uc3m | Universidad Carlos III de Madrid

Master's Degree in Computational Social Science
Academic Year 2024 - 2025

Master's Thesis

"Fraud detection performance and interpretability: controlled simulation of statistical and Machine Learning models"

Pablo Romero Medinilla

Supervisor

Ignacio Garrón Vedia

Madrid, September 2025

Abstract

Fraud in the insurance sector is a problem of economic and social significance, as it generates losses, increases premiums for legitimate policyholders, and erodes confidence in the institution. In this context, predictive analysis techniques have taken center stage as the key to early detection of such fraud. Traditionally, statistical models such as logistic regression have been used for their simplicity and interpretability, but their limitations have led to a migration to machine learning models. These have a greater capacity to model complex patterns and improve case detection, albeit at the cost of less transparency.

This paper addresses the issue by developing a controlled experiment through the generation of two synthetic databases in which to compare, under equal conditions, traditional statistical models and ML algorithms in terms of performance and interpretability. In addition, post-hoc tools such as SHAP and DALEX are added. The results show that logistic regression maintains competitive performance under global ranking criteria, while ML models achieve better levels of fraud detection and post-hoc techniques improve their explainability. It is concluded that the optimal choice depends on a necessary prior balance between predictive accuracy and interpretability.

Key words: Insurance fraud, logistic regression, machine learning, interpretability, performance, and SHAP.

Resumen

El fraude en el contexto el sector asegurador supone un problema de relevancia económica y social, al generar pérdidas, aumentar las primas a los asegurados legítimos y erosionar la confianza en la institución. En este contexto, las técnicas de análisis predictivo han tomado protagonismo como clave para la detección temprana de dichos fraudes. Tradicionalmente se han empleado modelos estadísticos como la regresión logística por su simplicidad e interpretabilidad, pero sus limitaciones han provocado una migración a modelos de Machine Learning. Estos tienen mayor capacidad de modelar patrones complejos y mejoran la detección de casos, aunque a costa de menor transparencia.

El presente trabajo incide en la problemática al desarrollar un experimento controlado mediante la generación de dos bases de datos sintéticas en las que comparar, en igualdad de condiciones, modelos estadísticos tradicionales y algoritmos de ML en términos de rendimiento e interpretabilidad. Además, se añaden herramientas post-hoc como SHAP y DALEX. Los resultados muestran que la regresión logística mantiene un rendimiento competitivo bajo criterios de ordenamiento global, mientras que los modelos de ML alcanzan mejores niveles de detección de fraude y las técnicas post-hoc mejoran su explicabilidad. Se concluye que la elección óptima depende de un equilibrio previo necesario entre precisión predictiva e interpretabilidad.

Palabras clave: fraude en seguros, regresión logística, Machine Learning, interpretabilidad, rendimiento y SHAP.

TABLE OF CONTENTS

1. INTRODUCTION

When we refer to fraud in the insurance world, we are talking about a problem that, although limited to the business sphere, has a direct impact on the lives of citizens in a country such as Spain. The obligation to have different types of insurance, as found in Article 2 of Law 21/2007, makes our country a large niche market for this type of company.

However, this obligation also has an undesirable effect: fraud. The most recent estimates indicate that, in our country alone, in 2024, 1.97% of insurance claims were fraudulent (Garrote, 2025). This type of fraud, in addition to the direct financial impact it can have on the company, has the undesirable effect of undermining confidence in control mechanisms and increasing premiums for those policyholders who make responsible use of their policies.

For decades, this concern and the increase in data has led to the creation of large departments within insurance companies to try to detect and prevent fraud. Traditionally, these departments used traditional statistical models to make their estimates. These models offer simplicity, transparency, and ease of interpretation, but they can be insufficient when the relationships between variables are complex or the number of characteristics is high (Ding et al., 2024). In addition, the changing context in which we find ourselves must be considered.

To address these limitations, insurance companies and academia suggest that tools such as Machine Learning (ML) and Artificial Intelligence (AI) will be key. The study by Ding et al. (2024) shows us an accuracy of 95% in detecting fraud in auto insurance, far exceeding the results of traditional models and demonstrating its usefulness in the context of insurance fraud, where patterns can be subtle, multidimensional, and infrequent (Ding et al., 2024).

However, the aforementioned improvements introduced by ML in our case come with new challenges and issues to consider. The real benefits and implications in terms of interpretability or overfitting of these models require rigorous analysis by the parties involved in implementing them. For all these reasons, this paper seeks to answer the following research question, which arises from this initial study of the situation to be addressed: What are the differences between traditional statistical models and machine learning techniques in terms of performance and interpretability in the field of insurance fraud detection? To answer this question, we have formulated the following main research objective: Compare the predictive performance of machine learning techniques with traditional statistical models and propose and evaluate mechanisms to improve interpretability.

Once we have outlined the main aspects of this research, we proceed to carry out a theoretical-methodological study that reveals the current state of the art of the subject under study and, in turn, provides us with key points for the subsequent methodological proposal. All of this will be done based on the following structure: Section 2 offers the conceptual framework related to the object of study and the methodology, while Section 3 presents the configurations to be used in the data simulation. Section 4 offers a brief explanation of the methodology applied, while Section 5 uses the simulated data to test the performance and interpretability of traditional statistical models compared to those of Machine Learning. Finally, Section 6 revisits the specific objectives of the research to review their status once we have obtained the necessary information and concludes the present research.

2. CONCEPTUAL FRAMEWORK

2.1. The problem of fraud in the insurance sector

When we talk about insurance fraud, we refer to any intentional action aimed at obtaining an undue benefit by distorting or falsifying information presented in a claim to the insurance company (Priya & Pushpa, 2017: 630). If we take a first look at the literature, we see that this can take various forms, from completely faking an accident to exaggerating the actual damage (idem). As we discussed in the introductory section, this type of practice represents a serious multidimensional problem that results in millions of euros in losses, erosion of customer confidence, and a general increase in premiums.

Thus, the integration of analytical tools capable of identifying suspicious patterns in large volumes of data that are impractical for human inspection has gained relevance in recent years, as demonstrated by the study by Patil and Godbole, who detect more than 13 possible combinations of algorithms to study fraud prediction using ML techniques (Patil & Godbole, 2018: 4362). The aim of these algorithms is to significantly increase detection efficiency, improve accuracy, and reduce operating costs (Adedayo et al, 2023: 760).

These aspects, together with the growing complexity of fraud and the availability of large volumes of data, have encouraged this search for more effective predictive solutions. However, it should not be forgotten that classic statistical models offered a solid basis for fraud prediction, such as logistic regression.

Furthermore, recent studies highlight how fraud not only affects the economic sphere of insurance companies, but also represents a systemic risk that creates a less equitable environment for legitimate policyholders and more intense regulatory pressure for companies (Polat & Reva, 2018). This increase in fraud, with the consequent diversion of resources to these verification and control tasks, reduces their overall efficiency.

In this context, data analysis has evolved from an auxiliary function to a critical tool in the design of anti-fraud strategies, rendering traditional approaches obsolete. In fact, the very structure of modern fraud means that its detection requires complex algorithmic solutions, often inspired by machine learning approaches, but which must be adapted to the regulatory requirements of the insurance sector (Plaisant van der Wal, 2018).

It should be noted that this issue has not gone unnoticed in institutional and academic circles. This review of the state of the art shows how academic institutions and insurance companies have encouraged the development of analytical models that integrate historical data, unstructured variables, and even social media analysis to identify patterns of fraudulent behavior (Óskarsdóttir et al., 2020). This trend, together with the extensive literature on the subject, demonstrates the importance of providing the insurance sector with robust scientific tools, not only to detect fraud, but also to actively prevent it.

2.2. Statistical and machine learning models to analyze

This section provides an overview of the models considered in our analysis. For a detailed description of the algorithms and statistical computations, we refer to James et al. (2013).

2.2.1. Traditional statistical models

Once we have understood the general problem we are facing throughout this academic work, we must delve deeper into the different techniques that will later be defined within a methodological proposal that attempts to respond to the objectives, research question, and hypothesis.

If we begin this analysis from the perspective of the traditional statistical models used to respond to and anticipate fraud in the insurance world, we must first discuss the logarithmic model.

This binary classification model estimates the probability that an observation belongs to a given class. Its mathematical structure is based on a logistic function that transforms a linear combination of independent variables into a probability between 0 and 1 (Latiesa, 1991). Among its advantages are its ability to provide valid estimates regardless of study design (Harrell, 2001), its robustness in small samples, and the possibility of estimating the relative weight of each explanatory variable from odds ratios (Latiesa, 1991), which allows for direct interpretability.

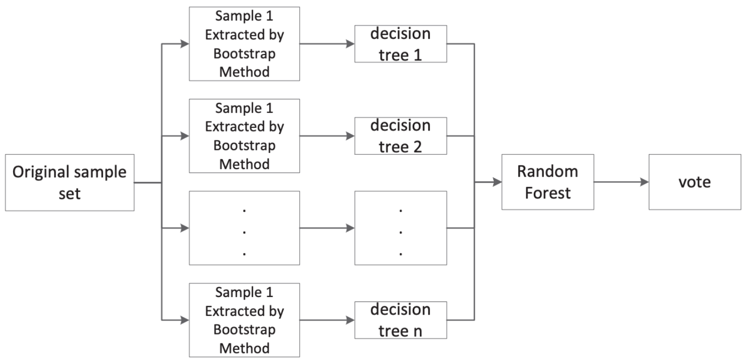
However, these models are based on the assumptions of predictor independence, a linear relationship between variables, and the logarithm of probabilities or the absence of multicollinearity. In contexts with high dimensionality or nonlinear relationships, such as the one presented in this case, these assumptions may be violated, resulting in a reduction in the predictive power of the model, as shown by Saddi et al. (2023), who found lower performance of logistic regression compared to methods such as Random Forest or XGBoost (Saddi et al., 2023).

2.2.2. Applied Machine Learning techniques

This finding regarding the application of machine learning techniques in the insurance sector has driven the development and adoption of these techniques to overcome the structural limitations. Taking a first theoretical approach to these techniques, we observe certain ones that are particularly relevant.

First, Random Forest is an ensemble algorithm based on the Bagging technique. It works by applying the Bootstrap method to the CART (Classification and Regression Trees) algorithm, generating multiple subsets of data through sampling with replacement. From each subset, an independent decision tree is constructed using the CART algorithm, without pruning the generated trees. Once constructed, they are combined to form the Random Forest, as shown in Figure I. For classification tasks, the result is obtained by majority vote among the predictions of each tree, while in regression, the average is used (Guo et al. 2019: 310).

FIGURE I. Generation of Random Forests


Original sample set
Sample 1 Extracted by Bootstrap Method
Sample 1 Extracted by Bootstrap Method
Sample 1 Extracted by Bootstrap Method
decision tree 1
decision tree 2
decision tree n
Random Forest
vote

Source: Guo et al. 2019: 310.

Secondly, Boosting methods, such as XGBoost, work sequentially, iteratively correcting the errors of the previous model, allowing for more aggressive optimization and greater generalization capacity (Sharma et al, 2023: 1 - 6).

A third technique of interest is the use of Support Vector Machines (SVM). This algorithm seeks to find the optimal hyperplane that separates classes with the largest possible margin and can be adapted to nonlinear problems using kernel functions. This method has been used in this context due to its generalization capacity and efficiency with high-dimensional data sets (Priya & Pushpa, 2017: 634 - 635).

All of these techniques, in addition to superior overall predictive performance, stand out for their ability to model complex relationships without the need to manually specify interactions or linearity assumptions. However, these advances bring with them new challenges, which can be summarized in three main areas. On the one hand, we observe a risk of overfitting, especially in complex SVM-type algorithms, when appropriate validation and adjustment strategies are not applied. On the other hand, we observe that the effectiveness of these models depends on the adjustment of hyperparameters, which implies a skilled and sensitive technical process. Thirdly and finally, we cannot forget the lower interpretability, like a "black box," that these models entail. This makes it difficult to explain their decisions and has led to the use of ad-hoc explainability techniques that allow predictions to be broken down into individual contributions by variables (Badhoutiya et al., 2023).

2.3. Performance measures

Once the different algorithms to be used have been defined, it becomes essential to establish a series of criteria that allow the effectiveness of these models to be evaluated objectively and comparably. Performance metrics are the tool that will allow us to understand this aspect, as they offer quantifiable information about the predictive capacity of the models. Among the most used metrics are accuracy, recall, and F1-Score. While accuracy measures the proportion of overall correct predictions, recall focuses on the model's ability to detect true positive cases (Adedotun et al., 2023) (Thanuj Kumar et al., 2021).

Other relevant metrics are the Area Under the ROC Curve (AUC-ROC), which measures the model's ability to discriminate between classes, and the confusion matrix, which allows for a detailed visualization of true positives, false positives, false negatives, and true negatives (Chakravarty & Singh, 2022: 209-210). These metrics allow the impact of class balancing strategies, decision threshold adjustments, or variable selection to be evaluated.

2.4. Measures of interpretability in predictive models

On the other hand, we must understand that, when applying these models to detect fraud, it is not enough to obtain good predictive results: it is essential to understand how these predictions were generated. This concept of interpretability is key to our study. We understand this as the ability to understand the decisions of a model, with a trade-off between this concept and explainability. The latter refers to methods that allow these decisions to be justified in complex models (Ali et al., 2025).

Throughout the literature, we identified two approaches to address interpretability. On the one hand, as highlighted in the previous section, there are intrinsically interpretable models, such as logistic regression or simple decision trees, whose structure allows for a direct reading of the relationships between variables. On the other hand, to explain more powerful but opaque models, a series of post-hoc interpretability techniques have been developed. Tools such as SHAP (Shapley Additive Explanations), LIME (Local Interpretable Model-Agnostic Explanations), and PDP (Partial Dependence Plots) break down predictions and show the individual contribution of each variable.

Delving deeper into these tools, we observe that SHAP uses game theory principles to assign a value of importance to each attribute, offering local and global explanations. On the other hand, LIME generates simplified linear models around a specific observation (Hakkoum et al., 2024: 4). Although they do not make models intrinsically transparent, they do improve their acceptability and auditability, as demonstrated by Gezici and Tarhan (2022). Thus, we observe the need for a balance between the predictive power of complex models and the clarity of simpler ones.

3. RESEARCH DESIGN: METHODOLOGICAL DESCRIPTION AND DATA SIMULATION

Once we have defined the state of the art and approached the subject of this academic work, we must emphasize the data that we are going to make available in it. Thus, based on the limitations and potentialities identified in the conceptual framework, we propose that this study be carried out through an experiment in a controlled environment that allows for the comparison, under equal conditions, of traditional statistical models and ML techniques.

Through the artificial generation of a database, a set of observations representing insurance claims will be designed, characterized by multiple independent variables (such as the age of the insured, type of claim, amount claimed, etc.) and a binary dependent variable that indicates whether or not we are dealing with fraud. The unit of analysis will therefore be each individual simulated observation, allowing for a direct analysis of the performance and behavior of the models. Once this simulation has been carried out, these models will be evaluated with respect to two axes: predictive accuracy—measured with AUC-ROC, precision, F1-score, etc.—and interpretability—either intrinsic or through post-hoc explanations.

These two distinct databases, which will be used to comparatively evaluate the performance of traditional statistical models and ML algorithms, will have the following structure. The first will consist solely of linear relationships between the explanatory variables and the response variable, while the second will deliberately incorporate an additional nonlinear term in the generation of the latent variable, introducing structural complexity. The aim is to measure the predictive performance of each model while comparing their degree of interpretability.

3.1. Simulation 1: data with linear features

This first phase will generate a database with a linear structure and a sample size of 5,000 observations. The set will include 10 numerical predictor variables and two additional categorical variables (gender and type of claim), in addition to the binary response variable "fraud/no fraud." The numeric variables will be simulated independently from a standard normal distribution N(0,1), taking as initial inspiration the typology developed by Pesántez-Narváez (2019). The categorical gender variable will be generated using a binomial distribution with a probability of 0.5 for male or female. The categorical variable claim type will be generated from a multinomial distribution with four categories:

collision 50%, theft 20%, fire 10%, and other 20%. The error will also follow a normal distribution.

All of this will make it possible to generate an optimal experimental situation to test the fit of the models. Based on these variables, a continuous latent variable Y will be constructed, defined by:

$$\mathbf{Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_k X_k + \epsilon ,}$$

To convert Y into a binary response variable Y ϵ {0,1}, the logistic function will be applied as a transformation. Subsequently, the output variable Y will be sampled as a realization of a Bernoulli variable with parameters p = P (Y = 1), that is:

$$\mathbf{Y \sim Bernoulli(p)}$$

3.2. Simulation 2: data with nonlinear features

On the other hand, the previous simulation will be extended by including a new variable $X_{k+1} \sim N(0,1)$ whose effect on the latent variable cannot be captured by a linear combination. In this scenario, the latent variable Y will be given by:

$$\mathbf{Y^* = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_k X_k + f(X_{k+1}) + \epsilon,}$$

Where $f(X_{k+1})$ is an extension given by the introduction of nonlinear terms in the dataset and will depend on the pattern that is finally simulated. This term introduces nonlinearity in the relationship between the explanatory variables and the probability of response. The variables that coincide between the two datasets will be constructed in the same way as in Simulation 1, the fundamental difference being that the calculation of the probability of fraud incorporates a nonlinear term defined as:

$$\mathbf{Z_i = 1.2 \cdot \sin(\pi \cdot pkmnig_i / 100) + 0.6 \cdot \log(1 + kmtotal_i)/10 + 0.8 \cdot (pkmexc_i / 100)^2 - 0.7 \cdot 1\{ageveh_i > 10\} + 0.5 \cdot (car\_power_i / 300) \cdot (pkmexc_i / 100) + 1.0 \cdot \sin(\pi \cdot telem\_score_i / 100)}$$

4. SPECIFIC OBJECTIVES AND RESEARCH DESIGN

Continuing with the methodological proposal, we must return to the general research hypothesis to define the research design, whereby we consider that Machine Learning techniques offer advantages in predictive performance over traditional statistical models but require additional mechanisms to achieve comparable levels of interpretability. To test this, we opted for a controlled synthetic data simulation methodology to accurately evaluate the behavior of different models under perfectly defined conditions.

In turn, we must compensate for this strategy by formulating specific objectives that will guide the subsequent process of operationalizing key dimensions and selecting analysis techniques. Thus, considering the general objective of the study—to compare the predictive performance of machine learning techniques with traditional statistical models and propose additional mechanisms that guarantee comparable levels of interpretability—we propose the following specific objectives:

- **Specific Objective 1:** Quantify the advantages in predictive performance of machine learning techniques compared to traditional statistical models and develop additional mechanisms that guarantee comparable levels of interpretability.
- **Specific Objective 2:** To evaluate the extent to which the use of post-hoc tools improves the interpretability of machine learning models, comparing their level of transparency with that of traditional linear models.
- **Specific Objective 3:** Identify and evaluate machine learning model configurations that achieve an optimal balance between predictive performance and interpretability, making them more viable for application in the insurance sector.

Taking all this into account, and to ensure a systematic evaluation linked to the objectives of this research, we establish the following matrix for the operationalization of objectives in Table I.

TABLE I. Operationalization of the objectives of this study

Specific objective	Technique or method	Dimension	Indicator	Operationalizatio n
1. Quantify the advantages in predictive performance of ML techniques compared to traditional	Comparative application of Logistic Regression, Random Forest, XGBoost, and SVM.	Performance	AUC-ROC	Calculation of AUC-ROC for each model in the test set
			PR-AUC	Calculation of PR-AUC for each

models and develop interpretabilit y mechanisms				model in the test set
			F1-Score	Harmonic mean between precision and recall
			Recall	Proportion of actual positive cases correctly identified
2. Evaluate the extent to which the use of post-hoc tools improves the interpretabilit y of ML models compared to traditional models	Implementation of post-hoc techniques (SHAP, DALEX) on the models.	Interpretability	Number of variables explained with logical meaning	Count of variables with significant and expected impact
			Stability of explanations	Comparison of SHAP and DALEX values
3. Identify and evaluate ML model configuration s that strike a balance between predictive performance and interpretabilit y	Joint analysis of metrics and performance and interpretability results	Performance/explainabili ty trade-off	Comparison and subjective interpretatio n of results	Comparison and subjective interpretation of results

Source: own elaboration.

To conclude this methodological section, we must specify that the proposed design is based on the principle of parsimony, which acts as the guiding criterion for the methodological approach adopted. This principle, as indicated by López-Roldán and Fachelli (2016), implies a necessary "loss of information and gain in significance" (p. 8). Within this framework, it is assumed that it is not possible to compare models that are completely different in all their dimensions, but it is possible to establish reasonable analytical criteria that maximize the usefulness and clarity of the analysis. As Beltrán (1985: 16) points out, "if the demands of analogy were taken to their logical conclusion, any comparative study would become impossible".

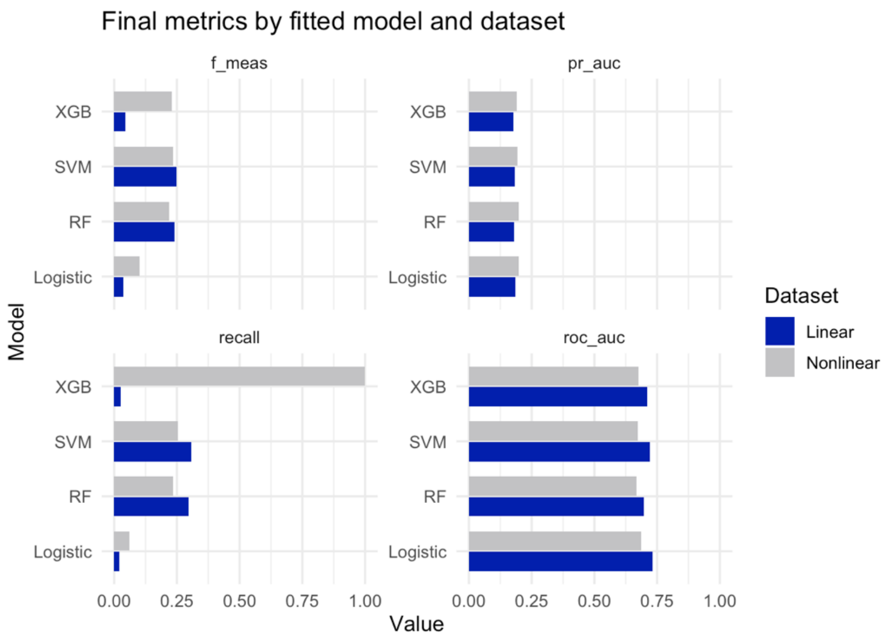
5. RESULTS

Based on the above, we have the relevant information to respond to the research objectives. Following the structure outlined in the previous section, after creating two datasets with linear and nonlinear terms, we present results according to scenario, concluding this section with a comparison of models—logistic, Random Forest, XGBoost, and SVM. It should be noted that the hyperparameters used were chosen after k-fold stratified cross-validation with the PR-AUC target metric if there is imbalance and AUC if there is not. This search was carried out in two stages: broad random grid and fine-tuning around the optimum, and all preprocessing was retrained within each fold to avoid information leakage. The code in "Rmd" and "html" formats corresponding to the results to be presented is publicly available in the corresponding GitHub repository: https://github.com/pablormd/masterthesis-MUCSS-PRM.

In this context, first, the aggregate results show that, with the aforementioned thresholds selected and taking into account the linear scenario, Logistic Regression offers the highest overall ranking (AUC 0.7321 and PR-AUC 0.186). In contrast, we observe lower sensitivity and F1 values. RF and SVM increase detection, with the latter obtaining the highest F1 and recall (F1 0.247 and recall 0.307), while RF values are similar. Regarding XGB, it maintains AUC PR-AUC values close to RF, but with lower recall and F1 (recall 0.025 and F1 0.043). If we review these results in the non-linear scenario, Logit retains the best AUC and PR-AUC (AUC 0.687 and PR-AUC), but still lags in recovery. XGB maximizes sensitivity at the expense of precision, which places its F1 result at 0.229. Compared to SVM, it achieves the best F1 of the set (0.236) with recall 0.255, while RF offers an intermediate compromise (F1 0.218 and recall 0.236).

Thus, once the main results of the different models have been observed, if the criterion is ranking (AUC/PR-AUC), Logit is preferable in both scenarios. If the criterion is positive capture and overall balance, SVM and RF dominate in linear and SVM together with XGB in nonlinear. In turn, we must point out that the choice of different thresholds explains part of the differences in recall and F1. Similarly, everything indicated here can be seen visually and summarized in Figure II:

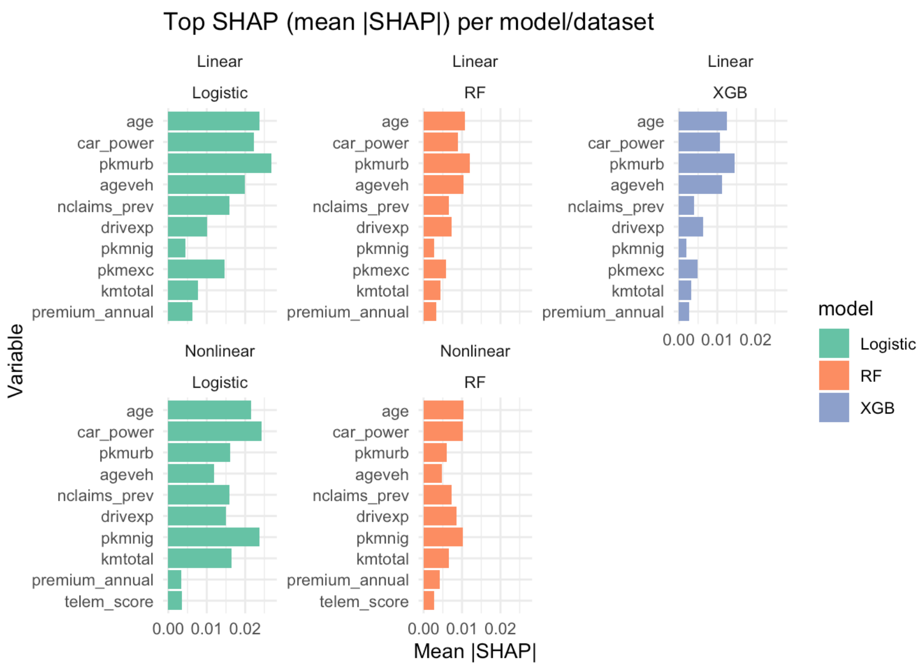
FIGURE II.



Source: own elaboration.

Continuing with the presentation of results, we delve into the second major aspect, which refers to local and global interpretations using SHAP and DALEX. SHAP consistently identifies the set of most influential variables according to the model. Thus, if we look at the five models with the variables that have the greatest influence on them, we see the result that is graphically available in Figure III.

FIGURE III.



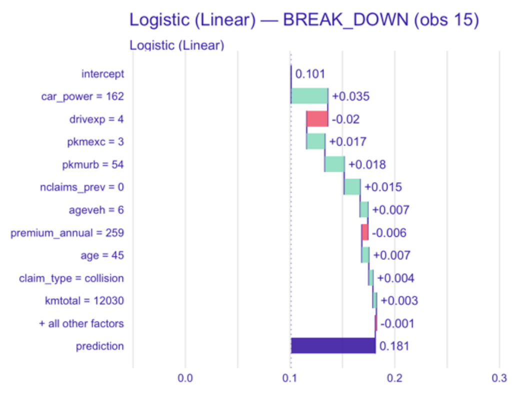
Source: own elaboration.

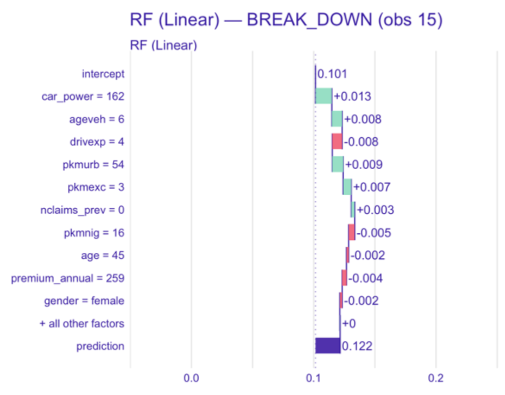
This figure shows the global importance measured as the average absolute value of SHAP in terms of magnitude, not direction. About linear models, although the order varies slightly, the core variables remain constant except for these small variations in order. This pattern is maintained in the nonlinear scenario: *age* and *car_power* are the most influential in the model, along with *pkmurb* and ageveh. *Telem_score*, -the variable with the nonlinear relationship- also appears, with a small contribution at the bottom of the ranking.

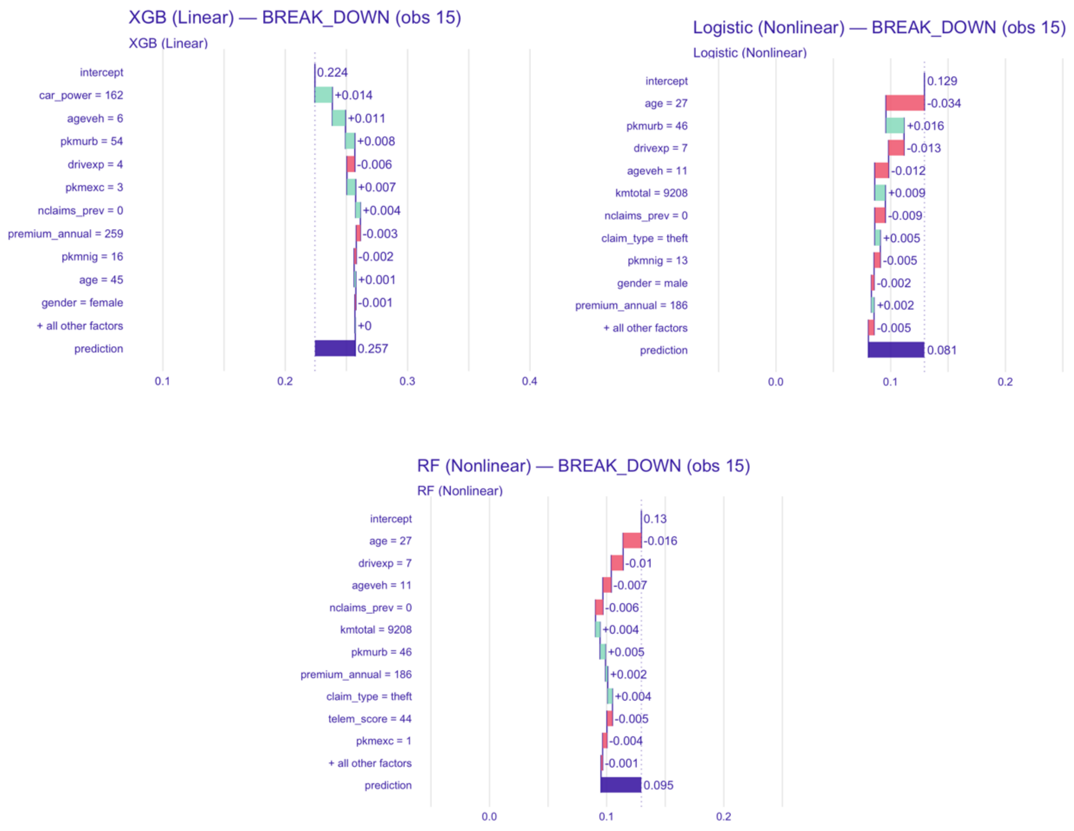
The similarity of rankings between models and scenarios suggests stability among key variables and coherence with the generating function. Since the SHAP result is on a probability scale, a mean SHAP value of, for example, 0.0269 implies that this variable shifts the predicted probability by 2.69 percentage points from the baseline. These results are low to moderate, indicating that no variable dominates the model. This can be verified comparatively; for example, *pkmurb* in linear logistic is only 1.7x greater than the fifth variable in that model.

Delving deeper into these explanations, we arrive at the DALEX results at the local level. As can be seen in Figures IV, V, VI, VII, and VIII, which refer to the top five indicated above, the breakdown starts from the base probability and adds contributions up to the prediction. In the linear simulation, the variables that, on average, increase the probability the most are *car_power, pkmurb, pkmexc*, and *ageveh*. The one that reduces it the most is *drivexp*. On the other hand, in nonlinear, negative effects of *age, drivexp,* and *ageveh* predominate, with *pkmurb* contributing moderately positive in some models.

FIGURES IV, V, VII, AND VIII







Source: own elaboration.

6. CONCLUSIONS AND DISCUSSION

After everything presented throughout this research project, we believe we have the relevant information to conclude it. Thus, we shall return to our research questions and objectives to analyze their status:

- **Research question.** What differences exist between traditional statistical models and machine learning techniques in terms of performance and interpretability in the field of insurance fraud detection? There are clear differences between the Logit model and those made with ML. Logit leads the overall ranking, while ML raises the capture to the calibrated thresholds. In addition, interpretability is intrinsic in Logit and operational in ML through SHAP/DALEX.
- **General Objective.** Compare the predictive performance of machine learning techniques with traditional statistical models and propose and evaluate mechanisms to improve interpretability. By comparing the logarithmic model with machine learning models in terms of a series of relevant measures, we consider this section to be fulfilled. In addition, both SHAP and DALEX provide useful global and local outputs which, although they do not guarantee intrinsic interpretability equivalent to that of Logit models, do provide operational comparability.
- **Specific Objective 1.** Quantify the advantages in predictive performance of ML techniques over traditional models and develop interpretability mechanisms. As can be seen in the results section, performance has been quantified, but there is no universal superiority of ML that allows clear advantages to be observed. The advantages appear in recall and F1 conditioned on the threshold.
- **Specific Objective 2.** Evaluate the extent to which the use of post-hoc tools improves the interpretability of ML models compared to traditional models. Post-hoc tools improve the operational interpretability of ML despite the low contribution results of the variables. SHAP offers stable global importance, and DALEX explains consistent local contributions.
- **Specific Objective 3.** Identify and evaluate ML model configurations that achieve a balance between predictive performance and interpretability. Configurations with a good balance are identified: SVM and RF achieve good F1/recall while maintaining AUC values close to those achieved by Logit. On the other hand, XGB is preferable if extreme sensitivity is prioritized. Finally, Logit would be the benchmark if we followed the principle of parsimony indicated above. This

reveals that the final selection will depend on the operational objective and the decision threshold to be taken into account.

6.1. Limitations and proposals for improvement

To conclude this project, we present a final section that brings together limitations and suggestions for improvement. First, we must emphasize that this work finds its analytical and methodological usefulness by focusing on interpretability and performance, which initially leads to the need to generate simulated datasets to carry out the bulk of the work. Although this is a good methodological strategy and has already been used by experts in the field, it may mean that the results are not entirely reliable or sufficiently positive as those we might have found when performing this exercise with real data. This is what has led us to accept results in metrics that, in a real situation of fraud detection through models, might not have been considered valid. Even so, the initial configuration of the variables has attempted to respond to a simulation that is as close to reality as possible. Thus, an initial proposal for improvement would be to propose the use of real data from insurance companies to repeat this research.

In line with this idea of the need for real data, we also note a lack of external validation of our results, as well as the need to introduce data with a real historical time drift. Finally, we highlight the possibility of proposing different types of models for the future, to expand the interpretative scope of this project. Similarly, we consider that, based on the criteria defined at the outset, this project responds accurately and reliably to a real need in the insurance sector.

7. BIBLIOGRAPHIC REFERENCES

Adedayo, A. F., Odusanya, O. A., Adesina, O. S., Adeyiga, J. A., Okagbue, H. I., & Oyewole, O. (2023). Prediction of automobile insurance fraud claims using machine learning. *The Scientific Temper*, 14(3), 756–762. https://doi.org/10.58414/SCIENTIFICTEMPER.2023.14.3.29.

Ali, A. M. O., Mohammed, A. A. M. A., & Haqi, M. M. N. (2025). Explainability in AI: Interpretable Models for Data Science. *International Journal for Research in Applied Science and Engineering Technology* (IJRASET), 13(4), 112–118. https://doi.org/10.22214/ijraset.2025.66968.

Badhoutiya, A., Laxminarayamma, K., Verma, R. P., Rao, A. L. N., Shrivastava, A., & Khan, A. K. (2023). Random Forest Classification in Healthcare Decision Support for Disease Diagnosis. *International Journal of Early Childhood Special Education* (INT-JECSE), 15(6), 3244–3255. ISSN: 1308-5581. https://www.int-jecse.net

Beltrán, M. (1985). Cinco vías de acceso a la realidad social. *Revista Española de Investigaciones Sociológicas*, 29, 7-41 https://doi.org/10.2307/40183084.

Chakravarty, K., & Singh, J. (2022). Optimizing Defect Removal Efficiency by Defect Prediction using Machine Learning. *2022 OITS International Conference on Information Technology (OCIT)*, 205–210. https://doi.org/10.1109/OCIT56763.2022.00047.

Ding, N., Ruan, X., Wang, H., & Liu, Y. (2024). Automobile insurance fraud detection based on PSO-XGBoost model and interpretable machine learning method. *Insurance: Mathematics and Economics*. https://doi.org/10.1016/j.insmatheco.2024.11.006.

Garrote, A. (2025, March 23). Insurance fraud continues to rise and reaches 1.9% in 2024. La Razón. Retrieved from https://www.larazon.es/economia/fraude-seguro-continua-ascenso-eleva-19-2024_2025032367e02206550aee00012e9e5c.html

Gezici, B., & Tarhan, A. K. (2022). *Explainable AI for Software Defect Prediction with Gradient Boosting Classifier*. In *Proceedings of the 7th International Conference on Computer Science and Engineering (UBMK 2022)* (pp. 49–54). Diyarbakır, Turkey. IEEE. https://doi.org/10.1109/UBMK55850.2022.9919490.

Guo, Y., Zhou, Y., Hu, X., & Cheng, W. (2019). Research on recommendation of insurance products based on random forest. *Proceedings of the 2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, 256–260. https://doi.org/10.1109/MLBDBI48998.2019.00069.

Hakkoum, H., Idri, A., & Abnane, I. (2024). Global and local interpretability techniques of supervised machine learning black box models for numerical medical data. *Engineering Applications of Artificial Intelligence*, 127, 107829. https://doi.org/10.1016/j.engappai.2023.107829.

Harrell, F. 2001. Regression Modeling Strategies. New York: Springer.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: with applications in R* (Vol. 103). New York: Springer.

Kumar, T., Deep, U., Shoiab, S., Atif, S., Bhatnagar, T., & Ramesh, T. (2021). Insurance fraud detection using machine learning. *International Journal of Advanced Information and Communication Technology, 8*(1), 1–4. https://www.ijaict.com/journals/ijaict/ijaict_abstract/2021_volume08/2021_v8i1/v8i1_1.html .

Latiesa, M. (1991). Introducción a los modelos logarítmicos lineales. *Revista de Sociología*, 37, 97–112. https://doi.org/10.5565/rev/papers/v37n0.1598.

Law 21/2007, of July 11, amending the revised text of the Law on civil liability and insurance in the circulation of motor vehicles, approved by Royal Legislative Decree 8/2004, of October 29, and the revised text of the Law on the regulation and supervision of private insurance, approved by Royal Legislative Decree 6/2004, of October 29. (2007). *Official State Gazette,* 166, 29946–29952. https://www.boe.es/eli/es/l/2007/07/11/21.

López-Roldán, P. y Fachelli, S. (2016). Análisis de Clasificación. En P. López-Roldán, y S. Fachelli (2016), Metodología de la Investigación Social Cuantitativa. Bellaterra (Cerdanyola del Vallès). Capítulo III.12. 1ª edición. Versión 2. http://ddd.uab.cat/record/142929

Óskarsdóttir, M., Ahmed, W., Antonio, K., Baesens, B., Dendievel, R., Donas, T., & Reynkens, T. (2020). Social network analytics for supervised fraud detection in insurance [Preprint]. University of Southampton E-Prints. https://eprints.soton.ac.uk/448443/

Patil, K. S & Godbole, A. (2018). A survey on machine learning techniques for insurance fraud prediction. *International Journal of Advance Engineering and Research Development, 8*(11), 4358–4363. https://doi.org/10.29042/2018-4358-4363.

Pesantez-Narvaez, J., Guillen, M., & Alcañiz, M. (2019). Predicting Motor Insurance Claims Using Telematics Data—XGBoost versus Logistic Regression. Risks, 7(2), 70. https://doi.org/10.3390/risks7020070.

Plaisant van der Wal, R. (2018). Detecting fraudulent insurance claims using machine learning methods (Master's thesis). Available at: https://consensus.app/papers/detecting-fraudulent-insurance-claims-using-machine-wal/0084edb932715dc08f1d1932eca19e5d/?

Polat, O., & Reva, C. (2018). *A case analysis: A self-mutilation case as insurance fraud from Forensic Medicine Viewpoint*. *International Journal of Law and Psychiatry*, 58, 72–75. https://www.gavinpublishers.com/article/view/a-case-analysis-a-selfmutilation-case-as-insurance-fraud-from-forensic-medicine-viewpoint.

Priya, K., & Pushpa, S. (2017). A survey on fraud analytics using predictive model in insurance claims. *International Journal of Pure and Applied Mathematics, 116*(21), 629–640.

Saddi, V. R., Gnanapa, B., Boddu, S., Gupta, K., & Logeshwaran, J. (2023). *Insurance fraud prediction: An analysis using supervised machine learning algorithms*. *Expert Systems with Applications, 213*, 118938. https://ieeexplore.ieee.org/document/10531397?denied=.

Sharma, Shubham, & Manu Vardhan. Hyperparameter Tuned Hybrid Convolutional Neural Network (H-CNN) for Accurate Plant Disease Classification. In 2023 International Conference on Communication, Circuits, and Systems (IC3S), pp. 1-6. IEEE, 2023.

8. APPENDICES

8.1. Code and modeling pipeline in R

~~~
```{r, warning=FALSE, message=FALSE}

# List of required packages
required_packages <- c(
  "tidymodels", "dplyr", "ggplot2", "stringr", "purrr", "tibble", "tidyr",
  "dials", "finetune", "fastshap", "lime", "yardstick", "recipes", "stats",
  "tibble", "DALEX", "knitr"
)

# Check which packages are not installed
missing_packages <- required_packages[!(required_packages %in%
                                    installed.packages()[, "Package"])]

# Install missing packages
if (length(missing_packages) > 0) {
  options(repos = c(CRAN = "https://cloud.r-project.org/")) # Set a CRAN mirror
  install.packages(missing_packages)
}

# Load the packages
invisible(lapply(required_packages, library, character.only = TRUE))

~~~

# Synthetic Data Generators

```{r}

generate_linear_data <- function(n = 10000, seed = 1) {
  set.seed(seed)
  age <- pmin(pmax(round(rnorm(n, 42, 12)), 18), 90)
  ageveh <- pmin(pmax(round(rnorm(n, 7, 4)), 0), 25)
  drivexp <- pmin(pmax(round(rnorm(n, 15, 8)), 0), pmax(age - 18, 0))
  pkmurb <- round(100 * rbeta(n, 5, 3))
  pkmnig <- round(100 * rbeta(n, 2, 12))
  pkmexc <- round(100 * rbeta(n, 1.5, 18))
  kmtotal <- round(rlnorm(n, meanlog = log(12000), sdlog = 0.5))
  nclaims_prev <- pmin(rpois(n, 0.3), 6)
  car_power <- pmin(pmax(round(rnorm(n, 120, 35)), 55), 300)
  premium_annual <- round(150 + 0.006 * kmtotal + 0.8 * nclaims_prev +
                          0.4 * pkmexc + 0.2 * (car_power - 120) + rnorm(n, 0, 50))
  premium_annual[premium_annual < 80] <- 80

  gender <- sample(c("male","female"), n, TRUE)
  claim_type <- sample(c("collision","theft","fire","other"), n, TRUE, prob = c(0.5,0.2,0.1,0.2))

  predictors <- data.frame(
    age, ageveh, drivexp, pkmurb, pkmnig, pkmexc,
    kmtotal, nclaims_prev, car_power, premium_annual
  )

  x_std <- scale(predictors)
  betas <- runif(ncol(predictors), -2, 2)
  epsilon <- rnorm(n)
```

```
  y_latent <- as.numeric(x_std %*% betas) + epsilon
  prob <- plogis(y_latent)
  y <- rbinom(n, 1, pmin(pmax(prob * 0.2, 0), 1))

  dplyr::bind_cols(
    predictors,
    gender = gender,
    claim_type = claim_type,
    y = factor(y, levels = c(0,1), labels = c("no_fraud","fraud"))
  )
}

generate_nonlinear_data <- function(n = 10000, seed = 1) {
  set.seed(seed)
  age <- pmin(pmax(round(rnorm(n, 42, 12)), 18), 90)
  ageveh <- pmin(pmax(round(rnorm(n, 7, 4)), 0), 25)
  drivexp <- pmin(pmax(round(rnorm(n, 15, 8)), 0), pmax(age - 18, 0))
  pkmurb <- round(100 * rbeta(n, 5, 3))
  pkmnig <- round(100 * rbeta(n, 2, 12))
  pkmexc <- round(100 * rbeta(n, 1.5, 18))
  kmtotal <- round(rlnorm(n, meanlog = log(12000), sdlog = 0.5))
  nclaims_prev <- pmin(rpois(n, 0.3), 6)
  car_power <- pmin(pmax(round(rnorm(n, 120, 35)), 55), 300)
  premium_annual <- round(150 + 0.006 * kmtotal + 0.8 * nclaims_prev +
                          0.4 * pkmexc + 0.2 * (car_power - 120) + rnorm(n, 0, 50))
  premium_annual[premium_annual < 80] <- 80

  gender <- sample(c("male","female"), n, TRUE)
  claim_type <- sample(c("collision","theft","fire","other"), n, TRUE, prob = c(0.5,0.2,0.1,0.2))

  telem_score <- round(100 * rbeta(n, 2.5, 4.5))

  predictors <- data.frame(
    age, ageveh, drivexp, pkmurb, pkmnig, pkmexc,
    kmtotal, nclaims_prev, car_power, premium_annual,
    telem_score
  )

  x_std <- scale(predictors)
  betas <- runif(ncol(predictors), -2, 2)
  epsilon <- rnorm(n)

  nonlinear_term <-
    1.2 * sin(pi * (pkmnig / 100)) +
    0.6 * log1p(kmtotal) / 10 +
    0.8 * (pkmexc / 100)^2 -
    0.7 * as.numeric(ageveh > 10) +
    0.5 * (car_power / 300) * (pkmexc / 100) +
    1.0 * sin(pi * (telem_score / 100))

  y_latent <- as.numeric(x_std %*% betas) + nonlinear_term + epsilon
  prob <- plogis(y_latent)
  y <- rbinom(n, 1, pmin(pmax(prob * 0.2, 0), 1))

  dplyr::bind_cols(
    predictors,
    gender = gender,
    claim_type = claim_type,
```

~~~
    y = factor(y, levels = c(0,1), labels = c("no_fraud","fraud"))
  )
}

```

```{r}

# Datasets simulation
linear_df    <- generate_linear_data(n = 10000, seed = 1)
nonlinear_df <- generate_nonlinear_data(n = 10000, seed = 1)

# Balance check
linear_df  |> count(y)  |> mutate(prop = n / sum(n))
nonlinear_df |> count(y) |> mutate(prop = n / sum(n))

# Train/Test + calibration (umbrals)
set.seed(3)
lin_split <- initial_split(linear_df, prop = 0.8, strata = y)
linear_train <- training(lin_split)
linear_test  <- testing(lin_split)

nl_split <- initial_split(nonlinear_df, prop = 0.8, strata = y)
nonlinear_train <- training(nl_split)
nonlinear_test  <- testing(nl_split)

```

# Preprocessing recipes

```{r}

rec_log <- recipe(y ~ ., data = linear_train) |>
  step_dummy(all_nominal_predictors()) |>
  step_normalize(all_numeric_predictors())

rec_rf  <- rec_log
rec_xgb <- rec_log
rec_svm <- rec_log

rec_log_nl <- recipe(y ~ ., data = nonlinear_train) |>
  step_dummy(all_nominal_predictors()) |>
  step_normalize(all_numeric_predictors())

rec_rf_nl  <- rec_log_nl
rec_xgb_nl <- rec_log_nl
rec_svm_nl <- rec_log_nl

```

# Model Specifications

```{r}

log_reg_spec <- logistic_reg() |>
  set_engine("glm") |>
  set_mode("classification")
~~~

~~~
rf_spec <- rand_forest(
  mtry  = tune(),
  trees = tune(),
  min_n = tune()
) |>
  set_engine("ranger", importance = "impurity") |>
  set_mode("classification")

xgb_spec <- boost_tree(
  mtry          = tune(),
  trees         = tune(),
  min_n         = tune(),
  tree_depth    = tune(),
  learn_rate    = tune(),
  loss_reduction= tune(),
  sample_size   = tune()
) |>
  set_engine("xgboost") |>
  set_mode("classification")

svm_spec <- svm_rbf(
  cost      = tune(),
  rbf_sigma = tune()
) |>
  set_engine("kernlab") |>
  set_mode("classification")

wf <- function(spec, rec) {
  workflow() |> add_model(spec) |> add_recipe(rec)
}

```

# Hyperparameters Grids and Cross-Validation

```{r}

set.seed(4)
folds_lin <- vfold_cv(linear_train, v = 4, strata = y)
folds_nl  <- vfold_cv(nonlinear_train, v = 4, strata = y)

# Calculate how many predictors remain after preprocessing and set the mtry range
lin_baked <- bake(prep(rec_log), new_data = linear_train) |> select(-y)
nl_baked  <- bake(prep(rec_log_nl), new_data = nonlinear_train) |> select(-y)

# RF
rf_grid <- grid_space_filling(
  finalize(mtry(), lin_baked),
  trees(),
  min_n(),
  size = 20
)

# XGB
xgb_grid <- grid_space_filling(
  trees(), tree_depth(), learn_rate(), loss_reduction(),
  sample_prop(), finalize(mtry(), lin_baked), min_n(),
  size = 20
~~~

~~~
)

# SVM
svm_grid <- grid_space_filling(
  cost(), rbf_sigma(),
  size = 20
)

# Tune metrics
tune_metrics <- metric_set(roc_auc, pr_auc, accuracy, f_meas, recall, precision)

```

## Hyperparameter Tuning - Linear Dataset

```{r, warning=FALSE}

# Logistic (without tuning)
logit_lin_fit <- fit(wf(log_reg_spec, rec_log), data = linear_train)

# RF
set.seed(5)
rf_lin_tuned <- tune_grid(
  wf(rf_spec, rec_rf),
  resamples = folds_lin,
  grid = rf_grid,
  metrics = tune_metrics
)
rf_lin_best  <- select_best(rf_lin_tuned, metric = "pr_auc")
rf_lin_final <- finalize_workflow(wf(rf_spec, rec_rf), rf_lin_best) |>
  fit(linear_train)

# XGB
set.seed(6)
xgb_lin_tuned <- tune_grid(
  wf(xgb_spec, rec_xgb),
  resamples = folds_lin,
  grid = xgb_grid,
  metrics = tune_metrics
)
xgb_lin_best  <- select_best(xgb_lin_tuned, metric = "pr_auc")
xgb_lin_final <- finalize_workflow(wf(xgb_spec, rec_xgb), xgb_lin_best) |>
  fit(linear_train)

# SVM
set.seed(7)
svm_lin_tuned <- tune_grid(
  wf(svm_spec, rec_svm),
  resamples = folds_lin,
  grid = svm_grid,
  metrics = tune_metrics
)
svm_lin_best  <- select_best(svm_lin_tuned, metric = "pr_auc")
svm_lin_final <- finalize_workflow(wf(svm_spec, rec_svm), svm_lin_best) |>
  fit(linear_train)

```
~~~

~~~
## Hyperparameter Tuning - Non-Linear Dataset

```{r, warning=FALSE}

# Logistic (without tuning)
logit_nl_fit <- fit(wf(log_reg_spec, rec_log_nl), data = nonlinear_train)

# RF
set.seed(8)
rf_nl_tuned <- tune_grid(
  wf(rf_spec, rec_rf_nl),
  resamples = folds_nl,
  grid = rf_grid,
  metrics = tune_metrics
)
rf_nl_best  <- select_best(rf_nl_tuned, metric = "pr_auc")
rf_nl_final <- finalize_workflow(wf(rf_spec, rec_rf_nl), rf_nl_best) |>
  fit(nonlinear_train)

# XGB
set.seed(9)
xgb_nl_tuned <- tune_grid(
  wf(xgb_spec, rec_xgb_nl),
  resamples = folds_nl,
  grid = xgb_grid,
  metrics = tune_metrics
)
xgb_nl_best  <- select_best(xgb_nl_tuned, metric = "pr_auc")
xgb_nl_final <- finalize_workflow(wf(xgb_spec, rec_xgb_nl), xgb_nl_best) |>
  fit(nonlinear_train)

# SVM
set.seed(10)
svm_nl_tuned <- tune_grid(
  wf(svm_spec, rec_svm_nl),
  resamples = folds_nl,
  grid = svm_grid,
  metrics = tune_metrics
)
svm_nl_best  <- select_best(svm_nl_tuned, metric = "pr_auc")
svm_nl_final <- finalize_workflow(wf(svm_spec, rec_svm_nl), svm_nl_best) |>
  fit(nonlinear_train)

```

# Engine + Recipe Probability Extraction and F1-Optimized Thresholding

```{r}

# Engine test predictions + recipe
get_prob_df_engine <- function(fit_engine, recipe_obj, new_data, outcome = "y") {
  processed <- bake(prep(recipe_obj, retain = TRUE), new_data = new_data)
  tibble(
    !!outcome := processed[[outcome]],
    .pred_fraud = predict(fit_engine, new_data = processed, type = "prob")[[".pred_fraud"]]
  )
}
~~~

```
find_best_threshold <- function(prob_df,
                                grid = seq(0.3, 0.60, by = 0.01),
                                event_level = "second",
                                min_pred_pos = 10) {
  res <- map_dfr(grid, function(t) {
    pred <- factor(ifelse(prob_df$.pred_fraud >= t, "fraud", "no_fraud"),
                   levels = c("no_fraud","fraud"))
    n_pos <- sum(pred == "fraud")
    if (n_pos < min_pred_pos) {
      return(tibble(threshold = t, accuracy = NA_real_, recall = NA_real_,
                    precision = NA_real_, f_meas = NA_real_, pred_pos = n_pos))
    }
    truth <- prob_df$y
    tibble(
      threshold = t,
      accuracy  = accuracy_vec(truth, pred),
      recall    = recall_vec(truth, pred, event_level = event_level),
      precision = precision_vec(truth, pred, event_level = event_level),
      f_meas    = f_meas_vec(truth, pred, event_level = event_level),
      pred_pos  = n_pos
    )
  }) |> dplyr::filter(is.finite(f_meas))

  if (nrow(res) == 0) {
    thr <- as.numeric(quantile(prob_df$.pred_fraud, 0.85, na.rm = TRUE))
    return(list(threshold = thr, table = tibble(threshold = thr)))
    }
  best <- res |> arrange(dplyr::desc(f_meas), dplyr::desc(recall), dplyr::desc(precision)) |>
dplyr::slice(1)
  list(threshold = best$threshold[[1]], table = res)
}


# Linear
lin_prob_log <- get_prob_df_engine(logit_lin_fit$fit$fit, rec_log,  linear_test)
thr_log_lin  <- find_best_threshold(lin_prob_log)$threshold

lin_prob_rf  <- get_prob_df_engine(rf_lin_final$fit$fit,  rec_rf,   linear_test)
thr_rf_lin   <- find_best_threshold(lin_prob_rf)$threshold

lin_prob_xgb <- get_prob_df_engine(xgb_lin_final$fit$fit, rec_xgb,  linear_test)
thr_xgb_lin  <- find_best_threshold(lin_prob_xgb)$threshold

lin_prob_svm <- get_prob_df_engine(svm_lin_final$fit$fit, rec_svm,  linear_test)
thr_svm_lin  <- find_best_threshold(lin_prob_svm)$threshold

# Non Linear
nl_prob_log  <- get_prob_df_engine(logit_nl_fit$fit$fit, rec_log_nl, nonlinear_test)
thr_log_nl   <- find_best_threshold(nl_prob_log)$threshold

nl_prob_rf   <- get_prob_df_engine(rf_nl_final$fit$fit,  rec_rf_nl, nonlinear_test)
thr_rf_nl    <- find_best_threshold(nl_prob_rf)$threshold

nl_prob_xgb  <- get_prob_df_engine(xgb_nl_final$fit$fit, rec_xgb_nl, nonlinear_test)
thr_xgb_nl   <- find_best_threshold(nl_prob_xgb)$threshold

nl_prob_svm  <- get_prob_df_engine(svm_nl_final$fit$fit, rec_svm_nl, nonlinear_test)
thr_svm_nl   <- find_best_threshold(nl_prob_svm)$threshold
```

~~~
tibble(
  dataset = c(rep("Linear",4), rep("Nonlinear",4)),
  model   = rep(c("Logistic","RF","XGB","SVM"), 2),
  threshold = c(thr_log_lin, thr_rf_lin, thr_xgb_lin, thr_svm_lin,
               thr_log_nl,  thr_rf_nl,  thr_xgb_nl,  thr_svm_nl))

```
# Final Metrics with Calibrated Thresholds for Fitted Workflows

```{r}

# Ensure outcome levels
fix_levels <- function(df) { df$y <- factor(df$y, levels = c("no_fraud","fraud")); df }
linear_test     <- fix_levels(linear_test)
nonlinear_test  <- fix_levels(nonlinear_test)

# Thresholds (use calibrated ones if present; otherwise 0.30
get_thr <- function(name, default = 0.30) if (exists(name, inherits = TRUE)) get(name, inherits =
TRUE) else default

thr_lin <- list(
  Logistic = get_thr("thr_log_lin"),
  RF       = get_thr("thr_rf_lin"),
  XGB      = get_thr("thr_xgb_lin"),
  SVM      = get_thr("thr_svm_lin")
)
thr_nl <- list(
  Logistic = get_thr("thr_log_nl"),
  RF       = get_thr("thr_rf_nl"),
  XGB      = get_thr("thr_xgb_nl"),
  SVM      = get_thr("thr_svm_nl")
)

# Evaluation helper
eval_4metrics <- function(truth, prob, pred) {
  tibble(
    roc_auc = roc_auc_vec(truth, prob, event_level = "second"),
    pr_auc  = pr_auc_vec( truth, prob, event_level = "second"),
    recall  = recall_vec( truth, pred, event_level = "second"),
    f_meas  = f_meas_vec( truth, pred, event_level = "second")
  )
}

# Scorer for a single fitted workflow
score_fit <- function(fitted_wflow, new_data, threshold, model_name, dataset_name) {
  prob  <- predict(fitted_wflow, new_data = new_data, type = "prob")$.pred_fraud
  truth <- factor(new_data$y, levels = c("no_fraud","fraud"))
  pred  <- factor(ifelse(prob >= threshold, "fraud", "no_fraud"),
                 levels = c("no_fraud","fraud"))
  eval_4metrics(truth, prob, pred) |>
    mutate(dataset = dataset_name, model = model_name, threshold = threshold, .before = 1)
}

# -Fitted workflows
fits_linear <- list(
  Logistic = logit_lin_fit,
  RF       = rf_lin_final,
~~~

~~~
  XGB      = xgb_lin_final,
  SVM      = svm_lin_final
)
fits_nonlinear <- list(
  Logistic = logit_nl_fit,
  RF       = rf_nl_final,
  XGB      = xgb_nl_final,
  SVM      = svm_nl_final
)

res_lin <- imap_dfr(fits_linear,   ~ score_fit(.x, linear_test,    thr_lin[[.y]], .y, "Linear"))
res_nl  <- imap_dfr(fits_nonlinear,~ score_fit(.x, nonlinear_test, thr_nl[[.y]],  .y, "Nonlinear"))

final_results <- bind_rows(res_lin, res_nl) |>
  relocate(dataset, model, threshold)

kable(final_results, digits = 3)

# Plot
final_results |>
  pivot_longer(cols = c(roc_auc, pr_auc, recall, f_meas),
               names_to = "metric", values_to = "value") |>
  ggplot(aes(x = model, y = value, fill = dataset)) +
  geom_col(position = position_dodge(width = 0.75), width = 0.7) +
  facet_wrap(~ metric, scales = "free_y") +
  scale_fill_manual(values = c(Linear = "#0006ad", Nonlinear = "#c2c2c4")) +
  labs(title = "Final metrics by fitted model and dataset",
       x = "Model", y = "Value", fill = "Dataset") +
  coord_flip() +
  theme_minimal(base_size = 12)

```

# SHAP Local Explanations

```{r}
# 1) SHAP helper with workflow parts (engine + recipe)
compute_shap_wflow <- function(fit_engine,
                               recipe_obj,
                               test_data,
                               y_col = "y",
                               nsim = 50) {

  # Baked data
  baked <- bake(prep(recipe_obj, retain = TRUE), new_data = test_data)
  pred_fun <- function(object, newdata) {
    predict(object, new_data = newdata, type = "prob")[[".pred_fraud"]]
  }

  X <- baked |> select(-all_of(y_col))
  shap_mat <- fastshap::explain(
    object       = fit_engine,
    X            = X,
    pred_wrapper = pred_fun,
    nsim         = nsim
  )

  as_tibble(shap_mat) |>
~~~

```
    summarise(across(everything(), ~ mean(abs(.x), na.rm = TRUE))) |>
    pivot_longer(everything(), names_to = "variable", values_to = "mean_abs_shap") |>
    arrange(desc(mean_abs_shap))
}

# SHAP tables for all models
## Linear
shap_nsim <- 50
shap_top  <- 10

shap_lin_log  <-  compute_shap_wflow(logit_lin_fit$fit$fit,  rec_log,        linear_test,   nsim  =
shap_nsim) |>
  mutate(dataset = "Linear",    model = "Logistic", .before = 1)
shap_lin_rf   <-  compute_shap_wflow(rf_lin_final$fit$fit,     rec_rf,         linear_test,   nsim  =
shap_nsim) |>
  mutate(dataset = "Linear",    model = "RF",       .before = 1)
shap_lin_xgb <- compute_shap_wflow(xgb_lin_final$fit$fit,   rec_xgb,       linear_test,   nsim =
shap_nsim) |>
  mutate(dataset = "Linear",    model = "XGB",      .before = 1)
shap_lin_svm <- compute_shap_wflow(svm_lin_final$fit$fit,   rec_svm,       linear_test,   nsim =
shap_nsim) |>
  mutate(dataset = "Linear",    model = "SVM",      .before = 1)

## Non Linear
shap_nl_log <- compute_shap_wflow(logit_nl_fit$fit$fit,     rec_log_nl,   nonlinear_test, nsim =
shap_nsim) |>
  mutate(dataset = "Nonlinear", model = "Logistic", .before = 1)
shap_nl_rf   <- compute_shap_wflow(rf_nl_final$fit$fit,       rec_rf_nl,    nonlinear_test, nsim =
shap_nsim) |>
  mutate(dataset = "Nonlinear", model = "RF",       .before = 1)
shap_nl_xgb <- compute_shap_wflow(xgb_nl_final$fit$fit,     rec_xgb_nl,  nonlinear_test, nsim
= shap_nsim) |>
  mutate(dataset = "Nonlinear", model = "XGB",      .before = 1)
shap_nl_svm <- compute_shap_wflow(svm_nl_final$fit$fit,     rec_svm_nl,  nonlinear_test, nsim
= shap_nsim) |>
  mutate(dataset = "Nonlinear", model = "SVM",      .before = 1)

shap_table <- bind_rows(
  shap_lin_log, shap_lin_rf, shap_lin_xgb, shap_lin_svm,
  shap_nl_log,  shap_nl_rf,  shap_nl_xgb,  shap_nl_svm
) |>
  group_by(dataset, model) |>
  slice_head(n = shap_top) |>
  ungroup()

shap_table

shap_plot <- bind_rows(
  shap_lin_log, shap_lin_rf, shap_lin_xgb, shap_nl_log,  shap_nl_rf
) |>
  group_by(dataset, model) |>
  slice_head(n = shap_top) |>
  ungroup()

# Best Models SHAP visualization
shap_plot |>
  ggplot(aes(x = reorder(variable, mean_abs_shap), y = mean_abs_shap, fill = model)) +
  geom_col(show.legend = TRUE) +
```

~~~
  coord_flip() +
  facet_wrap(dataset ~ model, scales = "free_y") +
  scale_fill_brewer(palette = "Set2") +
  labs(title = "Top SHAP (mean |SHAP|) per model/dataset",
       x = "Variable", y = "Mean |SHAP|") +
  theme_minimal(base_size = 11)
```

# DALEX Local Explanations

```{r}

# Predict wrapper for workflows
wf_prob <- function(model, newdata) {
  as.numeric(predict(model, new_data = newdata, type = "prob")[[".pred_fraud"]])
}

# Build DALEX explainer from a fitted workflow and raw train data
make_explainer <- function(fitted_wflow, train_df, label) {
  y_num <- as.numeric(train_df$y == "fraud")
  x_df  <- train_df |> dplyr::select(-y)

  DALEX::explain(
    model            = fitted_wflow,
    data             = x_df,
    y                = y_num,
    predict_function = wf_prob,
    label            = label,
    verbose          = FALSE
  )
}

# One-shot local explanation (break_down or shap) for a single observation
explain_local <- function(explainer, test_df, obs_index = 15, type = c("break_down","shap")) {
  type <- match.arg(type)
  x0   <- test_df[obs_index, , drop = FALSE] |> dplyr::select(-y)
  parts <- DALEX::predict_parts(explainer, new_observation = x0, type = type)
  list(parts = parts, plot = plot(parts) + ggplot2::labs(title = paste0(explainer$label, " — ",
toupper(type), " (obs ", obs_index, ")")))
}

# Build explainers for fitted workflows
## Linear
exp_lin_log <- make_explainer(logit_lin_fit, linear_train,    "Logistic (Linear)")
exp_lin_rf  <- make_explainer(rf_lin_final,  linear_train,    "RF (Linear)")
exp_lin_xgb <- make_explainer(xgb_lin_final, linear_train,    "XGB (Linear)")
exp_lin_svm <- make_explainer(svm_lin_final, linear_train,    "SVM (Linear)")

## Non Linear
exp_nl_log  <- make_explainer(logit_nl_fit,  nonlinear_train, "Logistic (Nonlinear)")
exp_nl_rf   <- make_explainer(rf_nl_final,   nonlinear_train, "RF (Nonlinear)")
exp_nl_xgb  <- make_explainer(xgb_nl_final,  nonlinear_train, "XGB (Nonlinear)")
exp_nl_svm  <- make_explainer(svm_nl_final,  nonlinear_train, "SVM (Nonlinear)")

# Generate local explanations
obs_id     <- 15
method_loc <- "break_down"
~~~

```
## Linear
loc_lin_log <- explain_local(exp_lin_log, linear_test,    obs_index = obs_id, type = method_loc)
loc_lin_rf  <- explain_local(exp_lin_rf,  linear_test,    obs_index = obs_id, type = method_loc)
loc_lin_xgb <- explain_local(exp_lin_xgb, linear_test,    obs_index = obs_id, type = method_loc)
loc_lin_svm <- explain_local(exp_lin_svm, linear_test,    obs_index = obs_id, type = method_loc)

## Nonlinear
loc_nl_log  <- explain_local(exp_nl_log,  nonlinear_test, obs_index = obs_id, type = method_loc)
loc_nl_rf   <- explain_local(exp_nl_rf,   nonlinear_test, obs_index = obs_id, type = method_loc)
loc_nl_xgb  <- explain_local(exp_nl_xgb,  nonlinear_test, obs_index = obs_id, type = method_loc)
loc_nl_svm  <- explain_local(exp_nl_svm,  nonlinear_test, obs_index = obs_id, type = method_loc)

# Tidy DALEK tables for each explanation
tidy_loc_tbl <- function(loc_obj, top_n = 6) {
  as_tibble(loc_obj$parts) |>
    transmute(variable = variable_name, contribution = contribution) |>
    arrange(desc(abs(contribution))) |>
    slice_head(n = top_n)
}

tidy_loc_tbl(loc_lin_log)
tidy_loc_tbl(loc_nl_log)
tidy_loc_tbl(loc_lin_rf)
tidy_loc_tbl(loc_nl_rf)
tidy_loc_tbl(loc_lin_xgb)
tidy_loc_tbl(loc_nl_xgb)
tidy_loc_tbl(loc_lin_svm)
tidy_loc_tbl(loc_nl_svm)

print(loc_lin_log$plot)
print(loc_lin_rf$plot)
print(loc_lin_xgb$plot)
print(loc_nl_log$plot)
print(loc_nl_rf$plot)

```