

Final Report

Predicting Insurance Fraud with Machine Learning Techniques

Written by:

Bertrand Tcheuffa
Nathalie Mugrauer
Quy-Manh Jurca-Tsan

April 2025

Table of Contents

Report 1: Data exploration, visualization and pre-processing.....	5
Objectives.....	5
Project Team.....	6
Framework.....	6
Relevance.....	7
Pre-processing and feature engineering.....	9
Handling NaN – values.....	9
Handling ‘?’ – values.....	10
Visualizations and Statistics.....	12
Quantitative features.....	12
Heatmap.....	13
Statistical Test - Quantitative Features.....	14
Qualitative Features.....	15
Feature reduction.....	17
Encoding.....	17
Summary.....	18
Report 2: Modelling and Results using ML.....	20
Introduction.....	20
Defining a suitable evaluation approach for the model.....	20
Modelling: First attempt with the imbalanced data set.....	21
Feature Engineering.....	23
Addressing the class Imbalance.....	24
Modelling with the resampled data set.....	24
Modelling with the RandomOverSampler (ROS) data.....	24
Modelling with the SMOTETomek resampled data.....	27
Feature importance analysis.....	29
Hyperparameter optimization with GridSearch.....	30
Model Optimization with Voting Classifier.....	31
Testing the model’s robustness with stratified k-fold cross validation.....	32
Model Interpretation.....	33
LDA.....	33
Logistic Regression.....	34
SVC.....	34
AdaBoost.....	35
Summary.....	36
Finals Conclusions.....	37
Challenges.....	37
Contributions.....	37
Benchmark.....	38
Achievement of Project Goals.....	38
Contribution to Scientific Knowledge.....	39
Lessons Learned.....	39
Bibliography.....	41

List of Tables

Table 1: Technical overview of the dataset	7
Table 2: Features organisation	7
Table 3: Classification of qualitative features for the choice of the encoding method	17
Table 4: Performance comparison of models between incident_severity encoded as nominal or ordinal variable	24
Table 5: Rebalancing of the training data set	24
Table 6: Models performance comparison between RandomOversampler and SMOTETomek	29
Table 7: Best scorer models performance with GridSearch	31
Table 8: Features importance ranking by models	35

List of Figures

Figure 1: Features Statistical summary	8
Figure 2: Target variable relationship to some features	9
Figure 3: authorities_contacted in relation to fraud_reported	10
Figure 4: incident_type in relation to authorities_contacted	10
Figure 5: police_report_available in relation to fraud_reported	11
Figure 6: property_damaged in relation to fraud_reported	11
Figure 7: incident_type in relation to collision_type	12
Figure 8: Distribution overview of quantitative variables	13
Figure 9: Heatmap of quantitative variables	14
Figure 10: Statistically most relevant quantitative features	14
Figure 11: Kbest identified quantitative variables	15
Figure 12: Statistical significance and strength of association of categorical variables to target variable	16
Figure 13: Distribution of fraud in several categorical variables	16
Figure 14: vehicule_claim_distribution	17
Figure 15: Performance of selected models on the imbalanced data set	21
Figure 16: Confusion matrix of selected models performance on the imbalanced data set	22
Figure 17: Performance of selected models with incident_severity encoded as ordinal on the imbalanced data set	23
Figure 18: Class imbalance of the target variable	24
Figure 19: Models performance with the RandomOversampler applied	25
Figure 20: Models' confusion matrix with RandomOversampler applied	26
Figure 21: Models performance with SMOTETomek applied	27
Figure 22: Models confusion matrix with SMOTETomek applied	28
Figure 23: Features importance analysis	29
Figure 24: Parameters selection for GridSearch	30
Figure 25: GridSearch implementation	31
Figure 26: Ensemble Voting implementation for performance optimization	31
Figure 27: 50-fold cross validation for best scorer models robustness	32
Figure 28: Best scorer models variance	32
Figure 29: SHAP for LDA model	33
Figure 30: SHAP for Logistic Regression	34
Figure 31: SHAP for SVC	34
Figure 32: SHAP for AdaBoost	35

Report 1: Data exploration, visualization and pre-processing

Insurance fraud is one of the main factors that **erode underwriting profits** for insurance companies. Therefore, it is crucial for insurers to identify whether a claim is legitimate or potentially fraudulent. If there is suspicion of fraud, it must be explained and justified before any investigation can begin. This requires resources to analyse data, assess patterns, and build a reasonable basis for suspicion. Due to the time and financial cost associated with fraud investigations, the ability to **quickly and accurately flag potentially fraudulent cases** is essential. This helps reduce investigation costs and minimize unnecessary claim payouts.

This project aims to address this challenge using **structured claim data** and **machine learning techniques**. The dataset "*insurance_claims.csv*" contains anonymized records of insurance claims from various providers, covering vehicle, property, and personal injury cases. Each row represents a single claim with associated features. The incidents span from January 1 to March 1, 2015. The dataset is concise, enabling fast model development, though it may limit generalizability. It includes both numerical and categorical features, requiring thoughtful preprocessing. **Handling missing values and encoding features properly** is particularly important.

From an **economic perspective**, the implications of fraud detection are substantial. The dataset includes 1,000 insurance claims, of which approximately 24.7% are identified as fraudulent. The total claim volume amounts to **\$52.2** million, with an estimated **\$14.9** million attributed to non-legitimate claims. It is important to note that **these figures are based on data from just a two-month period**, highlighting the potentially massive financial exposure insurance companies face even within a short timeframe. Accurately predicting fraudulent cases offers substantial economic value. By identifying high-risk cases early, insurers can reduce financial losses, optimize investigation resources, and streamline validation processes through data-driven thresholds.

Scientifically, it is important to first assess whether the problem requires a complex solution. Fraud detection may appear challenging, but it could be solved with a simple model if the right features and preprocessing steps are applied. The main question is whether the **relationships are linear or more complex**. For linear patterns, basic models like logistic regression may be enough — with proper resampling (e.g. SMOTE) or class weighting. If the data shows non-linear behaviour, tree-based models such as Random Forests or Gradient Boosting could perform better.

Objectives

The primary aim of this project is to develop a data-driven approach for detecting potentially fraudulent insurance claims. Beyond simply building a functional model, the focus lies in ensuring quality, stability, and real-world applicability. The core objectives are:

- Build accurate machine learning models capable of identifying fraudulent claims based on structured insurance data.
- Ensure model stability and reliability by achieving consistently high prediction scores across different subsets and unseen data.
- Apply the full **data science workflow** — from data understanding and preprocessing to feature engineering and model evaluation.

Throughout the process, we aim to combine theoretical knowledge with hands-on implementation to create a model that is both effective and interpretable in a real-world context.

Project Team

Bertrand Tcheuffa

- Background in Mechanical Engineering, starting the Data Science and Machine Learning journey.
- Passionate in learning from data for factual decision making in finance, insurance and health diagnostics.

Nathalie Mugrauer

- Background in financial controlling and business intelligence
- Looking to enhance my data mining skills in order to get the right insights out of data

Quy-Manh Jurca-Tsan

- Background in investigations with exposure to fraud-related contexts
- Self-taught Python skills, applied to personal and work-related tasks
- Looking to deepen technical skills through structured application of ML methods

Framework

The dataset used in this project, titled "*insurance_claims.csv*", is publicly available at the following link: <https://data.mendeley.com/datasets/992mh7dk9y/2>.

The table below provides a technical overview of the dataset used in this project. It includes key information such as the number of records, data types, missing values, and the distribution of the target variable.

Category	Description
Geographic Scope	Claims origination from the United States
Claim Type	Insurance claims related to car incidents
Dataset Format	Structured / tabular data

Incident Timeline	01.01.2015 - 01.03.2015
Number of Records	1,000 insurance claims
Missing values	1,091 missing entries ("_c39" empty column, will be deleted)
Features	40 total features - 18 quantitative, 20 categorical, 2 datetime
Target Variable	"fraud_reported"
Target Distribution	24,7 % fraudulent 75,3% non-fraudulent

Table 1: Technical overview of the dataset

In addition to the general dataset characteristics shown above, the available features can be further categorized into logical clusters based on their relevance to the insurance claim process.

The table below provides an excerpt from the dataset, illustrating how features are organized into key domains such as insurer information, vehicle details, incident data, and claim-specific attributes.





 Insurer Information <ul style="list-style-type: none"> • age • insured_sex • insured_hobbies 	 Vehicle Information <ul style="list-style-type: none"> • auto_make • auto_model • auto_year
 Incident Details <ul style="list-style-type: none"> • incident_date • incident_location • incident_type • authorities_contacted 	 Claim Details <ul style="list-style-type: none"> • total_claim_amount • vehicle_claim • injury_claim • property_claim

Table 2: Features organisation

Based on the available data and the already identified binary target variable, we will approach this problem using **supervised learning methods**.

Relevance

The dataset contains a variety of data types and a decent structure for analysis. It includes categorical, numerical, and time-based features. Some challenges exist, but the data quality overall allows for meaningful exploration.

	count	mean	std	min	25%	50%	75%	max		
months_as_customer	1000.0	203.95	115.11	0.00	115.75	199.5	276.25	479.00	policy_bind_date	951
age	1000.0	38.95	9.14	19.00	32.00	38.0	44.00	64.00	policy_state	3
policy_deductible	1000.0	1136.00	611.86	500.00	500.00	1000.0	2000.00	2000.00	policy_csl	3
policy_annual_premium	1000.0	1256.41	244.17	433.33	1089.61	1257.2	1415.70	2047.59	insured_sex	2
umbrella_limit	1000.0	1101000.00	2297406.60	-1000000.00	0.00	0.0	0.00	10000000.00	insured_education_level	7
insured_zip	1000.0	501214.49	71701.61	430104.00	448404.50	466445.5	603251.00	620962.00	insured_occupation	14
capital-gains	1000.0	25126.10	27872.19	0.00	0.00	0.0	51025.00	100500.00	insured_hobbies	20
capital-loss	1000.0	-26793.70	28104.10	-111100.00	-51500.00	-23250.0	0.00	0.00	insured_relationship	6
incident_hour_of_the_day	1000.0	11.64	6.95	0.00	6.00	12.0	17.00	23.00	incident_date	60
number_of_vehicles_involved	1000.0	1.84	1.02	1.00	1.00	1.0	3.00	4.00	incident_type	4
bodily_injuries	1000.0	0.99	0.82	0.00	0.00	1.0	2.00	2.00	collision_type	4
witnesses	1000.0	1.49	1.11	0.00	1.00	1.0	2.00	3.00	incident_severity	4
total_claim_amount	1000.0	52761.94	26401.53	100.00	41812.50	58055.0	70592.50	114920.00	authorities_contacted	4
injury_claim	1000.0	7433.42	4880.95	0.00	4295.00	6775.0	11305.00	21450.00	incident_state	7
property_claim	1000.0	7399.57	4824.73	0.00	4445.00	6750.0	10885.00	23670.00	incident_city	7
vehicle_claim	1000.0	37928.95	18886.25	70.00	30292.50	42100.0	50822.50	79560.00	incident_location	1000
auto_year	1000.0	2005.10	6.02	1995.00	2000.00	2005.0	2010.00	2015.00	property_damage	3
									police_report_available	3
									auto_make	14
									auto_model	39
									fraud_reported	2
									dtype: int64	

Figure 1: Features Statistical summary

The primary objective of this analysis is to determine whether a new insurance claim is potentially fraudulent or not. **The target variable is therefore binary:** indicating whether a given claim is classified as fraud or non-fraud. This prediction is based on a combination of information derived from the insurer, the reported incident, and the details of the insured policy.

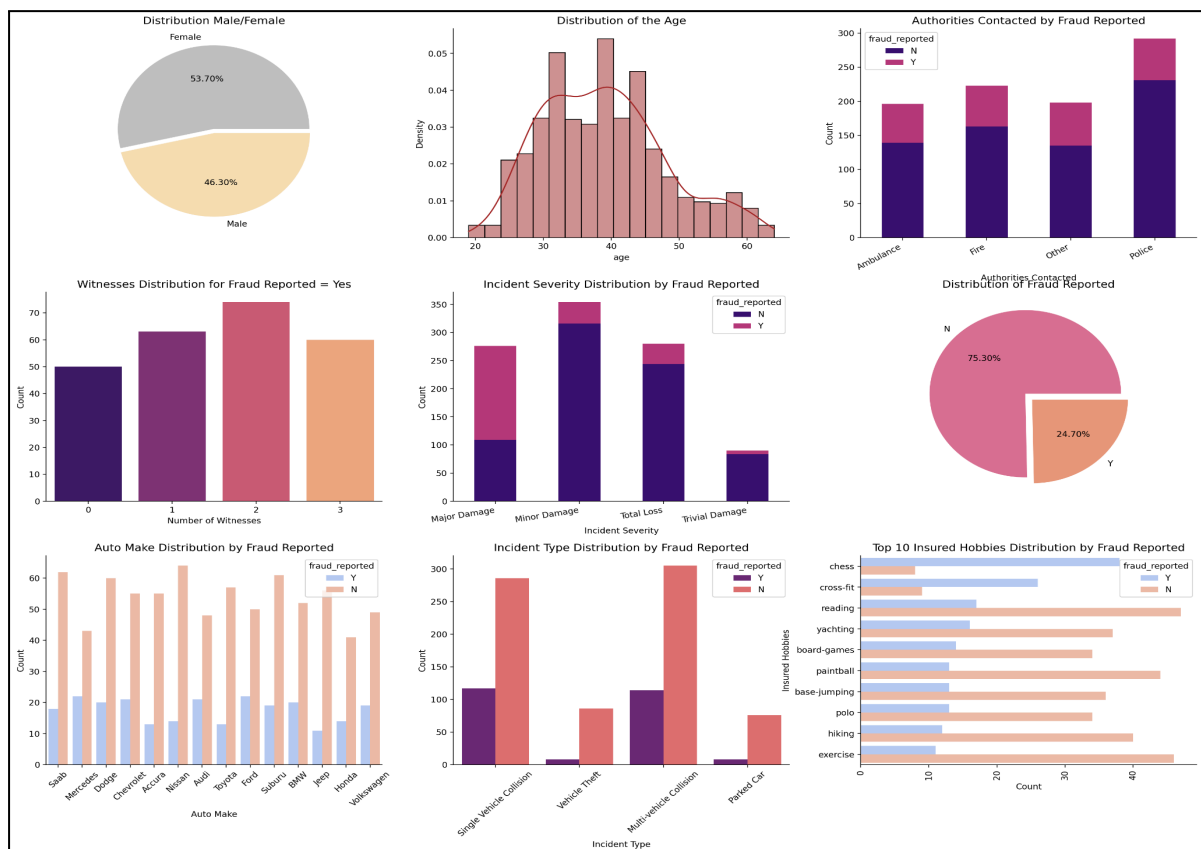


Figure 2: Target variable relationship to some features

Initial analysis and domain knowledge indicate several features that are particularly relevant to predicting fraudulent behaviour. Some are statistically significant; others are informative based on insurance domain insights. The most relevant features seem to be:

- Certain **hobbies** (e.g., chess, crossfit) are highly associated with fraud.
- **Incident severity** and **type** show clear class separation for fraud cases.
- The feature "vehicle_claim" showed a very **high F-score (29.72)** and **low p-value (6.29e-08)** in SelectKBest, confirming statistical relevance.

Subsequent sections of this report will provide a detailed explanation of the analysis results.

While the dataset is suitable for initial modelling, it comes with several limitations that may affect prediction generalizability. These should be considered when interpreting model results.

- The **time range** is short (only 2 months), limiting detection of seasonal effects.
- The **class distribution** is imbalanced, posing challenges for model training.
- **Bias is likely** present due to internal insurance data collection processes (e.g., major vs. minor damage follow-ups).

Pre-processing and feature engineering

Prior to modelling, several data cleaning and preprocessing steps were necessary to ensure data quality and reduce the risk of bias or data leakage.

Handling NaN – values

One key issue in the dataset was the presence of missing values. After deleting the empty feature "_c39", approximately 9.1% of the NaN-values in the column "authorities_contacted" needed to be handled. We do not want to remove these values, as they represent almost 10% of the data and our overall dataset is relatively small. Deleting them could lead to a loss of important information. Instead, we can apply different strategies to handle the missing (NaN) values.

	fraud	0	1
authorities_contacted			
Ambulance	0.709	0.291	
Fire	0.731	0.269	
Other	0.682	0.318	
Police	0.791	0.209	
NaN	0.934	0.066	

The distribution of fraud and non-fraud cases shows that the missing value is not necessarily an indicator of fraud. Therefore, we analysed the relationship between these missing values and other features to better understand any potential patterns.

For example, when looking at the crosstab between "authorities_contacted" and "incident_type", it becomes clear that missing values only appear in cases of "Parked Car" and "Vehicle Theft". In both situations, it is evident that the police were contacted.

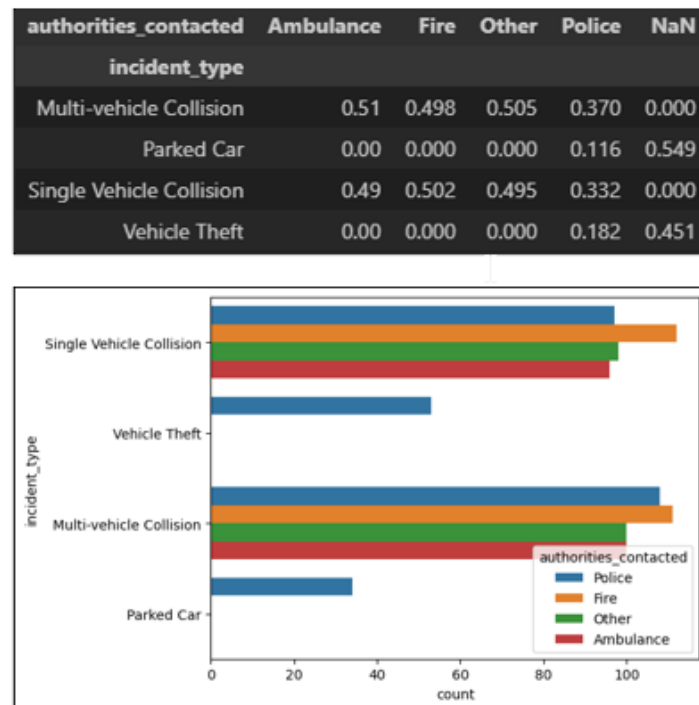


Figure 4: incident_type in relation to authorities_contacted

Since "Police" is also the most frequently occurring category, we decided to use it as a replacement value during encoding. This will be done using the SimpleImputer() method as part of the preprocessing pipeline.

Handling '?' – values

In addition to missing values, we also encountered placeholder values represented as "?" in three columns. Before proceeding with the next stage of analysis, it is necessary to address the presence of the "?" placeholder found in three columns. This character may either be treated as a missing value and replaced accordingly, or considered as a separate category. In addition, it is important to examine whether the occurrence of "?" shows any systematic relationship with fraudulent activity.

"police_report_available"

	fraud	0	1
police_report_available			
?		0.741	0.259
NO		0.749	0.251
YES		0.771	0.229

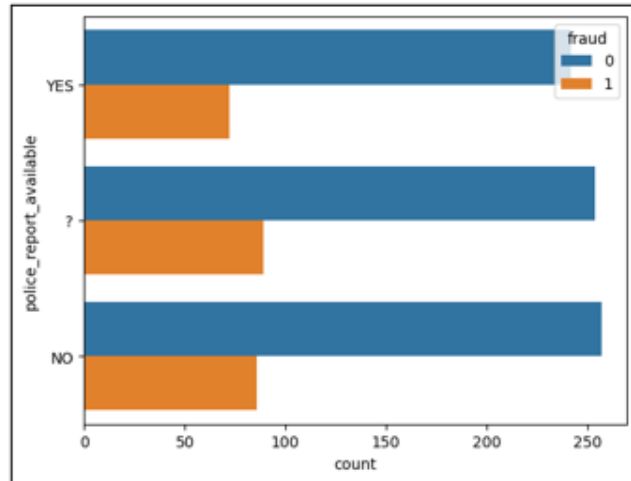


Figure 5: police_report_available in relation to fraud_reported

"property_damage"

	fraud	0	1
property_damage			
?		0.714	0.286
NO		0.805	0.195
YES		0.742	0.258

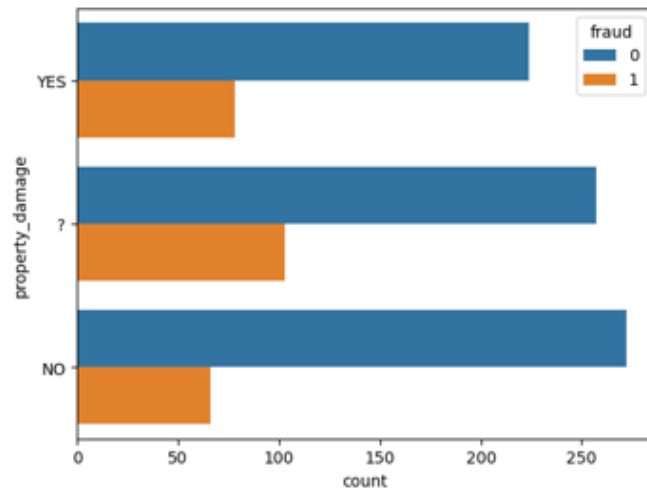


Figure 6: property_damaged in relation to fraud_reported

In "police_report_available" and "property_damage", these ambiguous values may indicate potential fraud, as the information could have been intentionally withheld. To avoid any misleading interpretations or data leakage, these entries were not imputed with common values from the distribution, but instead were explicitly replaced with a new category called "Unknown".

"collision_type"

collision_type	?	Front Collision	Rear Collision	Side Collision
incident_type				
Multi-vehicle Collision	0.000	0.453	0.521	0.551
Parked Car	0.472	0.000	0.000	0.000
Single Vehicle Collision	0.000	0.547	0.479	0.449
Vehicle Theft	0.528	0.000	0.000	0.000

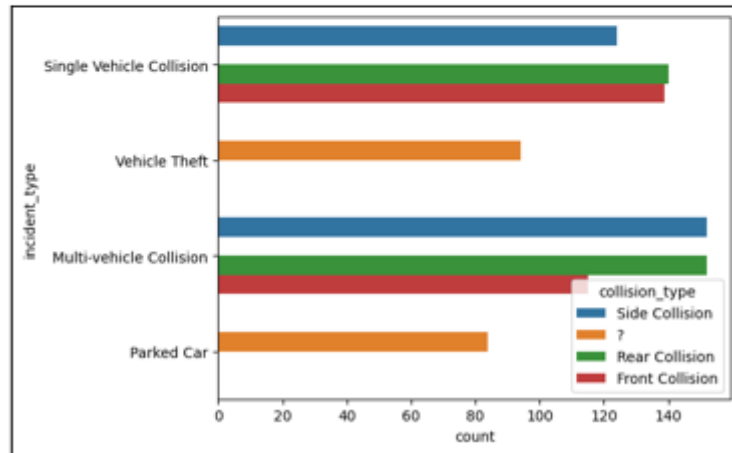


Figure 7: incident_type in relation to collision_type

The "?"-values in the "collision_type" column were observed only in relation to the incident types "Vehicle Theft" and "Parked Car". Given the context, this can be considered as a separate category. Therefore, the "?" entries were replaced with "No Collision" to accurately reflect the nature of these incidents.

Visualizations and Statistics

To better understand the relationships between the variables and to guide the modeling process, we conducted a series of exploratory data analyses, statistical tests, and visual inspections.

Quantitative features

We examined the distribution of all numeric features using distribution plots and KDE curves. We found that only the feature "policy_annual_premium" showed approximate normal distributions.

All other numerical features did not follow a normal distribution and would be therefore scaled using Min-Max-Normalization, to preserve their original shape while bringing them to a comparable range for model input.

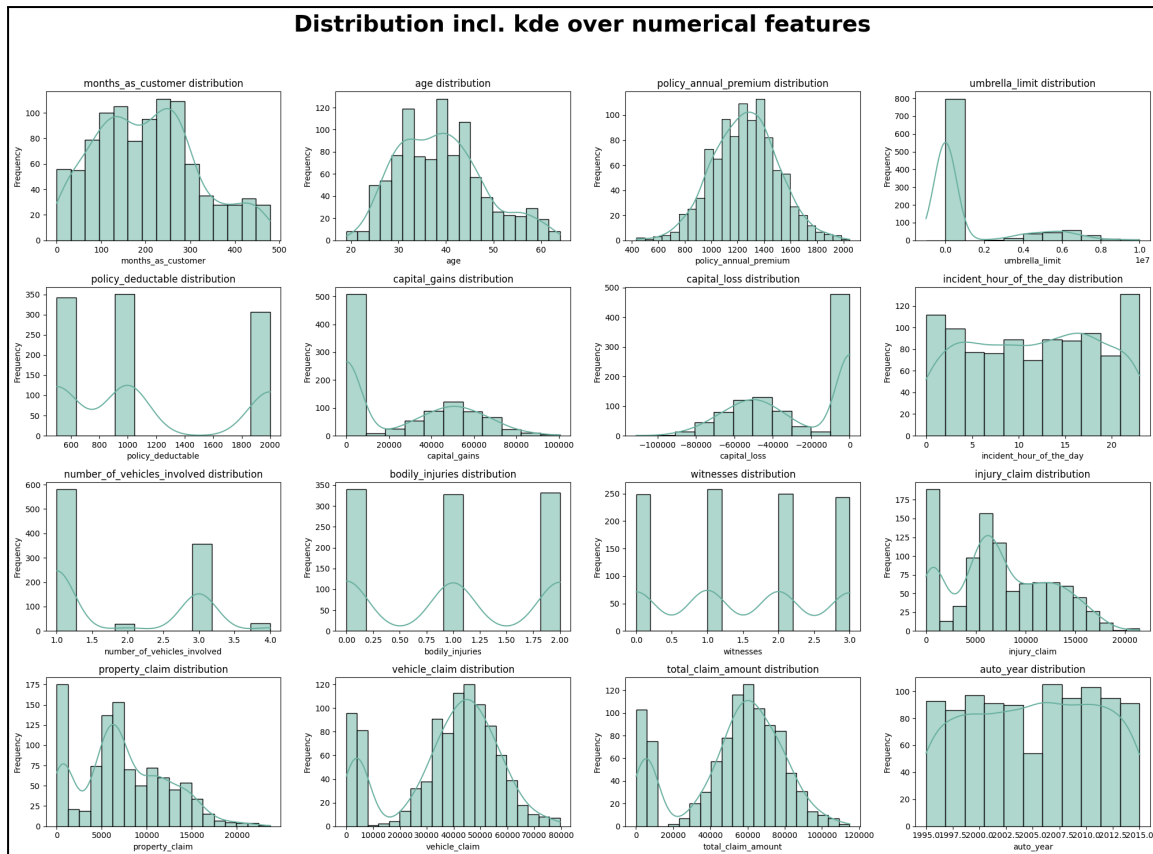


Figure 8: Distribution overview of quantitative variables

To analyse the relationships between the features, we used both a **heatmap and statistical tests**. The heatmap helped us to get a quick visual overview of correlations between variables. In addition, we performed statistical tests to evaluate the **relevance of each feature in relation to the target variable (fraud)**. This combination allowed us to identify not only strong correlations between features but also which variables are most useful for predicting fraud.

Heatmap

On the right-hand side, we present a heatmap displaying all quantitative features. In this visualization, lighter colors indicate strong correlations, while darker shades represent weaker or no correlation. Two distinct clusters of high correlation are clearly visible. The zoomed-in section on the right highlights the larger of the two clusters.

Within this area, we observe strong correlations between different types of claims—particularly between "vehicle_claim" and "total_claim_amount". To **prevent multicollinearity** in the modelling process, it is necessary to perform feature selection and retain only one of these variables for machine learning purposes.

The second cluster, located in the opposite corner of the heatmap, reveals a similarly strong

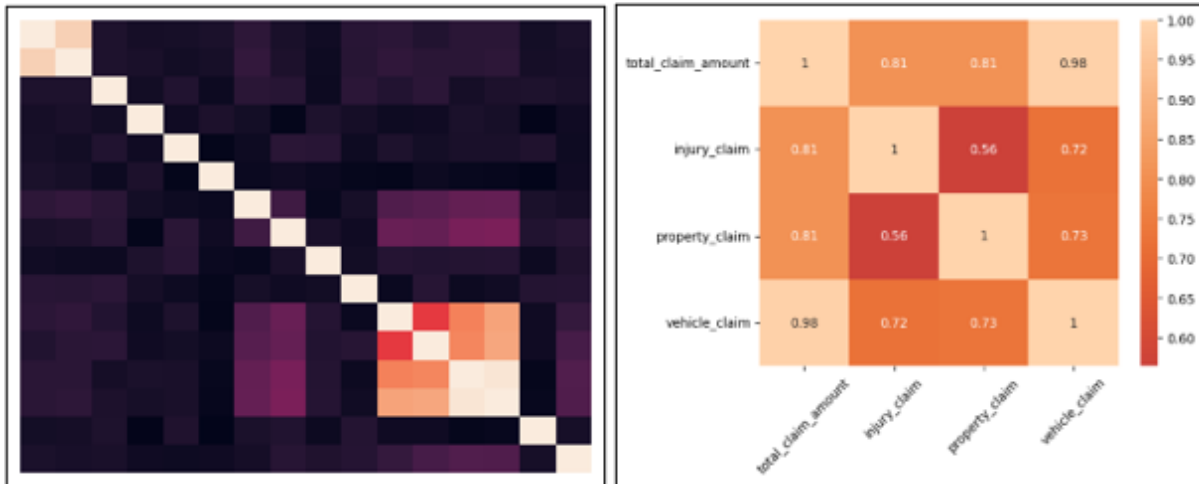


Figure 9: Heatmap of quantitative variables

correlation between "months_as_customer" and "age". As with the claim-related features, this relationship also calls for careful consideration to avoid redundancy and ensure model stability.

Statistical Test - Quantitative Features

When it comes to statistical testing of quantitative features, there are different types of tests that can be used. It is important to note that not all tests focus on linear relationships only. In our case, we tried different approaches to better understand the data and its connection to the target variable. Below, we present the most relevant results from these tests.

	feature	pearson_p_value	spearman_p_value
12	total_claim_amount	1.952936e-07	0.000010
13	injury_claim	3.986262e-03	0.005440
14	property_claim	1.218941e-05	0.000023
15	vehicle_claim	6.297261e-08	0.000005

Figure 10: Statistically most relevant quantitative features

The results showed that all claim-related features had **p-values below 0.05**, indicating a **statistically significant relationship** with the target variable. In contrast, other features showed p-values higher than 0.05 in different statistical tests, such as the t-test and the Mann-Whitney U test, suggesting that they are not statistically significant for the target values.

Additionally, SelectKBest was applied to the quantitative features, and the results are summarized below.

	Feature	F-Score	p-Value
14	vehicle_claim	29.718214	6.297261e-08
11	total_claim_amount	27.463776	1.952936e-07
13	property_claim	19.327729	1.218941e-05
12	injury_claim	8.328776	3.986262e-03
8	number_of_vehicles_involved	2.689100	1.013520e-01
10	witnesses	2.451026	1.177651e-01
9	bodily_injuries	1.146656	2.845093e-01
0	months_as_customer	0.421370	5.164037e-01
4	insured_zip	0.374497	5.407027e-01
5	capital_gains	0.366990	5.447871e-01
6	capital_loss	0.220519	6.387475e-01
2	policy_deductable	0.219163	6.397808e-01
3	policy_annual_premium	0.209284	6.474290e-01
1	age	0.147188	7.013189e-01
15	auto_year	0.062735	8.022760e-01
7	incident_hour_of_the_day	0.018590	8.915769e-01

Figure 11: Kbest identified quantitative variables

Based on these results, together with insights from the heatmap analysis, the feature "vehicle_claim" was selected for use in the machine learning model, while the other claim-related features were excluded to avoid multicollinearity.

Qualitative Features

For the categorical features, we used the Chi²-test. Similar to the approach used for numerical features, this test helps us evaluate the statistical significance of each feature in relation to the target variable. We used a threshold of $p < 0.05$ to consider a feature statistically significant.

In addition to significance, we also assessed the strength of the association by calculating Cramér's V. This metric gives us an idea of how strongly a categorical feature is related to the target:

- $\approx 0.1 \rightarrow$ weak
- $\approx 0.3 \rightarrow$ moderate
- $\geq 0.5 \rightarrow$ strong

This allowed us to evaluate both the relevance and the impact of categorical features before selecting them for modelling. The following table displays the relevant results:

	column	p	v_cramers
0	incident_severity	5.447034e-57	0.514040
1	insured_hobbies	8.989147e-25	0.402884
2	collision_type	7.118898e-07	0.177104
3	incident_type	2.101334e-06	0.170680
4	incident_state	1.307713e-02	0.127001
5	property_damage	1.803296e-02	0.089616
6	authorities_contacted	4.057387e-02	0.095438

Figure 12: Statistical significance and strength of association of categorical variables to target variable

As we can see from the table, the first seven features are statistically significant. "incident_severity" shows a strong association with the target variable. On the other hand, while "authorities_contacted" is also statistically significant, the strength of its association is weak according to the Cramér's V value.

The plots below show the distribution of fraud in several categorical features.

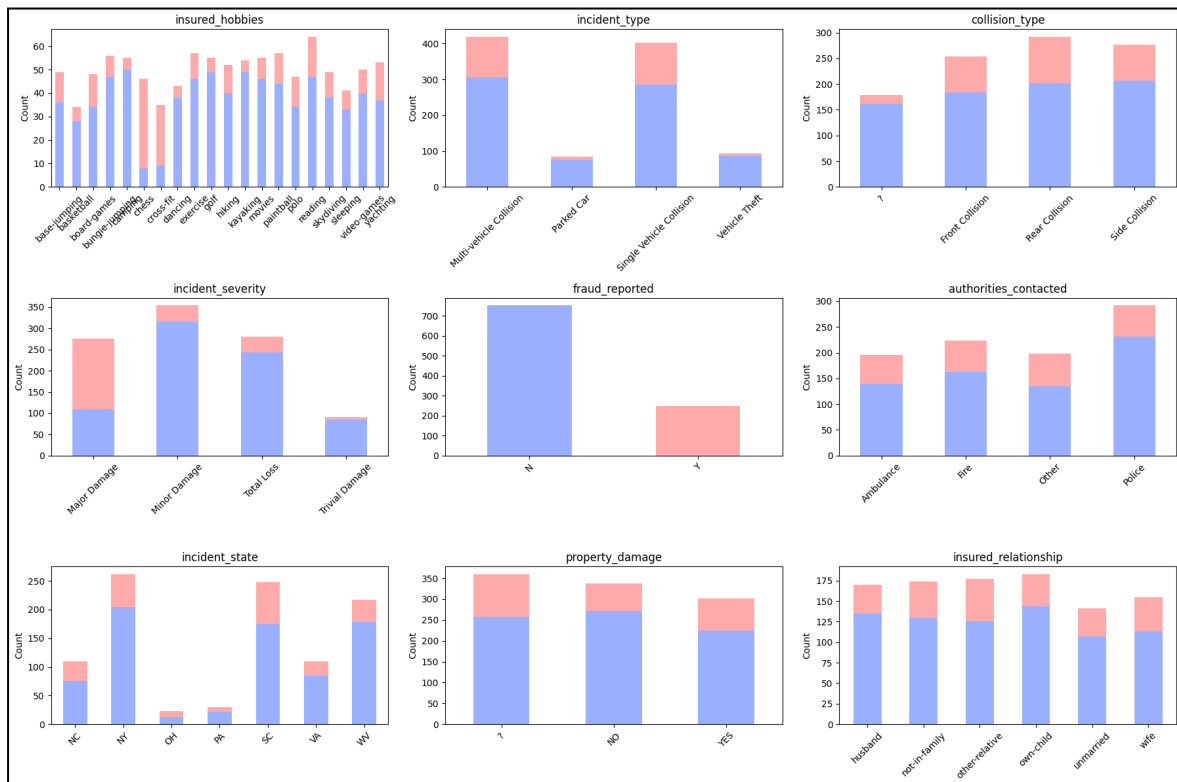


Figure 13: Distribution of fraud in several categorical variables

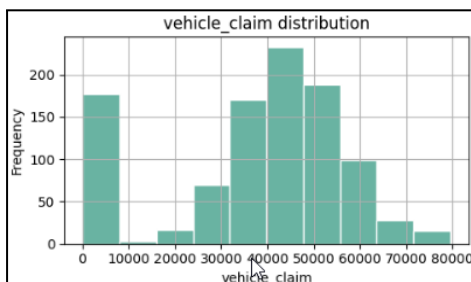
Feature reduction

Based on these analyses and as a first approach we decided to reduce our features by focusing on the p-value of the statistical tests. Meaning, we only kept the features with $p < 5\%$. This reduced our number of features from forty to seven categorical features, one quantitative feature (vehicle_claim) and one binary target variable (fraud_reported).

Encoding

Before we can start building the first models, we need to make sure that all variables are ready for machine learning.

For the quantitative feature, we need to consider the distribution of the feature. When the scale of the values of the explanatory variables differs, this can lead to algorithm non-convergence or to poor parameter estimation. All algorithms that involve a notion of distance, such as logistic regression, SVMs or KNN are very sensitive to the scale of the variables. Other algorithms based for example on DecisionTrees are less sensitive.



Since we are going to experiment with various models to search for the best that fits the data, we choose to scale the 'vehicle_claim'. It is not normally distributed. Therefore, we will apply the MinMaxScaler for normalization.

Categorical variables must also be transformed, since machine learning algorithms cannot process string values directly. Here, it is important to decide whether the variable is nominal or ordinal, so that we can choose the appropriate encoding method.

Feature	Nominal / Ordinal	Encoder
incident_severity	nominal / ordinal	OneHotEncoder / OrdinalEncoder
insured_hobbies	nominal	OneHotEncoder
collision_type	nominal / ordinal	OneHotEncoder / OrdinalEncoder
incident_type	nominal / ordinal	OneHotEncoder / OrdinalEncoder
incident_state	nominal	OneHotEncoder
property_damage	nominal	OneHotEncoder / OrdinalEncoder
authorities_contactd	nominal	OneHotEncoder

Table 3: Classification of qualitative features for the choice of the encoding method

In the table, certain features are marked as both *nominal* and *ordinal* because they could be interpreted differently depending on the modelling approach.

For example, **"incident_severity"** often includes levels such as *Minor*, *Major*, and *Total Loss*, which clearly follow a logical order in terms of severity. This makes the feature ordinal. However, if the categories are not used consistently or do not follow a clear ranking, it could also be treated as nominal.

"collision_type" and **"incident_type"** may also seem ordinal if you assume there is an increasing level of damage or risk associated with the categories. But if the categories simply describe different types of incidents without a natural order, they are considered nominal.

For simplicity, we initially decided to apply the OneHotEncoder to all categorical features. OneHotEncoding offers several advantages, especially when working with nominal categorical features. It is easy to implement and supported by most machine learning frameworks. One of the key benefits is that it prevents the model from assuming a false order between categories, since each category is represented as a separate binary column. This makes the encoded data suitable for many algorithms that require numerical input.

For completeness, we would like to mention that we also checked for outliers during the data preprocessing phase. We used boxplots to visualize the distributions. However, since only one quantitative feature ("vehicle_claim") remained after feature selection, we did not focus on outlier analysis in detail. The boxplot for this feature did not show any significant outliers.

As for dimensionality reduction techniques, we decided not to apply methods such as PCA at this stage. There are two main reasons for this decision:

- The dataset is relatively small, and using dimensionality reduction could increase the risk of overfitting.
- Second, after some initial model testing, we briefly considered PCA, but it did not improve the performance or interpretability of our models. Therefore, we chose to proceed without PCA.

Summary

This first report focused on exploring, cleaning, and preparing the dataset for the machine learning phase. We analysed 1,000 anonymized insurance claims and identified key patterns in both numerical and categorical features. Through a combination of visualizations and statistical tests, we found that a few selected features, such as "vehicle_claim" and "incident_severity", show strong relevance to fraud prediction.

Missing values and placeholder entries were handled carefully to avoid data loss and potential leakage. OneHotEncoding was used for categorical variables, and MinMaxScaler was chosen for numerical data due to the lack of normal distribution. Outliers were checked, and dimensionality reduction techniques were considered but not applied, as they did not improve performance in our initial tests.

Since we are dealing with a **supervised learning problem**—where the target variable (fraud or not fraud) is known—we are now well-positioned to apply and evaluate appropriate machine learning algorithms.

The dataset has now been cleaned, reduced, and encoded, making it ready for model building. In the upcoming second report, we will focus on modelling and solving the fraud detection problem using supervised learning techniques. We will train, evaluate, and compare different algorithms to find the most effective solution for this real-world challenge.

Report 2: Modelling and Results using ML

Introduction

The objective of this project is to develop machine learning models capable of accurately predicting fraudulent insurance claims with a high degree of confidence.

As outlined in the previous chapter, the data preprocessing phase included several essential steps to ensure data quality and prepare the dataset for modelling. The process began with **checking for duplicated records**, which were not present in the dataset.

Handling missing values was also a critical part of the pipeline. Specifically, missing values in the "authorities_contacted" variable were addressed **after** splitting the data into training and testing sets, in order to avoid data leakage.

To improve the model's performance and interpretability, the **cardinality of certain features was reduced** based on prior statistical tests. These tests helped identify eight variables—seven categorical and one numerical—with the strongest influence on the target variable. Highly correlated features such as "total_claim_amount", "injury_claim" and "property_claim" were removed in favour of a single, more representative variable: "vehicle_claim".

The **target variable**, "fraud_reported", was clearly defined as part of the task and, given the **binary nature of the classification problem**, encoded using a LabelEncoder. To prevent information from leaking between training and test data, the **dataset was split** early in the process into training (80%) and testing (20%) subsets.

Since most machine learning algorithms require numerical input, careful attention was paid to **encoding** and **scaling the features**. All categorical variables were transformed using one-hot encoding, ensuring each category was treated distinctly without introducing ordinal relationships. For the numerical variable "vehicle_claim", normalization was applied using MinMaxScaler, as its distribution was not normally distributed.

Defining a suitable evaluation approach for the model

Before proceeding with model development, it is important to define the evaluation metric that will be used to assess model performance. Selecting an appropriate metric is a crucial step, as it directly influences how we interpret the effectiveness of each model and guides the selection of the final solution.

Fraud cases that are not falsely classified as non-fraud (False Negative) by the model can cause a big financial damage to the company by settling this claim. It is therefore important to minimize the false negative rate.

On the other hand, non-fraud cases that are classified as fraud (False Positive) are to be minimized. Allocating resources to investigate cases that are not actually fraudulent should also be avoided, as it leads to unnecessary operational costs and inefficiencies. Additionally,

wrongly assigning penalties due to a misprediction may result in costly lawsuits against the company. For that reason, a high precision in the model prediction is also important.

In order to combine those two metrics and by considering the class imbalance of the target variable, the weighted F1-score will be used as the base metric to evaluate the model performance.

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

In the following sections, we will describe the process and results of the modelling workflow in detail. The overall approach consisted of several key steps designed to build, evaluate, and refine the machine learning models. These steps included:

- **Initial model testing** using simple algorithms on the imbalanced dataset
- **Resampling strategy selection** to address class imbalance
- **Model re-evaluation** with resampled data
- **Hyperparameter tuning**
- **Model robustness testing** via cross-validation

Each of these stages played an essential role in developing a reliable model for fraud detection.

Modelling: First attempt with the imbalanced data set

A variety of classifiers were trained on the imbalanced data set, including **linear models** (Linear Discriminant Analysis, SVC, Logistic Regression, Ridge Classifier), **tree-based models** (Decision Tree, Random Forest), and **ensemble methods** (XGBoost, AdaBoost). Model performance was measured with the weighted F1-score:

	Model	Train F1 (weighted)	Test F1 (weighted)	F1 (weighted) Difference
0	Logistic Regression	0.88	0.83	0.05
1	SVC	0.90	0.80	0.10
2	Ridge Classifier	0.88	0.85	0.03
3	Linear Discriminant Analysis	0.88	0.84	0.03
4	Decision Tree	1.00	0.78	0.22
5	Random Forest	1.00	0.81	0.19
6	AdaBoost	0.84	0.78	0.05
7	XGBoost	1.00	0.81	0.19

Figure 15: Performance of selected models on the imbalanced data set

The tree-based algorithms—Decision Tree, Random Forest, and XGBoost—achieved perfect performance on the training data (weighted F1 = 1.00). This result is likely a consequence of the small training-set size, which makes it easy for these flexible learners to memorise the samples. The pronounced gap between training and test scores confirms that the models are overfitting.

Conversely, algorithms that rely on distance measures or assume a degree of linear separability—such as Logistic Regression and the Support Vector Classifier—obtained respectable weighted F1-scores on the training data but did not overfit to the same extent.

The confusion matrices for the top-performing models highlight how class imbalance shapes their behaviour. Both the Ridge Classifier (weighted F1 = 0.85) and Linear Discriminant Analysis (weighted F1 = 0.84) achieve their scores chiefly by labelling the great majority of samples as **non-fraud**. Because the training data are dominated by legitimate transactions, these models learn decision boundaries that strongly favour the majority class, yielding many true negatives but comparatively few true positives.

Compared with Logistic Regression and the Support Vector Classifier—which post lower weighted F1-scores—the Ridge Classifier and LDA correctly recognise an even larger share of non-fraud cases, illustrating their pronounced bias toward the majority class.

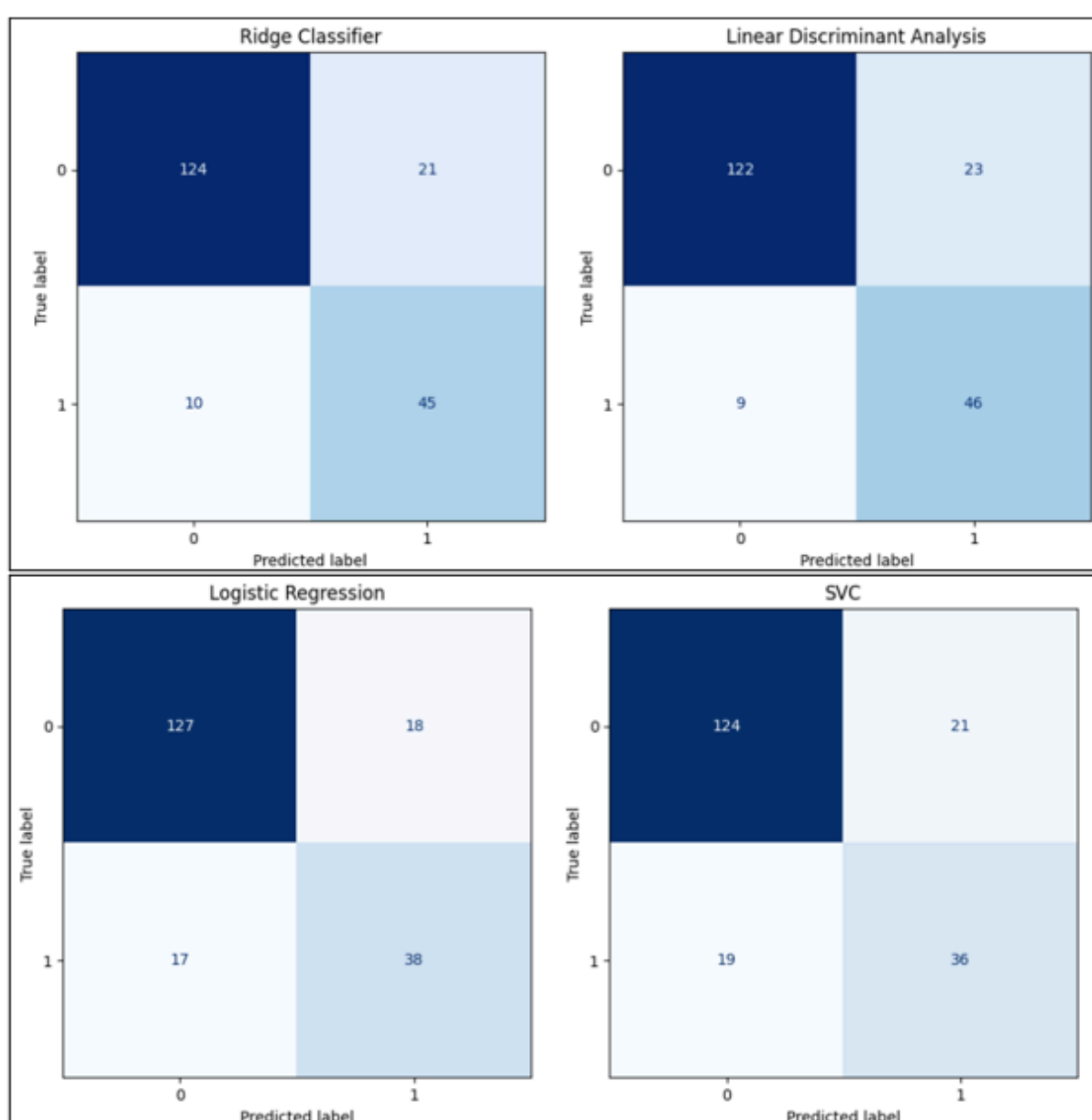


Figure 16: Confusion matrix of selected models performance on the imbalanced data set

The initial results were satisfactory. Our next step was to examine whether—and in what ways—the scores could be further improved.

Feature Engineering

At this stage, we reviewed the encoding and normalization steps to find ways to improve the weighted F1-score. It was quickly noticed that the variable **"incident_severity"** has a natural order, from *Trivial Damage* and *Minor Damage* up to *Major Damage* and *Total Loss*. Based on this, we assumed that treating **"incident_severity"** as an ordinal variable, instead of a nominal one, could lead to better results.

To test this idea, we encoded **"incident_severity"** with numerical values reflecting the order and re-trained all models. However, the results showed a clear drop in performance.

	Model	Train F1 (weighted)	Test F1 (weighted)	F1 (weighted) Difference
0	Logistic Regression	0.77	0.73	0.03
1	SVC	0.81	0.72	0.08
2	Ridge Classifier	0.77	0.73	0.04
3	Linear Discriminant Analysis	0.77	0.75	0.02
4	Decision Tree	1.00	0.79	0.21
5	Random Forest	1.00	0.76	0.24
6	AdaBoost	0.82	0.78	0.04
7	XGBoost	1.00	0.80	0.20

Figure 17: Performance of selected models with *incident_severity* encoded as ordinal on the imbalanced data set

For example, the weighted F1-score for the Ridge Classifier fell from 0.85 to 0.73, and for Linear Discriminant Analysis from 0.84 to 0.75. A similar decrease was observed for Logistic Regression and SVC. The table below summarises the results.

Model	Test F1 (weighted) "incident_severity" as nominal	Test F1 (weighted) "incident_severity" as ordinal
Logistic Regression	0.83	0.73 ↓
SVC	0.80	0.72 ↓
Ridge Classifier	0.85	0.73 ↓
Linear Discriminant Analysis	0.84	0.75 ↓
Decision Tree	0.78	0.79 →
Random Forest	0.81	0.76 ↓

AdaBoost	0.78	0.78 →
XGBoost	0.81	0.80 →

Table 4: Performance comparison of models between *incident_severity* encoded as nominal or ordinal variable

This drop can be explained by the fact that encoding with numbers assumes that the difference between categories, for example between *Trivial Damage* and *Minor Damage*, is the same as between *Major Damage* and *Total Loss*. In reality, these differences may not be equal, and forcing a linear relationship that does not exist can mislead the models.

Based on these findings, we decided to continue using the original approach of treating "**incident_severity**" as a nominal variable and encoding it with OneHotEncoder. In the next step, we will focus on balancing the dataset to further improve the model performance.

Addressing the class Imbalance

A closer examination of the target variable distribution revealed that the dataset is imbalanced. Approximately 76% of the observations are labelled as non-fraud cases, while only 24% are fraud cases.

```
fraud_reported
0    0.76
1    0.24
Name: proportion, dtype: float64
```

Figure 18: Class imbalance of the target variable

To address this imbalance, we tested different resampling strategies, including undersampling, oversampling, and the SMOTETomek method. However, the results obtained from undersampling were not satisfactory, likely due to the already limited number of samples in the dataset.

Given the small dataset size, the best approach was to apply resampling techniques that increase the number of samples in the minority class. For this purpose, we implemented both Random OverSampling (ROS) and the more advanced SMOTETomek method. SMOTETomek is a hybrid technique that combines oversampling and undersampling: it first generates synthetic samples for the minority class and then removes noisy examples from the majority class. This process clarifies the class boundaries, helping models better distinguish between fraud and non-fraud cases.

The rebalancing on the training set is illustrated in the following screenshot.

	Initial train set	ROS	SMOTETomek
Fraud cases	192	608	605
Non-Fraud cases	608	608	605

Table 5: Rebalancing of the training data set

The resampled data set was then used to train and evaluate the selected models.

Modelling with the resampled data set

Modelling with the RandomOverSampler (ROS) data

In order to evaluate and compare the predictions on the resampled training data, we selected the same models that were used on the initial training set.

The overall trend of the results remains similar to the initial training set. Tree-based models improved slightly, with test scores increasing by about 0.01. However, their training scores were again perfect, which indicates overfitting. The resampling did not solve the overfitting problem for the tree-based models.

Some of the linear models, on the other hand, showed noticeable improvements. In particular, AdaBoost, Logistic Regression, and SVC achieved better results. Their weighted F1-scores increased by more than 0.02, and the gap between the training and test scores was reduced. The results are presented below.

	Model	Train F1 (weighted)	Test F1 (weighted)	F1 (weighted) Difference
0	Logistic Regression	0.87	0.85	0.02
1	SVC	0.90	0.84	0.05
2	Ridge Classifier	0.87	0.84	0.03
3	Linear Discriminant Analysis	0.87	0.84	0.03
4	Decision Tree	1.00	0.77	0.23
5	Random Forest	1.00	0.82	0.18
6	AdaBoost	0.87	0.84	0.03
7	XGBoost	1.00	0.80	0.20

Figure 19: Models performance with the RandomOversampler applied

When comparing the results above with the new ones, we can see that the performance of Linear Discriminant Analysis, Logistic Regression, SVM, and Ridge Classifier is quite similar overall. In this case, Linear Discriminant Analysis and Ridge Classifier show a decrease in their scores, while Logistic Regression and SVM have improved. The confusion matrices provide an overview of the results:

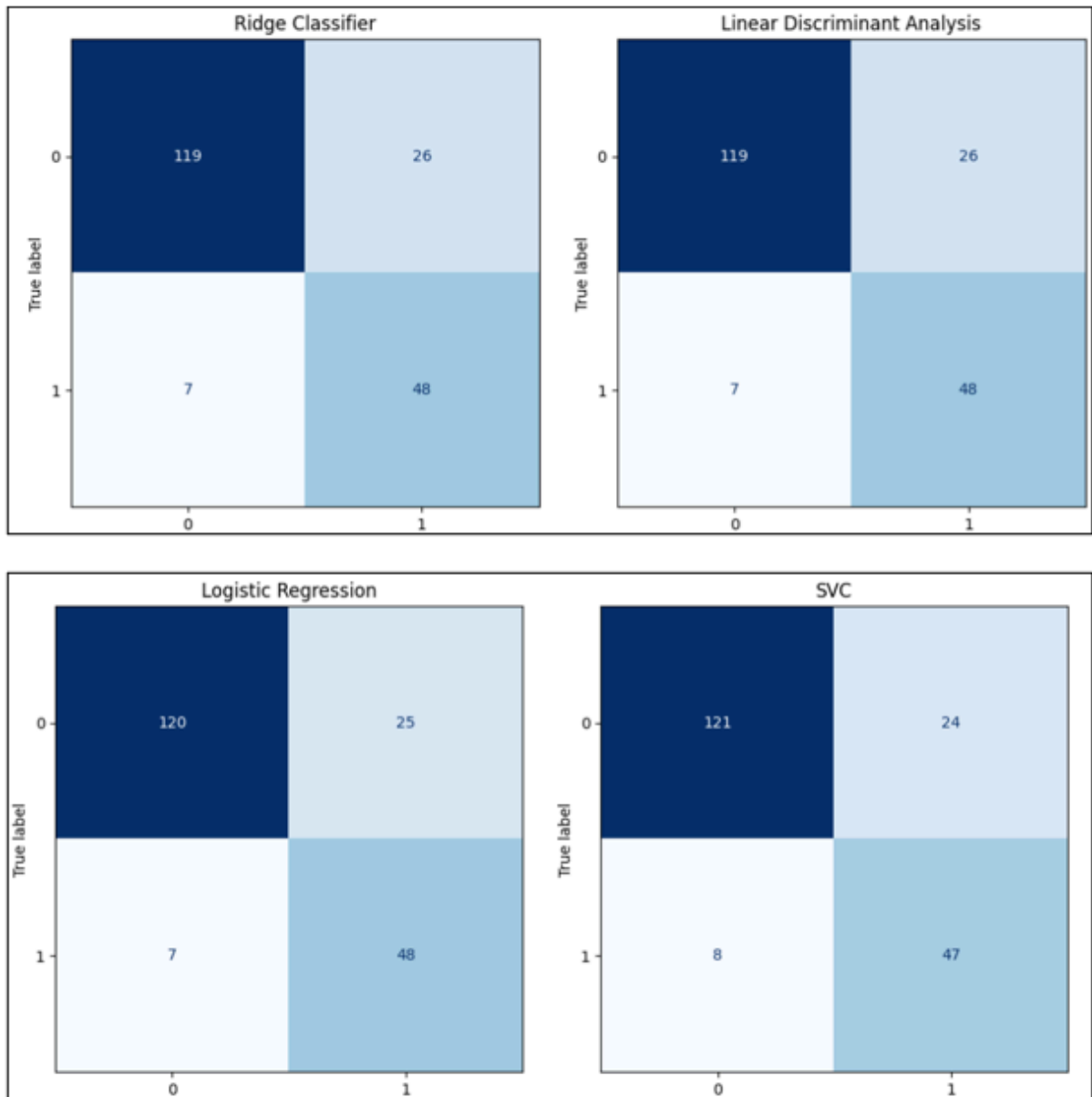


Figure 20. Models' confusion matrix with RandomOversampler applied

Modelling with the SMOTETomek resampled data

After applying SMOTETomek as the resampling technique, the best-performing models were Linear Discriminant Analysis, Ridge Classifier, Support Vector Classifier, and Logistic Regression. These models achieved the highest weighted F1-scores and showed the best balance between training and test performance.

Since SMOTETomek not only generates synthetic minority class examples but also removes noisy majority class instances, it helps the models to better learn the decision boundaries between classes.

	Model	Train F1 (weighted)	Test F1 (weighted)	F1 (weighted) Difference
0	Logistic Regression	0.89	0.84	0.05
1	SVC	0.93	0.83	0.10
2	Ridge Classifier	0.89	0.84	0.05
3	Linear Discriminant Analysis	0.89	0.84	0.05
4	Decision Tree	1.00	0.79	0.21
5	Random Forest	1.00	0.81	0.19
6	AdaBoost	0.90	0.83	0.07
7	XGBoost	1.00	0.82	0.18

Figure 21: Models performance with SMOTETomek applied

The resampling tests with SMOTETomek show similar results once again. We still observe overfitting in the tree-based models, while the other models show improvements in comparison to the initial train set results.

Again, a taste of the best scorer models is illustrated below:

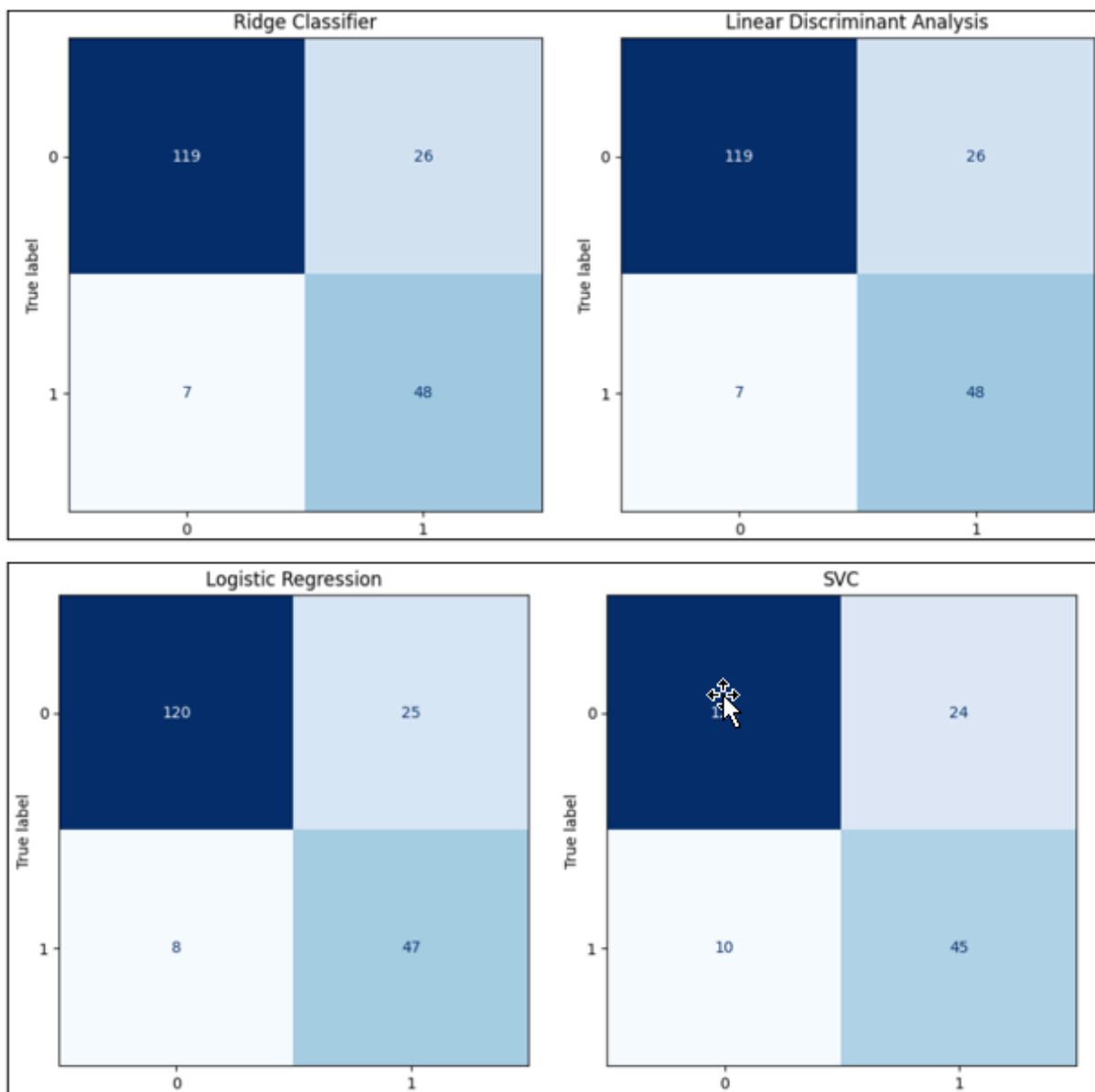


Figure 22: Models confusion matrix with SMOTETomek applied

In summary, Logistic Regression and Support Vector Classifier can be considered as the best scorer models with both resampling techniques as illustrated in the following table.

SMOTETomek resampling is designed to produce a clear boundary between the classes by combining oversampling the minority class and cleaning the majority class.

Based on the results, the decision was made to further optimize the Linear Discriminant Analysis and Ridge Classifier models.

	Best Weighted f1-score (Test set)		
	Initial Train Set	ROS Sampling	SMOTETomek Sampling
Logistic Regression	0.83	0.85	0.84
Support Vector Classifier	0.80	0.84	0.83
Ridge Classifier	0.85	0.84	0.84
Linear Discriminant Analysis	0.84	0.84	0.84

Table 6: Models performance comparison between RandomOversampler and SMOTETomek

Feature importance analysis

Since the Linear Discriminant Analysis with SMOTETomek was one of the best scorer models, performing a feature importance analysis will help determine how much each feature contributes to predicting the target variable. The result of the analysis can be used for model interpretability, and overall model optimization.

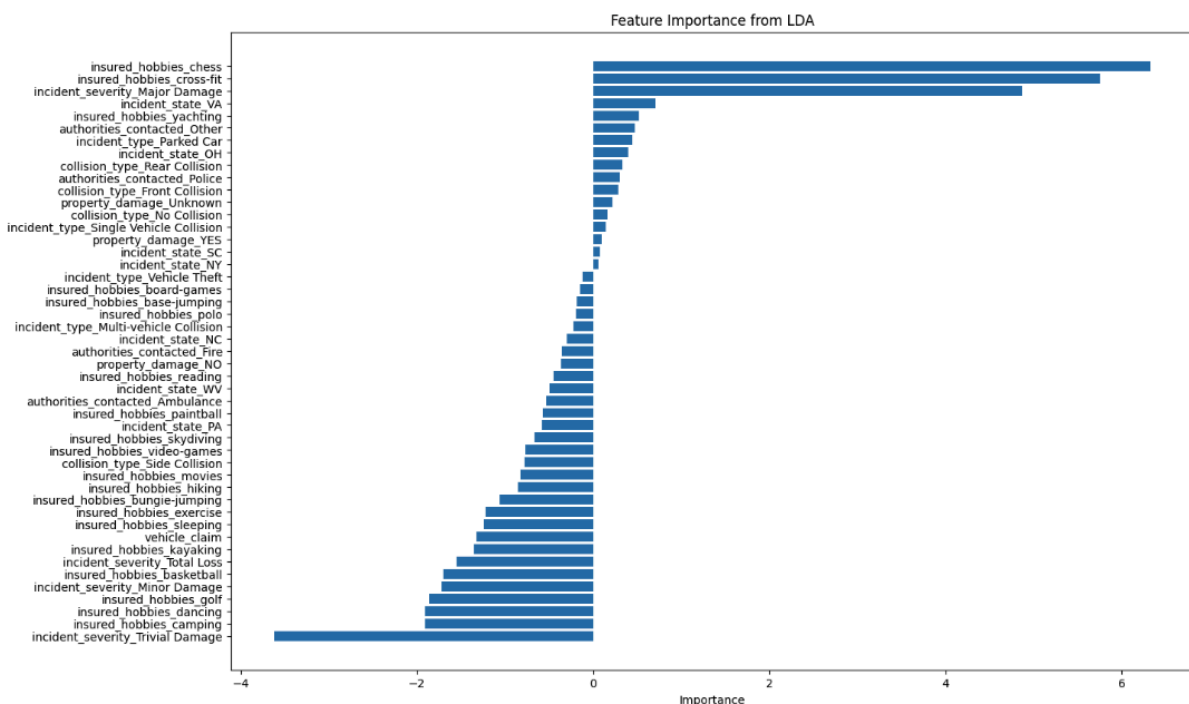


Figure 23: Features importance analysis

The top three features that contribute the most to predict the fraud cases are:

- 1) insured_hobbies_chess
- 2) insured_hobbies_crossfit
- 3) incident_severity_Major_Damage

It seems that by major damage, the likelihood of a fraudulent claim is bigger for a customer, due to its financial impact.

It can also be assumed that, since insurance claims with major damage are more expensive for the companies, those cases are more investigated, increasing the probability of detecting frauds.

Another way to further optimize the best scorer models is by performing a GridSearch.

Hyperparameter optimization with GridSearch

GridSearch performed an exhaustive search on a predefined hyperparameter set in order to find the best model that optimizes the f1-score.

GridSearch was performed on the following best scorer models with the SMOTETomek resampled data set:

- Linear Discriminant Analysis
- Ridge Classifier
- AdaBoost (for comparison *)
- Logistic Regression (for comparison *)

The selected parameters for the exhaustive GridSearch are illustrated in the following screenshot. For each model, they cover a wide range to search through.

```
# Define the parameter grid for each model
param_grid = {
    'Linear Discriminant Analysis': {
        'solver': ['svd', 'lsqr', 'eigen'],
        'shrinkage': [None, 'auto', 0.1, 0.5, 0.9]
    },
    'Ridge Classifier': {
        'alpha': [0.01, 0.1, 1.0, 10.0, 100.0],
        'solver': ['auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga']
    },
    'AdaBoostClassifier': {
        'n_estimators': [25, 50, 100, 200, 300],
        'learning_rate': [0.001, 0.01, 0.1, 1.0, 10.0],
        'algorithm': ['SAMME', 'SAMME.R']
    },
    'LogisticRegression': {
        'penalty': ['l1', 'l2', 'elasticnet', 'none'],
        'C': [0.01, 0.1, 1, 10, 100],
        'solver': ['liblinear', 'saga'],
        'l1_ratio': [0, 0.5, 1]
    }
}
```

Figure 24: Parameters selection for GridSearch

The code snippet below shows the implementation of the GridSearch.

```

for model_name, model in models.items():
    print(f"Grid Search for {model_name}")
    grid_search = GridSearchCV(model, param_grid[model_name], cv=5, scoring='f1', n_jobs=-1)
    grid_search.fit(X_train_smotetom, y_train_smotetom)
    best_models[model_name] = grid_search.best_estimator_
    test_predictions[model_name] = grid_search.best_estimator_.predict(X_test)
    train_predictions[model_name] = grid_search.best_estimator_.predict(X_train_smotetom)
    grid_search_results[model_name] = {
        'best_params': grid_search.best_params_,
        'best_f1_score': grid_search.best_score_
    }

```

Figure 25: GridSearch implementation

GridSearch outputs the parameters that optimize the f1-score for the best scorer models in the specified grid.

	GridSearch Best Weighted f1-score on test set (SMOTETomek Sampling)
Linear Discriminant Analysis	0.8414
Ridge Classifier	0.8414
AdaBoost (*)	0.8414
Logistic Regression(*)	0.8409

Table 7: Best scorer models performance with GridSearch

The achieved weighted f1-score on the test set for both the Linear Discriminant Analysis and the Ridge Classifier has not been improved with **0.8414**. The best hyperparameters seem to have already been found.

Model Optimization with Voting Classifier

A voting classifier with all the best scorer models has been implemented with the majority voting in order to check if the combination of several base models working together will further improve the weighted F1-score.

```

# voting classifier using lda, ridge svc, adaboost, and smotetom dataset
voting_clf = VotingClassifier(
    estimators=[
        ('lda', LinearDiscriminantAnalysis(shrinkage = None, solver = 'svd')),
        ('ridge', RidgeClassifier(alpha = 0.01, solver = 'auto')),
        ('svc', SVC( C = 10, gamma= 0.0001, kernel= 'rbf')),
        ('adaboost', AdaBoostClassifier(algorithm = 'SAMME', learning_rate = 0.1, n_estimators = 300)),
        ('logreg', LogisticRegression()),
    ],
    voting='hard' # Majority voting
)

```

Figure 26: Ensemble Voting implementation for performance optimization

A **weighted F1-score of 0.8414** was achieved by the Voting Classifier, marking no further improvement.

The focus will now be set on ensuring the best scorer models robustness with a stratified k-fold cross validation.

Testing the model's robustness with stratified k-fold cross validation

A 50-folds stratified cross validation has been performed and the F1-score has been monitored. The goal is to ensure a low variance in the weighted F1-score over the folds by guaranteeing the same class distribution as in the SMOTETomek resampled train set.

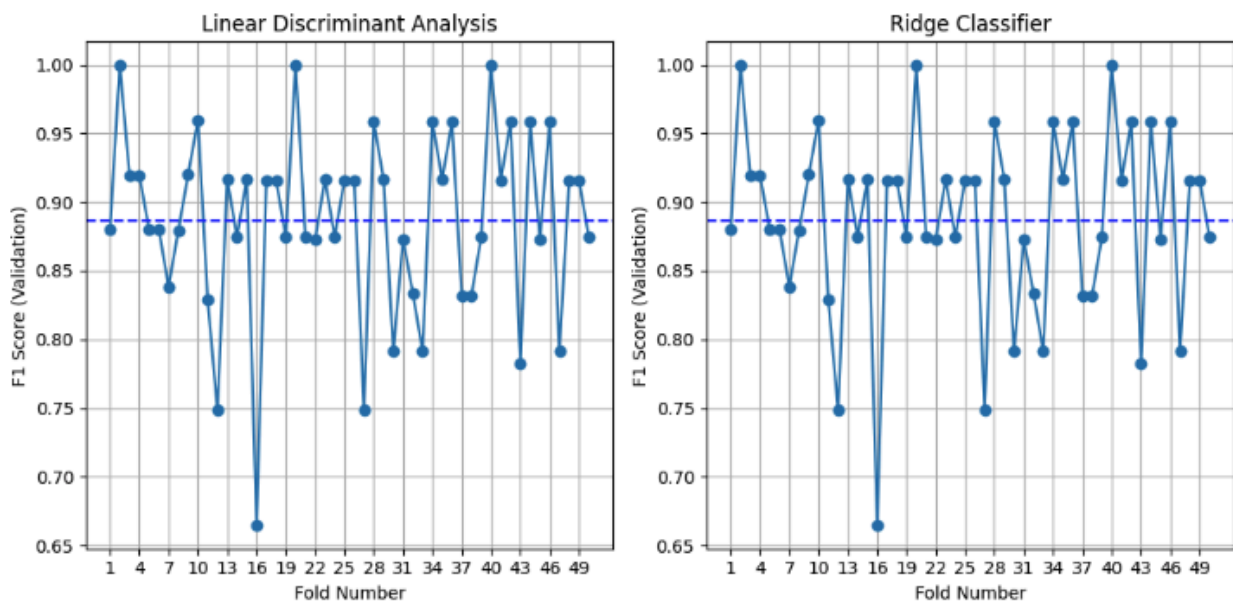


Figure 27: 50-fold cross validation for best scorer models robustness

On the best scorer models, a low variance of the F1-score 0.0046 was measured.

```
Best Model (based on average validation F1 with SMOTETomek): AdaBoostClassifier
Test F1 Score (on original unseen data): 0.7442
Variance of F1 Scores for each model:
Linear Discriminant Analysis: 0.0046
Ridge Classifier: 0.0046
AdaBoostClassifier: 0.0043
```

Figure 28: Best scorer models variance

This suggests that these models may exhibit reasonably stable performance on unseen data, potentially making them suitable for production use.

Model Interpretation

In our analysis, we evaluated several classification algorithms—Support Vector Classifier, Logistic Regression, AdaBoost, and Linear Discriminant Analysis—to find the best fraud detector. To understand not only *which* model performs best but also *why* it makes its decisions, we now turn to model interpretation. Specifically, we want to see which features consistently drive the predictions across different algorithms.

For this purpose, we use **SHAP** (SHapley Additive exPlanations). SHAP assigns each feature an importance value for a given prediction by estimating how much that feature changes the output compared to a baseline. We will apply the appropriate SHAP explainer to each model (KernelExplainer for SVC and AdaBoost, LinearExplainer for Logistic Regression and LDA) and then compare the resulting feature-importance rankings. This will reveal the key drivers of fraud predictions on average, helping us to interpret and trust our models more deeply.

LDA

We begin our SHAP analysis with Linear Discriminant Analysis. The results show that **"incident_severity_Major Damage"** is the most influential feature, followed by **"incident_severity_Minor Damage"** in second place. Among the hobby variables, **"insured_hobbies_chess"** ranks third, while **"insured_hobbies_crossfit"** appears in sixth position

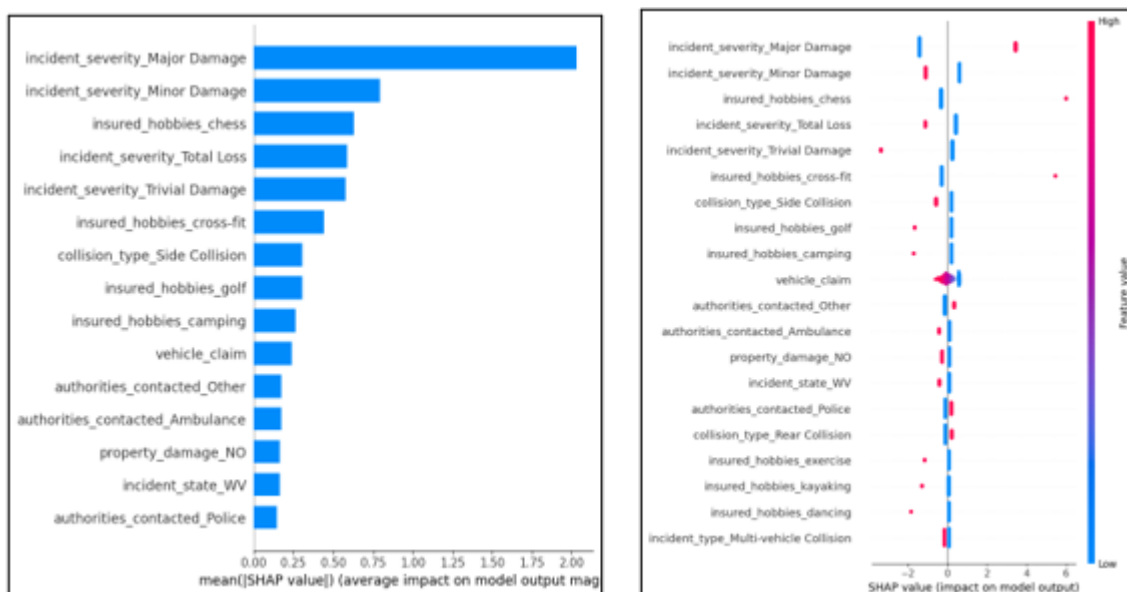


Figure 29: SHAP for LDA model

Logistic Regression

We also performed a feature importance analysis for Logistic Regression using SHAP values. The results show that the top three features are similar to those identified for Linear Discriminant Analysis, although the ranking of the features differs. SHAP provides a more detailed interpretation by showing how each feature contributes to the model's predictions, allowing for a clearer comparison between the models.

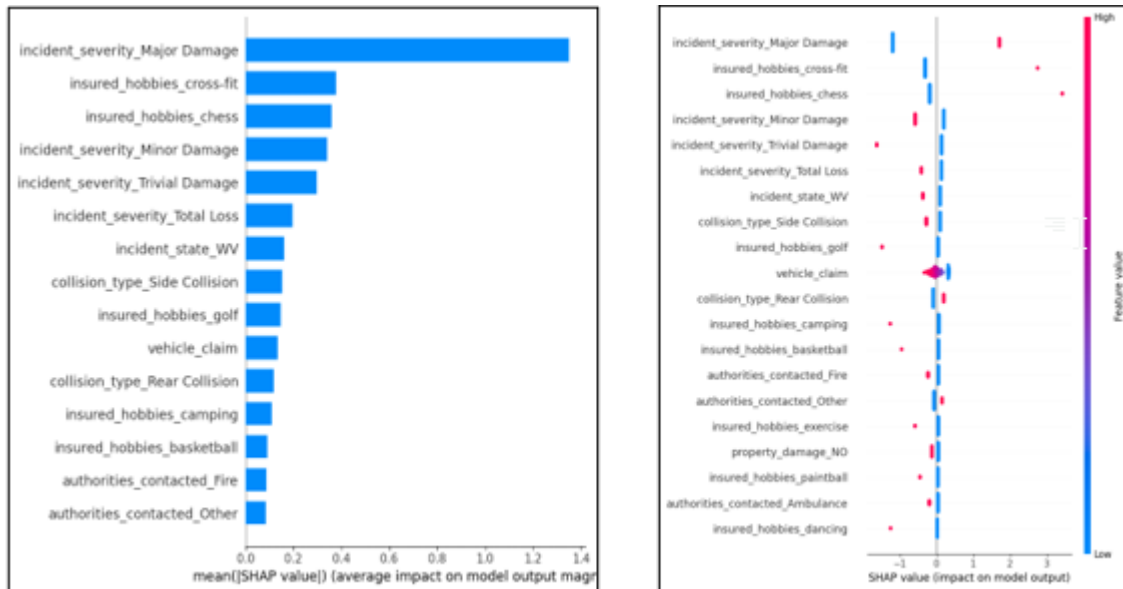


Figure 30: SHAP for Logistic Regression

SVC

We repeated the analysis using the SVC model to see how its feature importance compares. In this case, "**incident_severity_major_damage**" again emerges as a key driver. At the same time, the **crossfit** feature drops to fifth place in the ranking, while "**incident_severity_minor_damage**" also appears to have a noticeable impact.

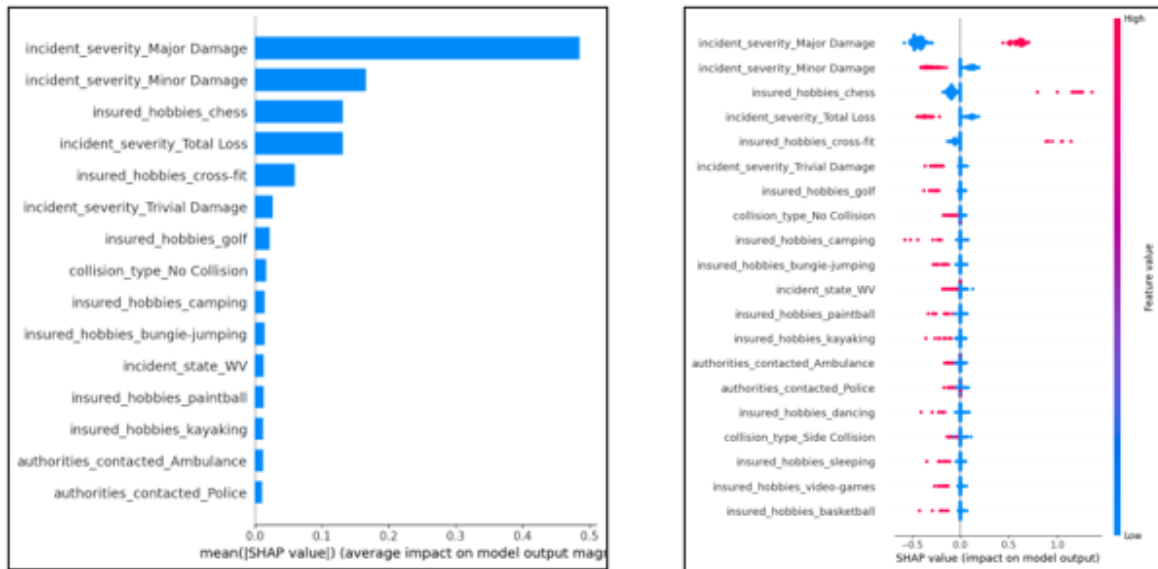


Figure 31: SHAP for SVC

AdaBoost

In conclusion, we applied SHAP to the AdaBoost classifier trained on our SMOTETomek-resampled data to gain a deeper understanding of its decision process. By wrapping the model's `predict_proba` call in a simple function and using `shap.Explainer`, we generated both a summary bar plot and waterfall plots for individual test samples.

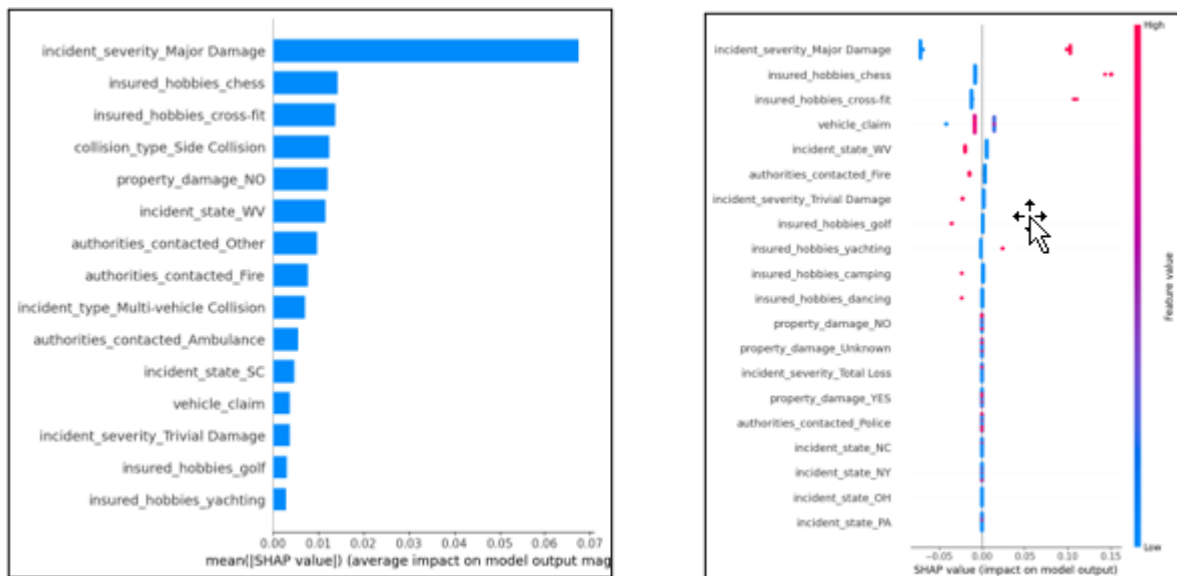


Figure 32: SHAP for AdaBoost

Summary

The expanded table now includes AdaBoost alongside LDA, Logistic Regression, and SVC

	Rank LDA	Rank LR	Rank SVC	Rank Ada
incident_severity_Major_Damage	1	1	1	1
insured_hobbies_chess	3	3	3	2
insured_hobbies_crossfit	6	2	5	3
incident_severity_Minor_Damage	2	4	2	-

Table 8: Features importance ranking by models

Key takeaways:

- incident_severity_Major_Damage dominates everywhere.
Across all four models, "**incident_severity_Major_Damage**" is consistently ranked first, confirming its overwhelming predictive power for fraud.
- incident_severity_Minor_Damage plays a secondary role except in AdaBoost. It ranks second for LDA and SVC and fourth for Logistic Regression, but AdaBoost gives it no top-4 placement, choosing instead to focus on the hobby features after Major Damage.
- **Hobby features vary by algorithm.**
 - insured_hobbies_chess remains a reliable predictor (ranked second–third in all but LDA, where it's third).
 - insured_hobbies_crossfit is second for LR, third for AdaBoost, but falls to fifth for SVC and only sixth for LDA.

We also ran a **GridSearchCV** for each model, which yielded only marginal gains in weighted F1-score and did not alter this feature hierarchy. This stability suggests that our chosen hyperparameters already allow each classifier to fully leverage its most informative inputs.

Finally, this SHAP-based interpretation helps explain why our earlier **ordinal encoding** of "**incident_severity**" underperformed: by imposing equal numeric spacing between categories (e.g. "Trivial → Minor" vs. "Major → Total Loss"), we inadvertently dampened the critical, non-linear leap that "incident_severity_Major_Damage" provides. Keeping the feature one-hot encoded preserves its true impact across all models.

Finals Conclusions

The culmination of our modelling efforts yielded promising results in predicting insurance fraud. Our top-performing models, namely **Linear Discriminant Analysis** (using the Smote Tomek resampler) followed by the **Ridge Classifier**, and **AdaBoost**, achieved weighted average F1-scores of 0.84-0.85. Given the imbalanced nature of our dataset, with a significantly higher proportion of non-fraudulent claims, we prioritized the F1-score as our primary evaluation metric, as it provides a balanced view of precision and recall.

The feature importance analysis from these models provided valuable insights into the factors deemed most influential in predicting fraudulent activity, (namely **"insured_hobbies_chess"**, **"insured_hobbies_crossfit"** and **"incidentseverity_major_damage"**).

Furthermore, the robustness of our selected models was assessed through stratified k-fold cross-validation, demonstrating consistent performance across different data partitions.

Challenges

The achieved best F1-scores of 0.84-0.85 suggest a reasonable capability of our models to identify potentially fraudulent insurance claims. This outcome addresses our initial objective of developing effective models for fraud detection. However, the project was not without its challenges. We encountered a plateau in score improvement after the initial model runs, leading to a phase of iterative experimentation that, at times, **felt cyclical**. This difficulty could potentially be attributed to the limited size of our dataset.

As newcomers to the intricacies of these specific machine learning models, our approach was largely one of **"learning by doing."** This involved navigating the complexities of feature encoding, selection, and the optimal sequencing of preprocessing steps, which inevitably introduced delays as we iteratively refined our understanding. The practical **challenges of collaborative coding**, such as code loss during branch merging, also presented temporary setbacks, although these **ultimately fostered a deeper understanding** through the need of rewriting. Similarly, initial uncertainties regarding the order of preprocessing steps required **significant code revisions** to ensure the correct flow of data transformation. Managing the array of models explored also became cumbersome, necessitating a focused effort to consolidate our findings and identify the most promising candidates based on our evaluation metrics. The small sample size of 1000 claims and the anonymized nature of the data, particularly the lack of specific location information and a limited temporal window, posed inherent limitations on the generalizability of our findings. Lastly, our team's initial lack of domain expertise in the insurance industry required us to rely on external resources to understand the significance of various features.

Contributions

Our contributions to the project were diverse and targeted toward building a robust fraud detection system. We began with a **comprehensive data cleaning and exploration**, including handling missing values and identifying key variables through **statistical analysis**.

The dataset was carefully **prepared for modelling**, with categorical variables encoded, numerical features scaled, and an early **train-test split** to prevent data leakage.

We trained a wide range of **binary classification models**—including Logistic Regression, SVM, Random Forest, Decision Trees, LDA, Ridge Classifier, AdaBoost, and XGBoost. To address the **significant class imbalance**, we applied different **resampling techniques** such as RandomOverSampler, RandomUnderSampler, and SMOTETomek, and selected the **F1-score as the main evaluation metric**.

The most relevant features were identified through **visualization techniques**, and we performed **hyperparameter tuning using GridSearch** on the top-performing models (LDA, Ridge, and AdaBoost). To ensure the **reliability of results**, we used **Stratified K-Fold Cross-Validation**. Throughout the process, we also **refined the codebase** to improve efficiency and maintainability.

Benchmark

Due to the anonymized nature and specific characteristics of the provided insurance claims dataset, identifying a direct external benchmark for comparison was challenging. Nevertheless, we reviewed similar projects on platforms such as **Kaggle** and **GitHub** to gain context.

One referenced project **did not address class imbalance** and relied heavily on **accuracy as the primary metric**, making their results unsuitable for comparison with our F1 score-driven approach. In contrast, another project did incorporate **class balancing techniques** and implemented creative **feature engineering**, such as generating categorical variables based on **time-of-day segments** (morning, midday, evening, night). While insightful, their best **weighted average F1 score of 0.67** remained significantly lower than our **best score of 0.85**.

We attribute this performance difference to several key factors. Our approach included a **rigorous feature selection process**, retaining only those variables with a **p-value below 0.05** or high inter-correlation with the target, thereby reducing noise and improving model focus.

Additionally, our consistent use of **one-hot encoding** for categorical variables appears to have contributed positively to model performance. In contrast, preliminary tests using **ordinal encoding** showed no performance gains and may have inadvertently introduced **false ordinal relationships** which could distort model learning.

These methodological choices—particularly regarding **feature selection and encoding strategies**—likely played a major role in achieving stronger and more reliable model outcomes.

Achievement of Project Goals

We believe that we have successfully achieved the primary goals of this project. The developed models demonstrate a capability to detect potentially fraudulent insurance claims,

as evidenced by the F1 scores around 84%. Furthermore, the implementation of stratified k-fold cross-validation suggests that the prediction scores are reasonably stable and robust across different subsets of the data. Our findings align with the broader economic context of insurance fraud detection, indicating the potential for our models to contribute to the early identification of high-risk cases. This, in turn, could enable insurers to reduce financial losses, optimize investigation resources, and streamline claim validation processes through data-driven thresholds.

Contribution to Scientific Knowledge

While the primary focus of this project was the practical application of machine learning techniques, it also contributes to scientific understanding by showcasing a **comprehensive end-to-end data science workflow** applied to a real-world problem in the **insurance domain**. The analysis yielded **specific insights into fraud-related patterns** within the dataset and emphasized the importance of **thoughtful feature selection**, **proper handling of class imbalance**, and **robust model evaluation** strategies.

In addition, the project offered valuable **practical lessons**, including managing the **learning curve of unfamiliar models** and handling the **complexities of iterative development**. By prioritizing the **F1 score** as the key evaluation metric, the project reinforces the necessity of choosing **problem-appropriate metrics**, especially in contexts involving **imbalanced datasets**.

These reflections and methodological choices provide **useful guidance for future applied machine learning projects** in similar domains.

Lessons Learned

Through this project we gained several valuable insights that will inform our future work. Firstly, we recognized the importance of **developing a well-defined plan for the entire process** from the outset. Establishing a clear sequence of steps can significantly optimize our workflow and prevent redundant efforts. Secondly, we learned the benefit of **setting aside the test dataset early in the project lifecycle**. This ensures an unbiased evaluation of our final models.

Furthermore, the necessity of **conducting a thorough audit and cleaning of the data** became evident. A comprehensive understanding of the data's characteristics and meticulous preparation are paramount for successful modelling. Moving forward, **if possible, we should strive to obtain more data, aiming for a larger sample size**, as this could potentially enhance the generalizability and robustness of our models. We found that our strategic approach to feature engineering during the modelling phase proved effective; however, we also recognize the need to **be more creative in feature engineering and explore different approaches** to potentially uncover more subtle patterns in the data.

Another key takeaway was the value of **adopting an incremental approach to model development**. Instead of attempting multiple complex tasks simultaneously, it is more beneficial to begin with simpler baseline models to establish initial performance metrics. In

retrospect, starting with a "lazy classifier" could have provided a useful early indicator.

Finally, we identified an area for future exploration in **experimenting more extensively with different encoding strategies** for categorical variables. Further investigation into various encoding techniques may reveal additional insights and potentially improve model performance.

In conclusion, this project has successfully demonstrated the potential of machine learning techniques for the effective detection of insurance fraud within the constraints of the provided dataset. The insights gained through rigorous experimentation, feature analysis, and model evaluation provide a solid foundation for future endeavours in this domain. By addressing the identified limitations and incorporating the lessons learned, subsequent iterations of this work could further enhance predictive accuracy and contribute even more significantly to mitigating the financial impact of fraudulent activities within the insurance industry.

Bibliography

- Geron, Aurelien (2023): "Hands-On Machine Learning with Scikit Learn, Keras and TensorFlow", Third Edition
- Raschka, Sebastain; Mirjalili, Vahid (2021): "Machine Learning mit Python und Keras, TensorFlow 2 und Scikit-learn", 3., aktualisierte und erweiterte Auflage
- Bruce, Peter; Bruce, Andrew; Gedeck, Peter (2021): "Praktische Statistik für Data Scientists", Übersetzung der 2. Auflage
- DataScientest Training Material

Comparable Projects found in the web:

<https://www.kaggle.com/code/niteshyadav3103/insurance-fraud-detection-using-12-models/notebook>

<https://github.com/saritmaitra/Fraud-detection--Insurance>