP values of each feature to arrive at the final model output (−4.442).

Features such as NumOfProducts = 2, Balance = 0, and IsActiveMember = 1 contributed negatively to the prediction, indicating a downward influence. Conversely, features like Age = 43, CreditScore = 623, and Tenure = 1 exerted a positive influence, albeit insufficient to offset the dominant negative contributions. This breakdown enhances interpretability by quantifying and visualizing the directional impact of each feature on the individual prediction.

4.8.3 Local Explainability with Lime

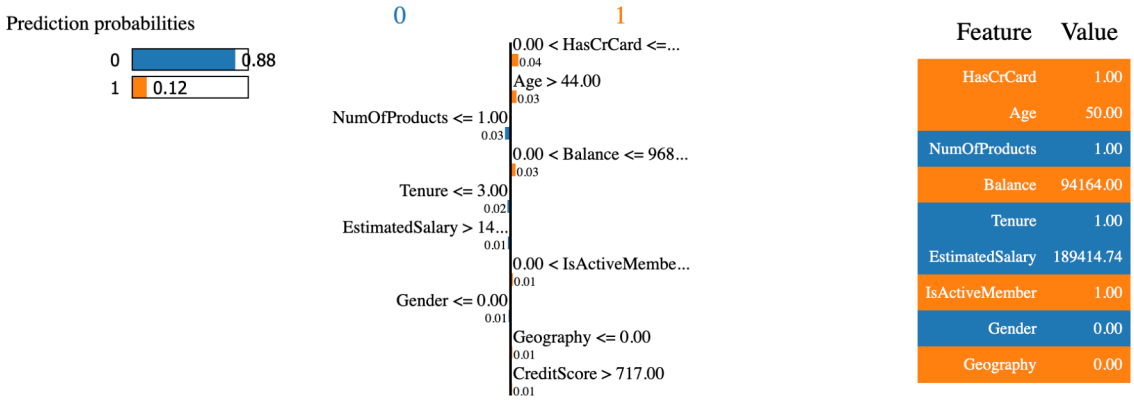

Figure 15: Explainability with Lime

For the selected customer, the model predicted a **low likelihood of churn (12%)**, indicating a **high probability (88%) of retention**. Key features that contributed to the **non-churn prediction** included:

- Having only one product,
- A high estimated salary,
- Short tenure with the bank,
- And demographic features such as gender and geography.

Conversely, features such as older age (50), a high account balance, being an active member, and holding a credit card slightly increased the probability of churn, but their impact was not strong enough to change the overall prediction.This explanation highlights the importance of understanding feature influence at the individual level, enabling more targeted customer retention strategies.

5) Results and Discussion

In this thesis, explainability was examined for the XGBoost model, which achieved the highest predictive performance among all trained models. Due to its ensemble structure and complexity, XGBoost does not provide immediate interpretability. To address this, SHAP (SHapley Additive exPlanations) was applied as the primary method for explaining individual predictions and assessing global and local feature impact.

SHAP, grounded in cooperative game theory, quantifies the contribution of each feature to a specific prediction by considering all possible feature value permutations. Unlike LIME, which relies on locally fitted surrogate models and provides only local interpretability, SHAP offers both global and local explanations with consistency and theoretical soundness. Its integration enables a detailed understanding of how the model arrives at specific decisions and which features most strongly influence predictions.

The SHAP analysis revealed that the most influential features in the XGBoost model were **age**, **number of products**, and **is_active_member**. These features consistently showed the highest average SHAP values across the dataset. Visualizations such as SHAP summary plots and beeswarm plots made it possible to inspect not only the magnitude but also the direction of each feature's impact on the model's output. Besides, these insights were in close alignment with the internal feature importance rankings produced by the XGBoost base model, providing additional credibility to the explanations derived from SHAP.

Moreover, SHAP facilitated the identification of certain limitations within the dataset. Specifically, when the model encountered applicants with feature combinations not represented during training such as atypical values for the number of products or age the resulting SHAP values often indicated elevated uncertainty or an over-reliance on correlated features. These patterns underscore the value of interpretability not only for understanding model behavior, but also for detecting potential biases, data sparsity issues, and limitations in the model's generalizability.

Table 4: Shap vs Lime:

Metrics	SHAP	LIME
Concept	Applies to the model as-is	Fits a local surrogate model to explain the complex model
Theory	Additive feature attribution based on game theory	Feature perturbation method
Type	Post-hoc model-agnostic	Post-hoc model-agnostic
Data type	Images, tabular data and signals	Images, tabular data and signals
Explanation	Global, local	Local
Collinearity consideration	Not in the original method	No
Non-linear decision	Depends on the used model	Incapable
Computing time	Higher	Lower
Visualization	Waterfall, Beeswarm and Summary plots	One single plot

Table 4(Salih et al., 2024) indicates a structured comparison between SHAP and LIME across several key metrics relevant to model interpretability. Both techniques are categorized as **post-hoc model-agnostic** methods, meaning they can be applied after model training and are not restricted to specific model types. However, their conceptual foundations differ significantly. **SHAP** operates based on **additive feature attribution using game theory**, allowing it to apply directly to the model as-is. In contrast, **LIME** constructs a **local surrogate model** to approximate the behavior of the original model and explain individual predictions.

In terms of **data compatibility**, both SHAP and LIME support **images, tabular data, and signals**, offering flexibility in application across domains. SHAP supports **both global and local explanations**, while LIME is limited to **local interpretability**.

With regard to **collinearity**, SHAP's original method does **not explicitly account for it**, and LIME also **does not consider** collinearity among features. When handling **non-linear decision boundaries**, SHAP's performance **depends on the underlying model**, whereas LIME is generally **incapable** of accurately representing such complexities due to its linear surrogate approximation.

From a performance perspective, SHAP has a **higher computational cost** due to its reliance on Shapley value calculations, while LIME is **faster** and more computationally efficient. For **visualization**, SHAP offers a rich suite of tools including **waterfall plots, beeswarm plots, and summary plots**, whereas LIME provides **a single simplified plot** for interpretation.

This comparative analysis highlights that SHAP offers more comprehensive and theoretically grounded explanations but at a higher computational expense, whereas LIME provides a lightweight, interpretable, and efficient alternative suitable for localized analysis.

6) Conclusion

This thesis presented a complete and structured machine learning workflow applied to publicly available data, covering all stages from data preprocessing to model training, evaluation, and interpretation. After cleaning, enriching, and labelling the dataset, several supervised models were trained, including logistic regression classifier, decision trees, random forests, and XGBoost methods. Among these, **XGBoost** achieved the highest performance, within evaluated classification metrics, and was therefore selected for in-depth analysis and interpretability.

To ensure transparency and foster trust in the model's decisions, the explainability of the predictions was critically examined. Both **SHAP** and **LIME** were explored as post-hoc model-agnostic explanation methods. While **LIME** provided localized insights through surrogate models, its interpretability was limited to individual predictions and lacked global explanatory power. **SHAP**, grounded in cooperative game theory, offered a more consistent and theoretically robust alternative. It enabled both local and global analysis and supported rich visualizations such as summary and beeswarm plots, which provided comprehensive insights into how different features influenced model outputs.

The SHAP analysis revealed that **age**, **number of products**, and **is_active_member** were the most influential features in the model's decisions. These findings aligned closely with the internal feature importance scores of XGBoost, reinforcing the validity of the interpretability approach. Furthermore, SHAP helped uncover limitations in the dataset, particularly in cases involving unseen or rare feature combinations, where it revealed elevated uncertainty or over-reliance on correlated attributes. This illustrates the value of explainability not only for understanding model behavior, but also for identifying generalization weaknesses and potential sources of bias.

Although other models such as random forests and decision trees performed competitively, explainability efforts were intentionally centered on XGBoost due to its superior performance and practical relevance for deployment in real-world decision-support scenarios. Analyzing all tree-based models would have introduced methodological redundancy, as their structural similarities would likely yield overlapping interpretive insights.

To sum up, this thesis demonstrates the effectiveness of combining high-performing machine learning models with robust explainability techniques. The results highlight the importance of interpretable AI in sensitive decision-making contexts such as admissions, where accountability, fairness, and transparency are essential. Future work should focus on expanding the dataset, incorporating fairness-aware modeling approaches, and integrating user-facing interpretability interfaces to support ethical, informed, and data-driven decisions.

7) References

Adadi, A., & Berrada, M. (2018). Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, *PP*, 1. https://doi.org/10.1109/ACCESS.2018.2870052

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115. https://doi.org/https://doi.org/10.1016/j.inffus.2019.12.012

Ben Jabeur, S., Stef, N., & Carmona, P. (2023). Bankruptcy Prediction using the XGBoost Algorithm and Variable Importance Feature Engineering. *Computational Economics*, *61*(2), 715–741. https://doi.org/10.1007/s10614-021-10227-1

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *ArXiv Preprint ArXiv:1702.08608*.

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2018). *Explaining Explanations: An Overview of Interpretability of Machine Learning*. http://arxiv.org/abs/1806.00069

Guidotti, R. (2024). Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, *38*(5), 2770–2824. https://doi.org/10.1007/s10618-022-00831-6

Guliyev, H., & Yerdelen, F. (2021). Customer churn analysis in banking sector: Evidence from explainable machine learning models. *Journal of Applied Microeconometrics*, *1*, 85–99. https://doi.org/10.53753/jame.1.2.03

Hassija, V., Chamola, V., Mahapatra, A., Singal, A., Goel, D., Huang, K., Scardapane, S., Spinelli, I., Mahmud, M., & Hussain, A. (2023). Interpreting Black-Box Models: A Review on Explainable Artificial Intelligence. *Cognitive Computation*, *16*. https://doi.org/10.1007/s12559-023-10179-8

Hassija, V., Chamola, V., Mahapatra, A., Singal, A., Goel, D., Huang, K., Scardapane, S., Spinelli, I., Mahmud, M., & Hussain, A. (2024). Interpreting Black-Box Models: A Review on Explainable Artificial Intelligence. *Cognitive Computation*, *16*(1), 45–74. https://doi.org/10.1007/s12559-023-10179-8

Hoens, T., & Chawla, N. (2013). *Imbalanced Datasets: From Sampling to Classifiers* (pp. 43–59). https://doi.org/10.1002/9781118646106.ch3

Juneja, A., Kumar, V., & Singla, S. K. (2024). Partition Explainer Next Generation in Single Image Dehazing. *2024 IEEE International Conference on Contemporary Computing and Communications (InC4)*, *1*, 1–6. https://doi.org/10.1109/InC460750.2024.10649081

Kuncheva, L. (2014). Combining Pattern Classifiers: Methods and Algorithms: Second Edition. In *Combining Pattern Classifiers: Methods and Algorithms: Second Edition* (Vol. 47). https://doi.org/10.1002/0471660264

Lambert, J., & Lipkovich, I. (2008). *A Macro For Getting More Out Of Your ROC Curve*. https://api.semanticscholar.org/CorpusID:15055367

Lipton, Z. C. (2017). *The Mythos of Model Interpretability*. https://arxiv.org/abs/1606.03490

Mai, X. T., La, M. T. K., Tran, H. T., & Tran, T.-A. (2025). A Multilevel Classification Approach for Chart Identification. In N. Thai-Nghe, T.-N. Do, & S. Benferhat (Eds.), *Intelligent Systems and Data Science* (pp. 173–187). Springer Nature Singapore.

Miller, C. A. (2021). Chapter 11 - Trust, transparency, explanation, and planning: Why we need a lifecycle perspective on human-automation interaction. In C. S. Nam & J. B. Lyons (Eds.), *Trust in Human-Robot Interaction* (pp. 233–257). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-819472-0.00011-3

Miller, T. (2018). *Explanation in Artificial Intelligence: Insights from the Social Sciences*. https://arxiv.org/abs/1706.07269

Mosca, E., Szigeti, F., Tragianni, S., Gallagher, D., & Groh, G. (2022). SHAP-Based Explanation Methods: A Review for NLP Interpretability. In N. Calzolari, C.-R. Huang, H. Kim, J. Pustejovsky, L. Wanner, K.-S. Choi, P.-M. Ryu, H.-H. Chen, L. Donatelli, H. Ji, S. Kurohashi, P. Paggio, N. Xue, S. Kim, Y. Hahm, Z. He, T. K. Lee, E. Santus, F. Bond, & S.-H. Na (Eds.), *Proceedings of the 29th International Conference on Computational Linguistics* (pp. 4593–4603). International Committee on Computational Linguistics. https://aclanthology.org/2022.coling-1.406/

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., & Yu, B. (2019). Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, *116*(44), 22071–22080. https://doi.org/10.1073/pnas.1900654116

Roshan, K., & Zafar, A. (2022). Using Kernel SHAP XAI Method to Optimize the Network Anomaly Detection Model. *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, 74–80. https://doi.org/10.23919/INDIACom54597.2022.9763241

Salih, A., Raisi, Z., Boscolo Galazzo, I., Radeva, P., Petersen, S., Lekadir, K., & Menegaz, G. (2024). A Perspective on Explainable Artificial Intelligence Methods: SHAP and LIME. *Advanced Intelligent Systems*, *7*. https://doi.org/10.1002/aisy.202400304

Sharma, P., Mirzan, S. R., Bhandari, A., Pimpley, A., Eswaran, A., Srinivasan, S., & Shao, L. (2020). Evaluating Tree Explanation Methods for Anomaly Reasoning: A Case Study of SHAP TreeExplainer and TreeInterpreter. In G. Grossmann & S. Ram (Eds.), *Advances in Conceptual Modeling* (pp. 35–45). Springer International Publishing.

Suthaharan, S. (2016). Support Vector Machine. In S. Suthaharan (Ed.), *Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning* (pp. 207–235). Springer US. https://doi.org/10.1007/978-1-4899-7641-3_9

Temenos, A., Temenos, N., Kaselimi, M., Doulamis, A., & Doulamis, N. (2023). Interpretable Deep Learning Framework for Land Use and Land Cover Classification in Remote Sensing Using SHAP. *IEEE Geoscience and Remote Sensing Letters*, *20*, 1–5. https://doi.org/10.1109/LGRS.2023.3251652

Viswan, V., Shaffi, N., Mahmud, M., Subramanian, K., & Hajamohideen, F. (2023). Explainable Artificial Intelligence in Alzheimer's Disease Classification: A Systematic Review. *Cognitive Computation*, *16*. https://doi.org/10.1007/s12559-023-10192-x

Volovoi, V. (2012). Review Article System Reliability at the Crossroads. *ISRN Applied Mathematics*, *2012*, ID850686. https://doi.org/10.5402/2012/850686

Winter, E. (2002). Chapter 53 The shapley value. In *Handbook of Game Theory with Economic Applications* (Vol. 3, pp. 2025–2054). Elsevier. https://doi.org/https://doi.org/10.1016/S1574-0005(02)03016-3

Woźniak, M., Graña, M., & Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion*, *16*, 3–17. https://doi.org/https://doi.org/10.1016/j.inffus.2013.04.006

8) List of pictures, tables, graphs and abbreviations

8.1 List of pictures

8.2 List of tables