Car Insurance Fraud Detection

Applied Machine Learning 2026

Anika Hedegaard, Anna Brendstrup, Christian Albæk & Freja Junge

June 10, 2026

Problem & Strategy

To what extent can machine learning models improve the detection of fraudulent car insurance claims?

Strategy

- Supervised classification: Identify fraud using information about policyholder in tabular data
- Supervised classification: Identify fraud using images of the damage
- Exploratory unsupervised approach: Use clustering to find groups with high rate of fraud worth a closer look

Does any of the datasets/approaches flag fraud reliably?

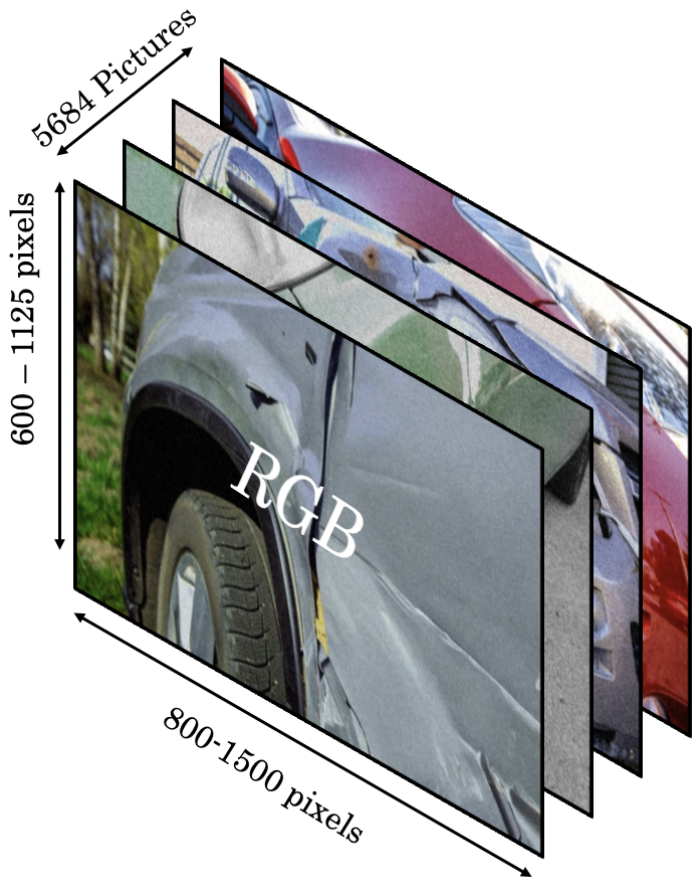
Data Overview

Two Datasets:

Tabular data & Image data

15,420 Rows

32 features				Target
6 Numerical		17 Categorical	9 Ordinal	Fraud
Data	Age 320 NA	Data	Data	Data

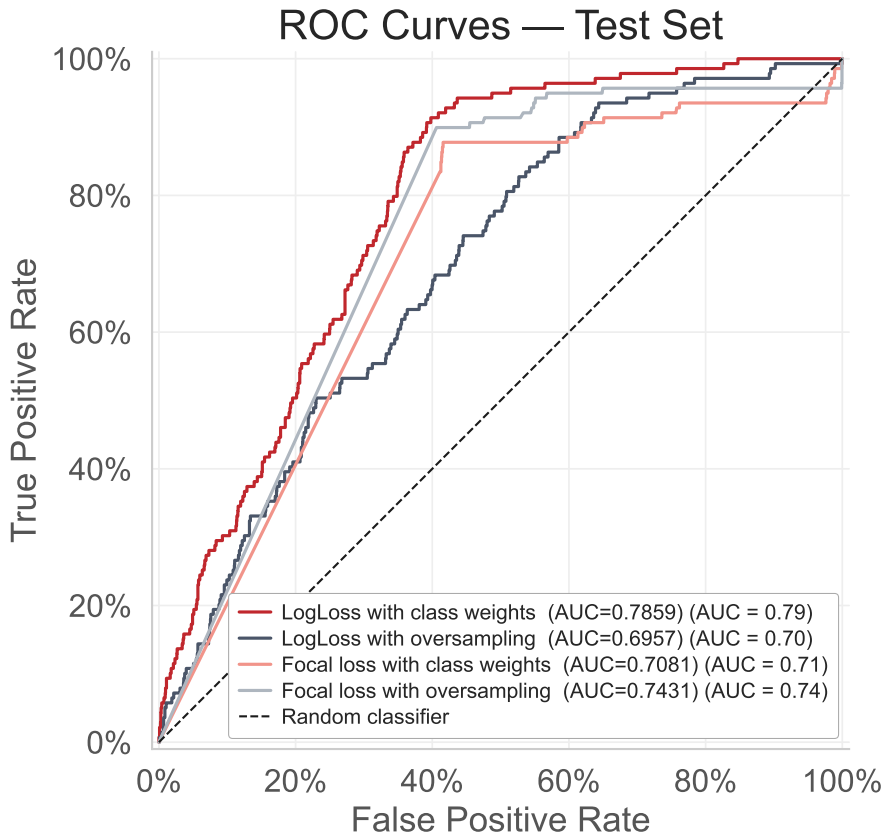


Class Imbalance

Problem: Both datasets are highly imbalanced (6% fraud tabular, 4% fraud images).

Based on the tabular data we compared different methods using CatBoost:

- LogLoss with class weights
- Focal loss with class weights
- LogLoss with x5 oversampling
- Focal loss with x5 oversampling



Takeaway: Class weights with LogLoss beats the other methods

Tabular data: Modelling Pipeline

- **Preprocessing**
  - Removed ID variables, ordinal + one-hot encoding
  - Age imputation (2% of data)
  - Stratified 70/15/15 split
- **Baseline comparison**
  - Default params: XGBoost, CatBoost, LightGBM, Random Forest, PyTorch & TensorFlow NN
  - Tree-based model outperformed NN-based models.
- **Hyperparameter tuning**
  - Random search $\rightarrow$ Bayesian optimisation (Optuna)
  - Objective: maximise validation AUC
- **Final evaluation**
  - Retrained on train + val, evaluated once on test
- **Threshold optimisation?**
  - The optimal threshold is likely an administrative decision

Tabular data: Preprocessing

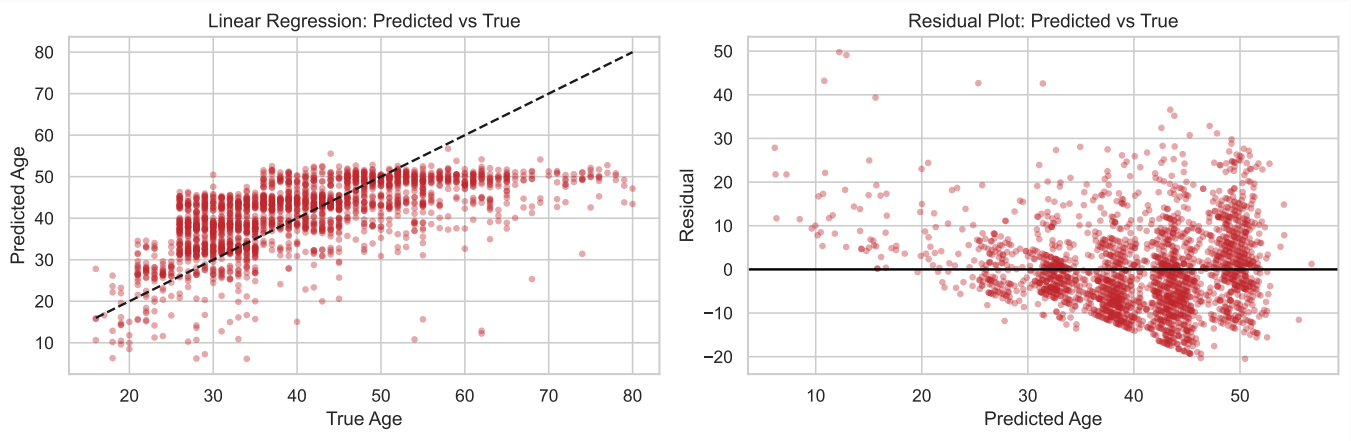
One-hot encoding

VehicleCategory	_Sport	_Utility
Sedan	0	0
Sport	1	0
Utility	0	1

Age imputation

Replace mislabelled zeros (2% of dataset).

- Compare Linear Regression, tree-based models and NN models
- Non-linear ML models are strongly motivated
- A tuned Catboost model was used for the imputation based on AUC



Tabular data: Results

Bayesian Optimisation Results

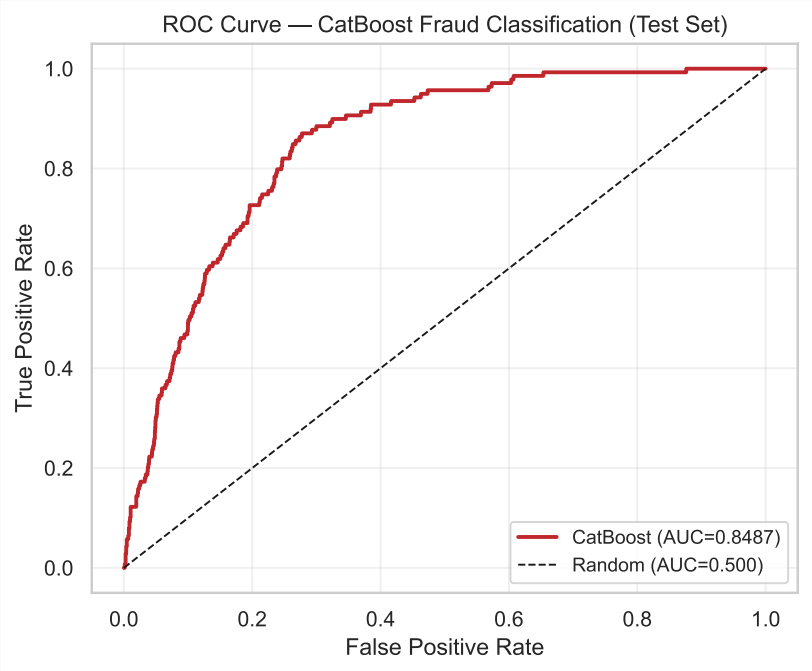
Validation Set

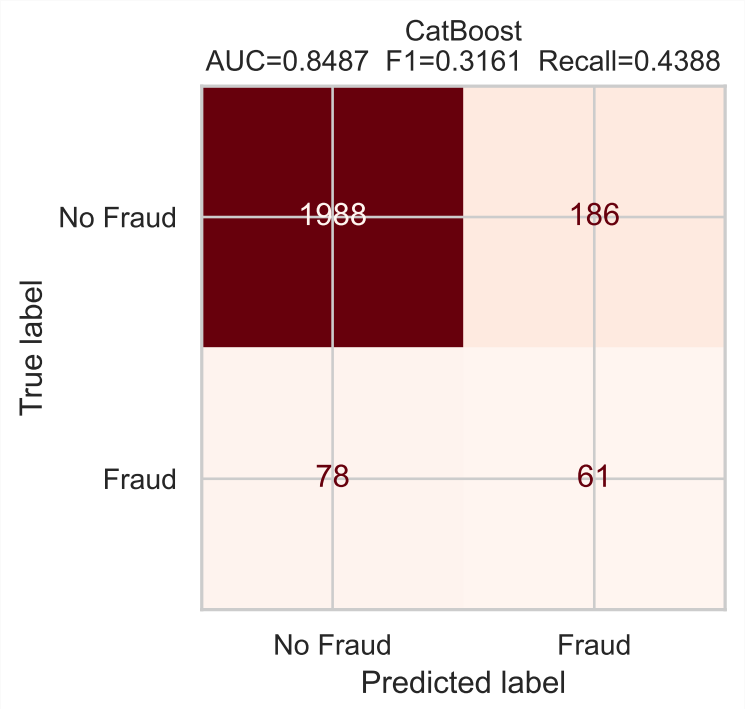
Model	Initial AUC	Tuned AUC
CatBoost	0.856	0.873
XGBoost	0.859	0.868
LightGBM	0.802	0.861
Random Forest	0.815	0.847
PyTorch NN	0.792	0.829
TensorFlow NN	0.814	0.829

Test Set

Model	AUC (ROC)	F1
CatBoost	0.849	0.316
XGBoost	0.846	0.297
LightGBM	0.830	0.267
Random Forest	0.828	0.265
PyTorch NN	0.791	0.174
TensorFlow NN	0.789	0.160

Final model: CatBoost

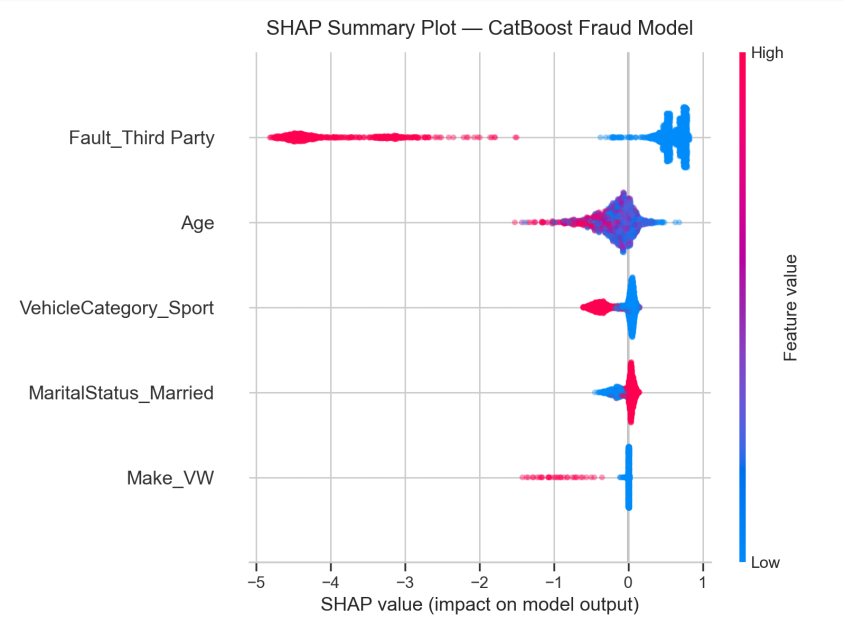




Tabular Data: Explaining the Final Fraud Model

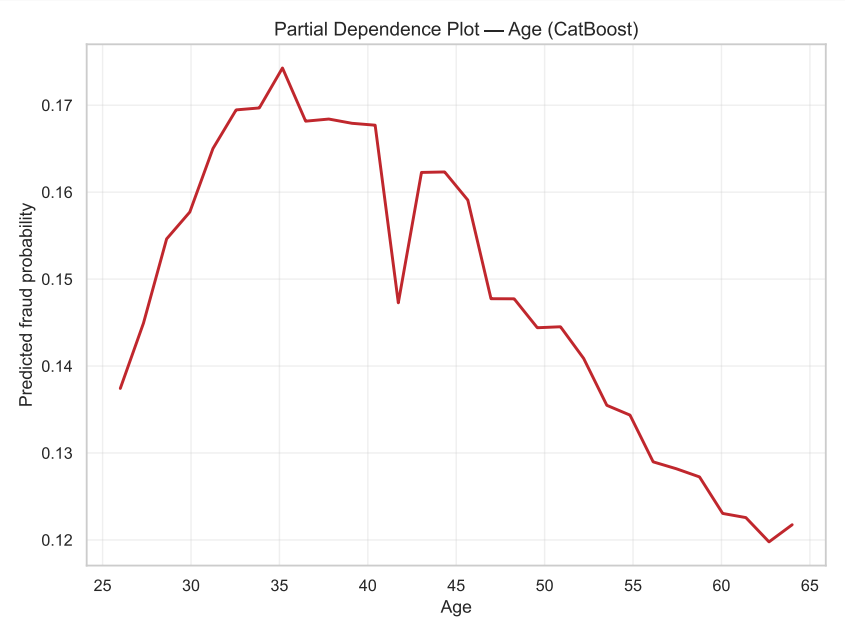
Why? Forsikringsaftaleloven § 3b (Law on Insurancecontracts) requires insurers to justify a refusal and cancellation. A prediction alone is not enough.

SHAP: Feature importance



- Claim where a third party is at risk is most influential

PDP: Effect of Age



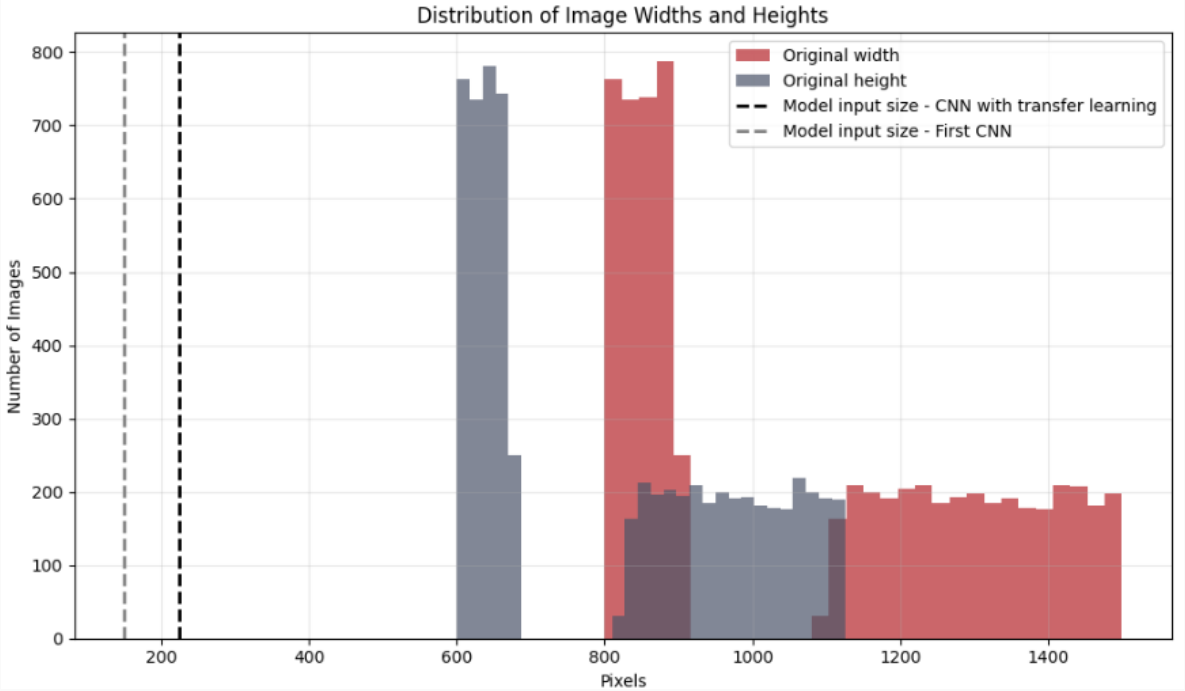
- Fraud risk peaks around 35

Image Data: Preprocess and Handling of Image Data

Image pixels are standardized

Images are resized

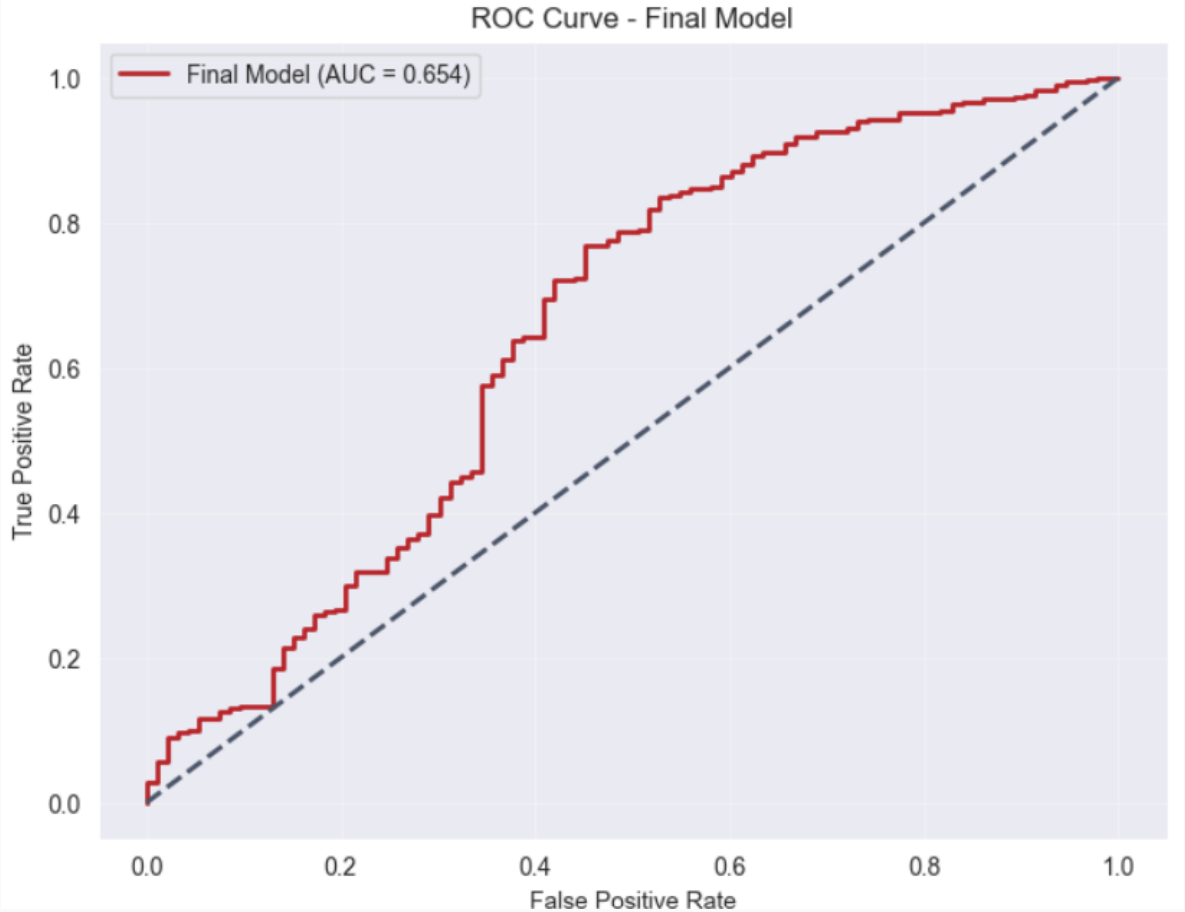
- Input size in CNN without transfer learning (150, 150, 3)
- Input size in CNN with transfer learning (224, 224, 3)



Data is labeled with 4% Fraud and 96% Non-fraud $\rightarrow$ class weights

Image Data: CNN Without Transfer Learning

The best CNN without transfer learning



Conclusion: It is not efficient to build a fraud predicting CNN without transfer learning on the image data.

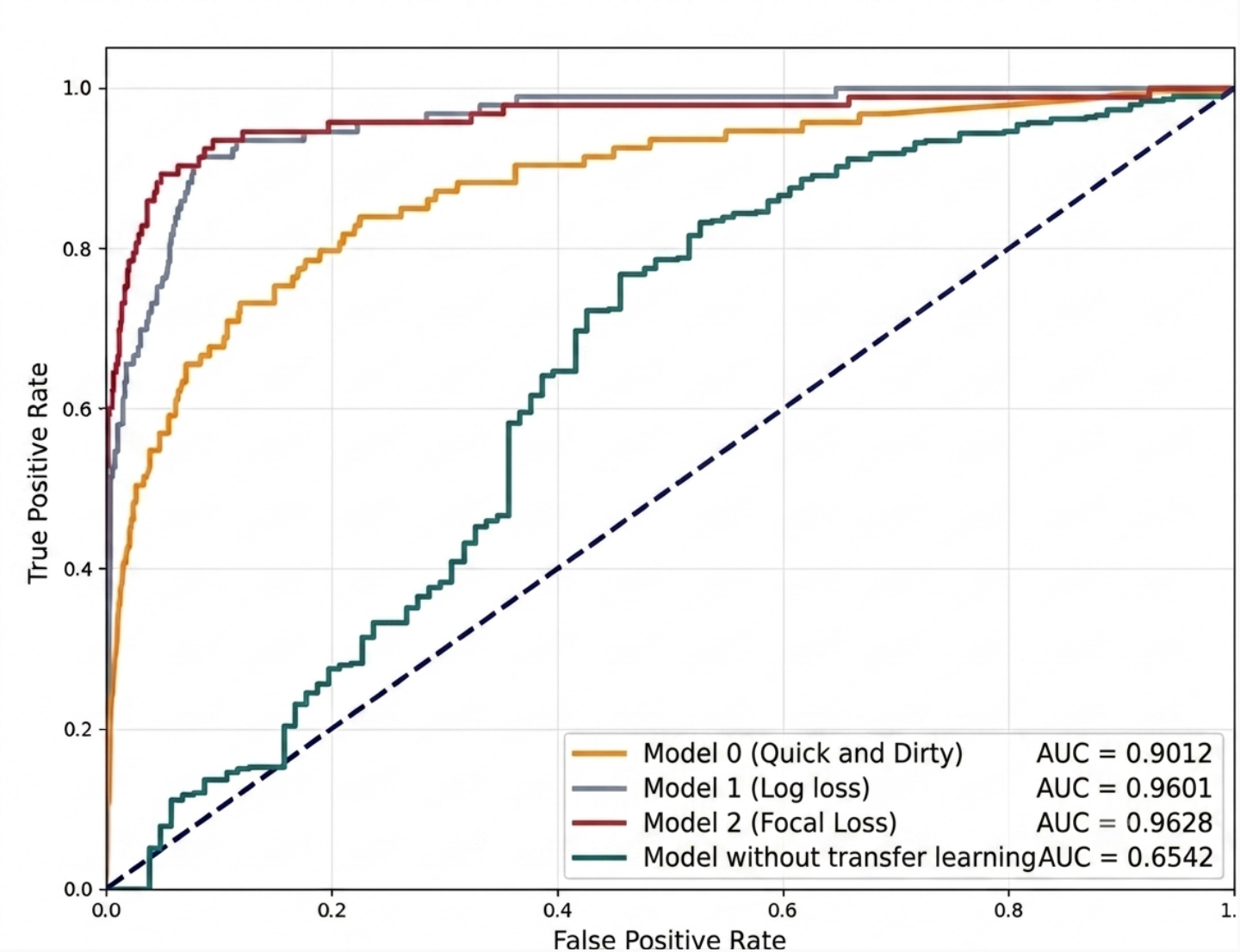
Image Data: Transfer learning

1. Preprocessing (Same as described earlier)
2. First Model - Quick & Dirty (AUC 0.90)
3. Data Augmentation (AUC 0.91)
4. Hyperparameters (AUC 0.93)
5. Fine Tuning (AUC 0.94)
6. EfficientNet50 (AUC 0.94 and lower computational time)
7. Cropping
    - Region of interest (AUC 0.94)
    - Center Crop (AUC 0.96)
    - Grey Scale (AUC 0.90)

Image Data: What does the model look at?

Image Data: ROC-Curves

Last change: Focal loss (AUC 0.96)



Overall Conclusion (Supervised)

- **Best approach?**
  - Image data without transfer learning (AUC 0.6542)
  - Tabular data (AUC 0.8487)
  - Image data with transfer learning (AUC 0.9628)
- **How could we compare tabular and image data more rigorously?**

Image Data: Can fraud be clustered?

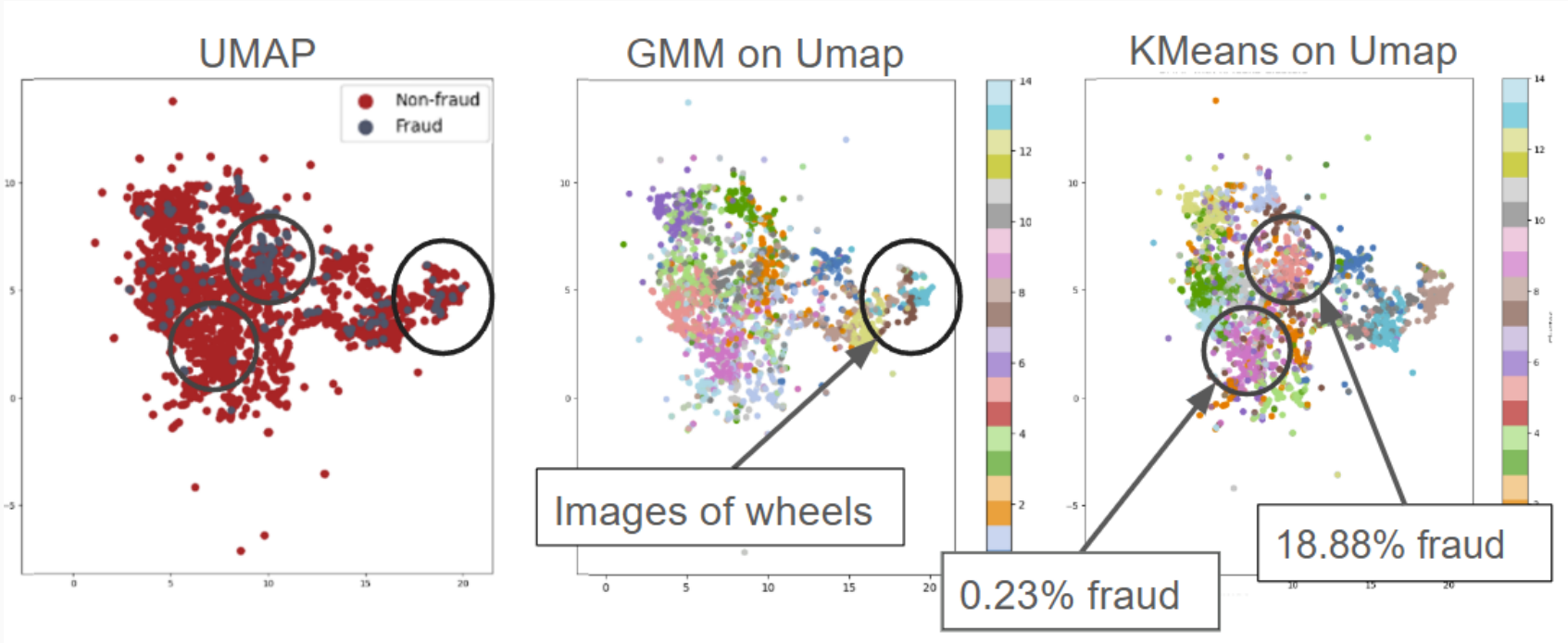
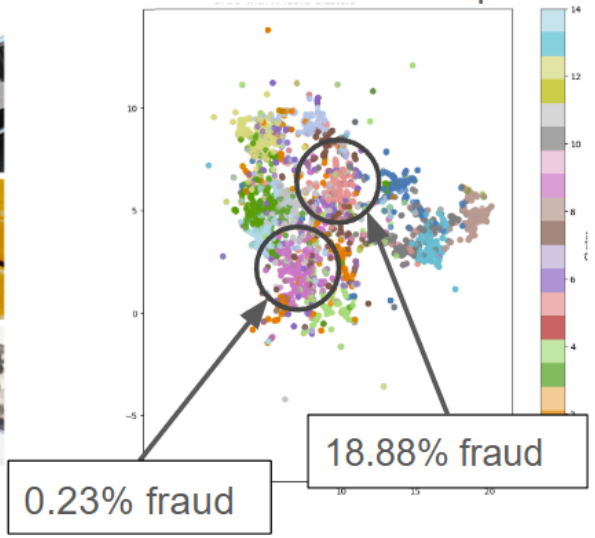


Image Data: Can fraud be clustered?

18.88% fraud

0.23% fraud

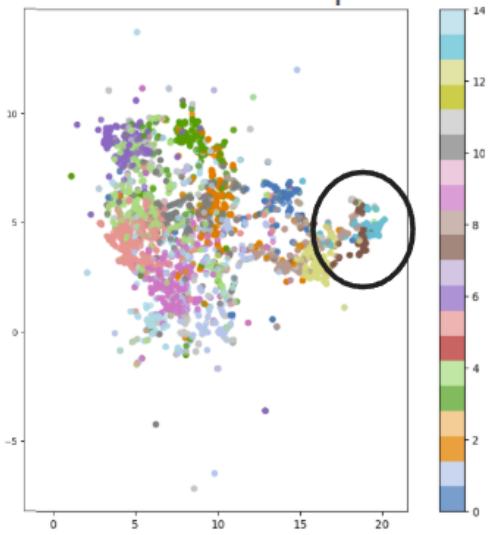
KMeans on Umap



11% fraud

2% fraud

GMM on Umap



Appendix

Aim of Appendix

This appendix documents the full modelling pipeline, providing transparency into the choices, strategies and considerations underlying the presented.

- Details on the data preprocessing steps, including feature encoding and handling of missing values
- Document the modelling decisions made at each stage so that the reasoning behind each choice is clear

Motivation

- Insurance fraud costs the Danish insurance industry approximately 500 million DKK annually in detected cases alone[1]
- Fraud drives up premiums for honest customers and increases operational costs across the industry
- Can machine learning offer a data-driven approach to detecting fraudulent claims early?

[1] If Forsikring (2025). *If og forsikringssvindel*. https://www.if.dk/om-if/om-os/if-og-forsikringssvindel

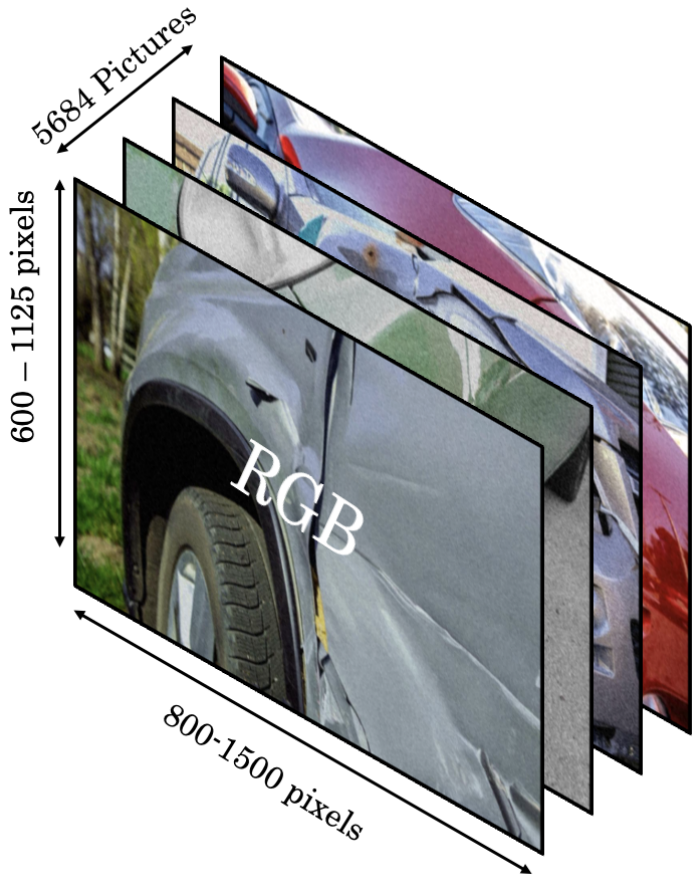
Problem Statement

- We investigate fraud detection using two fundamentally different data sets:
  - **Tabular data:** structured claim records with policyholder and accident information
  - **Image data:** Images of damaged vehicles associated with claims
- For each dataset, we train and compare multiple machine learning models to determine which approach yields the best fraud detection performance.
- The goal is to assess whether structured data or visual data is most effective for identifying fraudulent claims

Data Overview

We use two distinct and independent datasets presented below: a tabular dataset (link to tabular data) and an image dataset (link to image data).

	32 features			Target
	6 Numerical	17 Categorical	9 Ordinal	Fraud
15,420 Rows	Data	Data	Data	Data
	Age: 320 NA			



The target contains 6% fraud in tabular data and 4% for images.

Overall Project Pipeline

We approach the problem in two stages: an exploratory clustering phase across both datasets, followed by separate supervised modelling on each.

- **Clustering (exploratory):**
  - Unsupervised
  - Goal: find groups with elevated fraud rates worth a closer look
  - Done before any supervised modelling and with no fraud labels used in fitting
- **Tabular modelling:**
  - Supervised classification on the 32 tabular features
  - Handle class imbalance (6% fraud)
- **Image modelling:**
  - Supervised classification on the RGB images (CNN)
  - Handle class imbalance (4% fraud)
- **Conclusion:**
  - Does any of the datasets flag fraud reliably?
  - Can any model be used as an alert system in an insurance company?

Clustering

On both tabular data and images

Clustering: Introduction

Before training supervised models separately on each dataset, we first explore the data using unsupervised clustering.

- **Aim:** We will use clustering exploartory to possibly find natural groups in the data and check whether some groups have a substantially higher fraud rate than others. That is, segments worth keeping an extra eye on.
- **No labels used in fitting:** Clusters are formed without the fraud label. We only inspect the fraud rate afterwards to see if the structure is relevant for fraud.
- **Tabular vs. images:** We apply clustering to both the tabular data and the images, letting us compare what structure each data type reveals.

This sets the stage for the separate supervised modelling of the tabular and image data that follows.

Clustering: Tabular Data

Preprocessing:

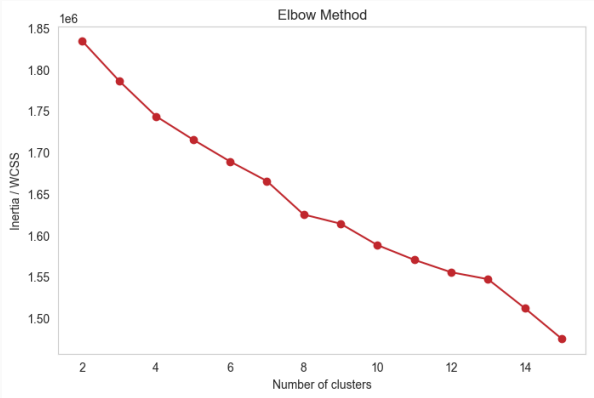
- Removed identifiers (PolicyNumber, RepNumber).
- Included numerical and categorical variables.
- One-hot encoded categorical variables.
- Standardized features.
- Applied PCA to reduce dimensionality and remove noise.

Clustering algorithm:

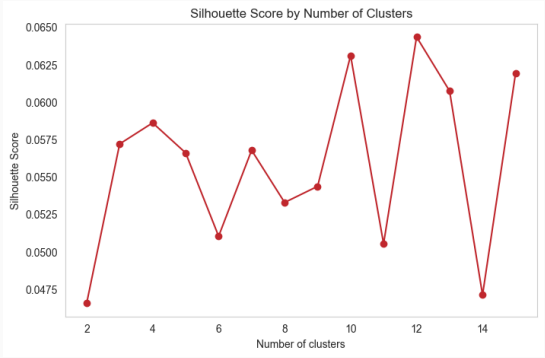
- K-Means and GMM clustering.
- Number of clusters evaluated for $k = 2, \ldots, 15$.
- Cluster quality assessed using:
  - Elbow Method (Inertia) (K-Means)
  - AIC/BIC Score
  - Silhouette Score
  - Davies-Bouldin Score

Clustering: Tabular Data

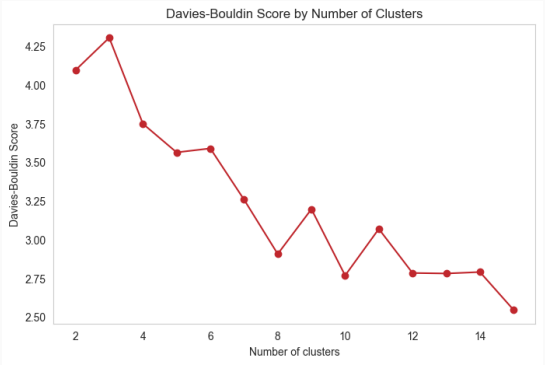
K-Means



Elbow Method



Silhouette Score



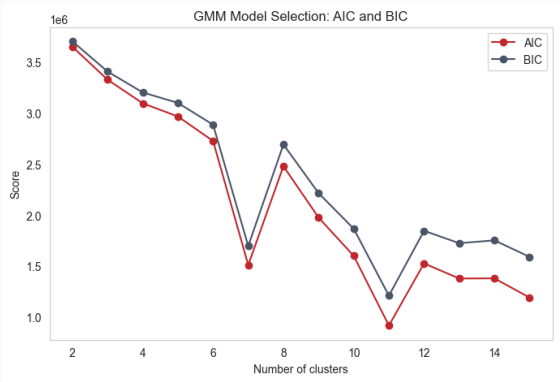
Davies–Bouldin

Selected Solution

The Elbow Method did not identify a clear optimum. The closest to an elbow looks like a number of 8 clusters. Hence we move on with 8 clusters.

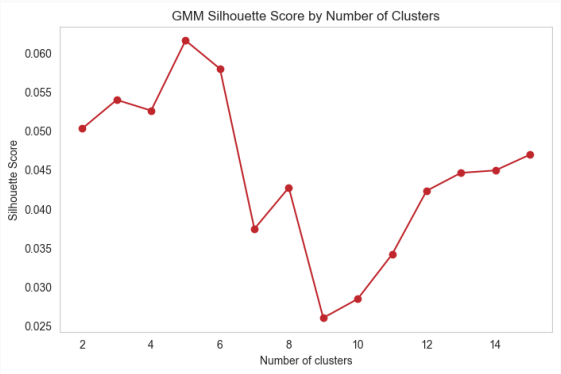
Clustering: Tabular Data

GMM



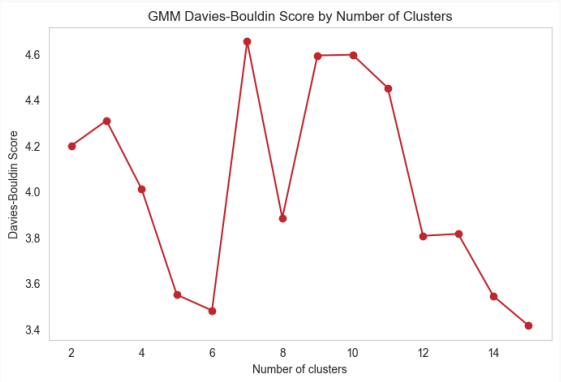
AIC/BIC

Model fit improves for larger numbers of clusters.



Silhouette

Higher silhouette values are primarily for lower amount of clusters.



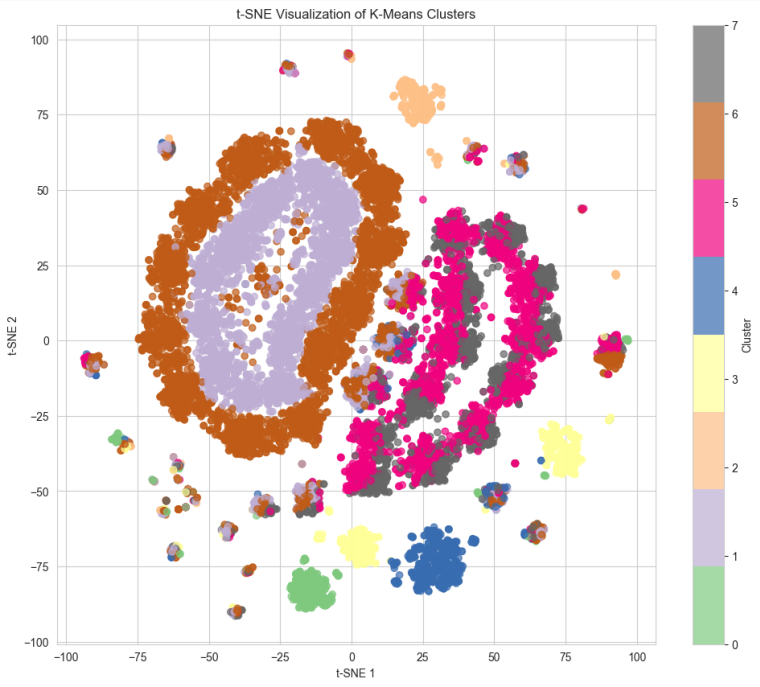
Davies–Bouldin

Substantially higher than K-Means.

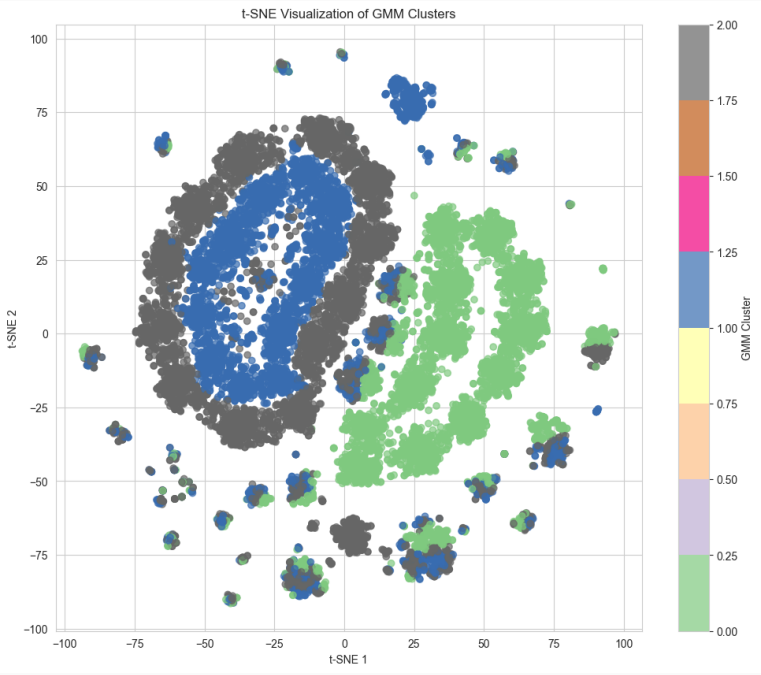
Conclusion

GMM produced weaker cluster separation than K-Means. However, we choose to model three clusters.

Comparison of Clustering Methods



K-Means ($k = 8$)



GMM ($k = 3$)

We have used t-SNE to visualize data and its clustering. In the above plot the clusters are found using K-Means and a Gaussian mixture model (GMM).

Clustering: Tabular Data

K-Means ($k = 8$)

Cluster	Count	Fraud	Rate
0	403	16	3.97%
1	3676	370	10.07%
2	393	44	11.20%
3	674	74	10.98%
4	697	39	5.60%
5	2797	21	0.75%
6	5063	349	6.89%
7	1717	10	0.58%

GMM ($k = 3$)

Cluster	Count	Fraud	Rate
0	5007	36	0.72%
1	4482	455	10.15%
2	5931	432	7.28%

Observation

Both methods identify clusters with substantially different fraud rates, suggesting that clustering captures relevant structure in the claims data.

Clustering: Tabular Data - Key Findings

K-Means ($k = 8$): VehicleCategory (%):

Cl.	Sedan	Sport	Util.	Fraud rate
0	75.4	24.6	0	3.97%
1	100	0	0	10.07%
2	0	0.5	99.5	11.20%
3	26.1	73.9	0	10.98%
4	64.8	35.2	0	5.60%
5	0	100	0	0.75%
6	100	0	0	6.89%
7	0	100	0	0.58%

K-Means ($k = 8$): BasePolicy (%):

Cl.	All P.	Coll.	Liab.	Fraud rate
0	29.3	48.6	22.1	3.97%
1	100	0	0	10.07%
2	86.5	7.6	5.9	11.20%
3	19.6	58.2	22.3	10.98%
4	26.3	40.3	33.4	5.60%
5	0.0	0.0	100.0	0.75%
6	0.0	100.0	0.0	6.89%
7	0.0	0.0	100.0	0.58%

GMM ($k = 3$): VehicleCategory (%):

Cl.	Sedan	Sport	Util.	Fraud rate
0	0	99.6	0.4	0.72%
1	91.2	0.5	8.3	10.15%
2	94.1	5.9	0	7.28%

GMM ($k = 3$): BasePolicy (%):

Cl.	All P.	Coll.	Liab.	Fraud rate
0	0	0	100	0.72%
1	99.3	0.7	0	10.15%
2	0	100	0	7.28%

Observations:

- K-Means clusters 1-3 and GMM cluster 1 stand out with fraud rates above 10%.
- High-fraud clusters are dominated by **All Perils** policies meaning broader coverage may attract more fraud.
- **Sport vehicles** appear in both high and low fraud clusters meaning that vehicle type alone is not sufficient to determine fraud risk.

Clustering: Images

Pipeline:

1. Preprocess the images and extract features using MobileNetV2 CNN
2. Use KMeans, DBSCAN and Gaussian Mixture Model to find clusters
3. Use t-SNE and UMAP to inspect data.
4. Inspect clusters

Clustering: Images - Preprocess the images and extract features

Preprocess:

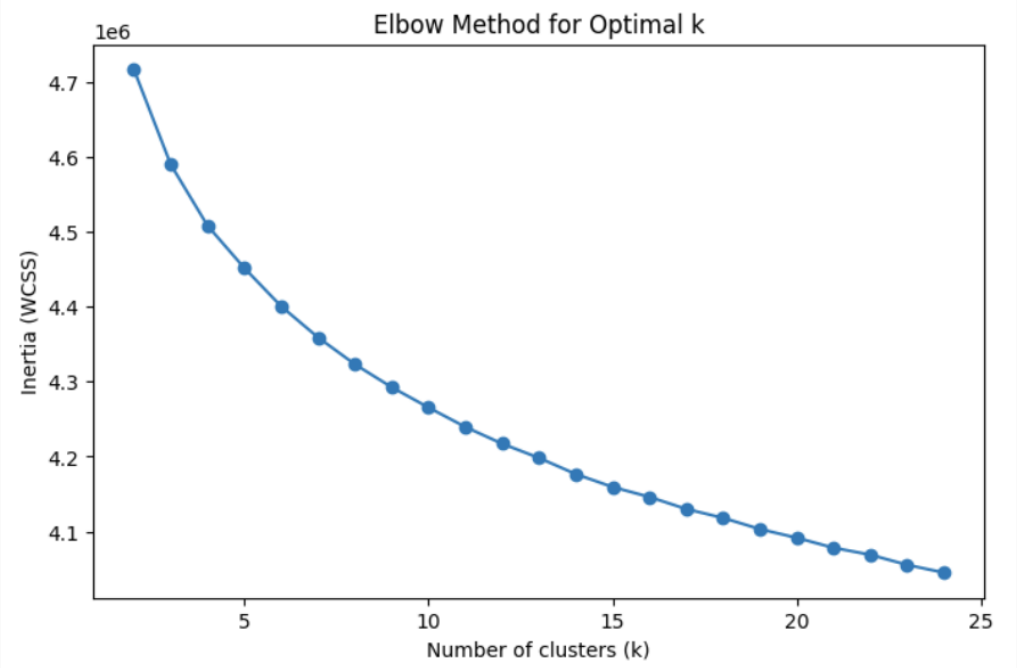
- **Resize:** All images are resized to (150x150) pixel.
- **Input size:** All images are in colour RGB and takes input (150, 150, 3) in the later CNN archtecture.
- **Normalization:** The pixel values get normalized i.e (0, 255) $\rightarrow$ (0, 1)

Extract features:

- Use MobileNetV2 from Tensorflow Keras to extract 1280 "features" for each image

Clustering: Images - Number of clusters

- From the elbow method below, no number of clusters were obvious.
- A small number of clusters, can cluster wheels, windsheeld etc.
- A large number of clusters might cluster within these groups.
- We hope to find clusters with different ratios of fraud. We therefore choose 15 clusters.



Clustering: Images - Use KMeans, DBSCAN and GMM to find clusters

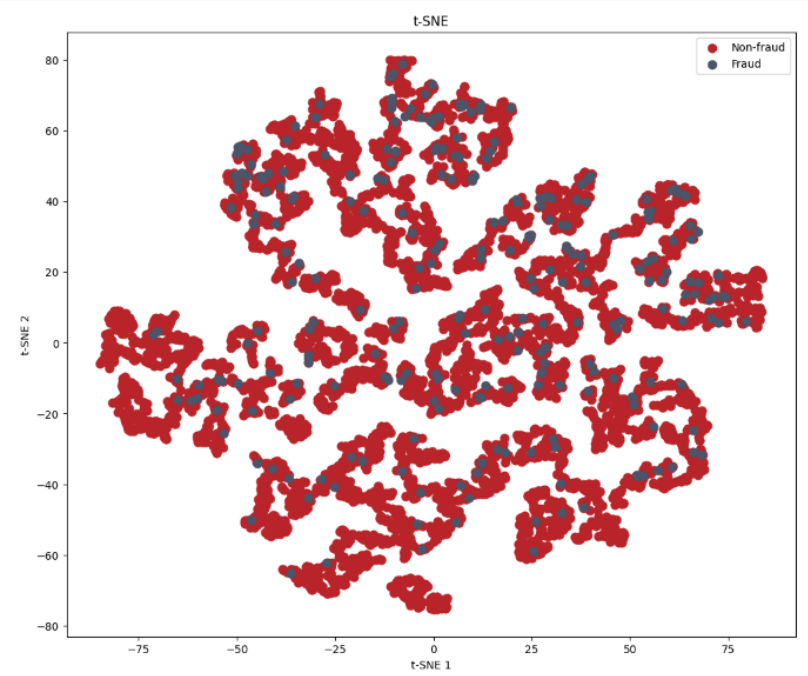
We try three different methods to find clusters:

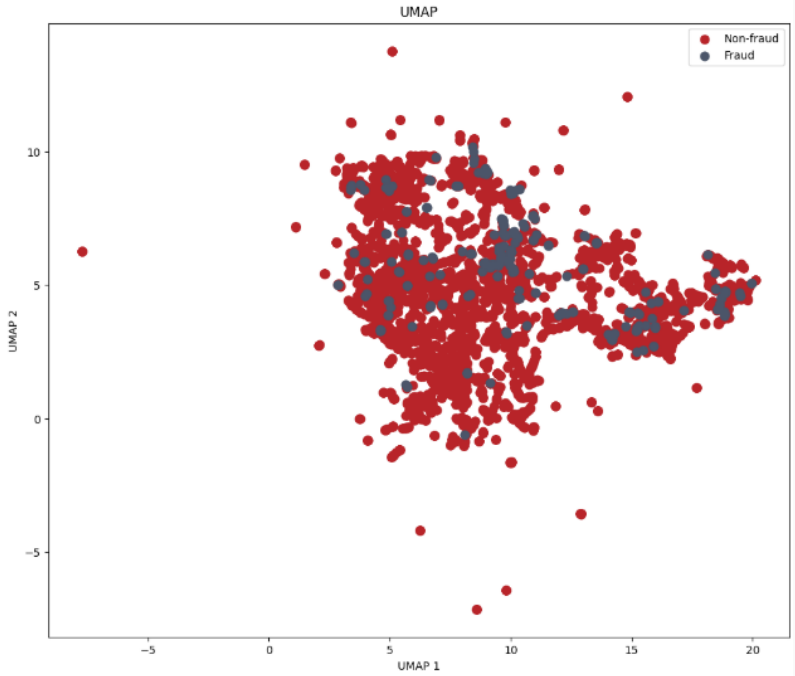
1. **KMeans:** we use KMeans from Sklearn to find 15 clusters.
2. **GMM:** we standard scale features and use GaussianMixture from Sklearn to find 15 clusters.
3. **DBSCAN:** we standard scale features and use PCA with 40 components. Then we do DBSCAN from Sklearn with epsilon=17 and minimum sample=50. We find three clusters (668 in the first, 281 in the second and 50 in the third) and 5587 outliers.

Clustering: Images - t-SNE and UMAP

We visualize data in two dimensions using t-SNE and UMAP

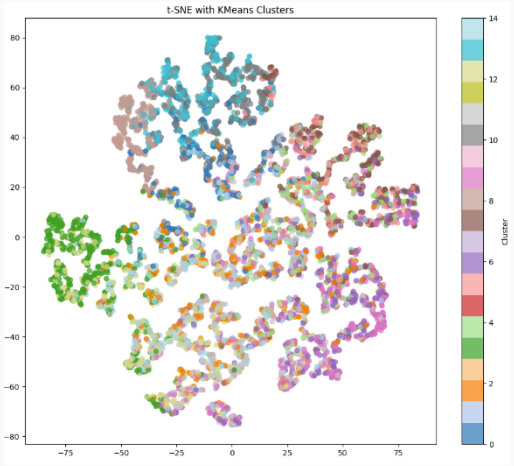
- Red observations are non-fraud while blue are fraud.
- Notice the faudulant claims are scattered, however some areas have larger proportions of fraud than other areas.

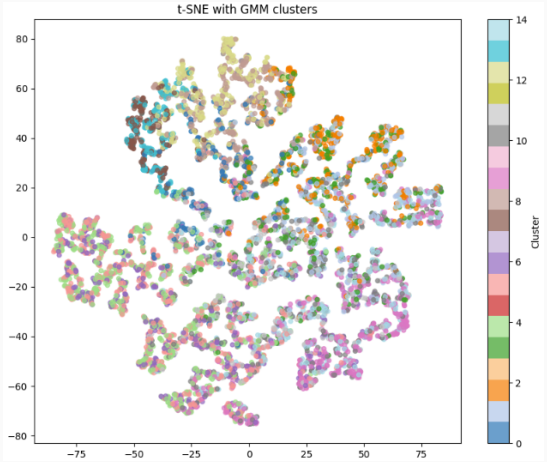


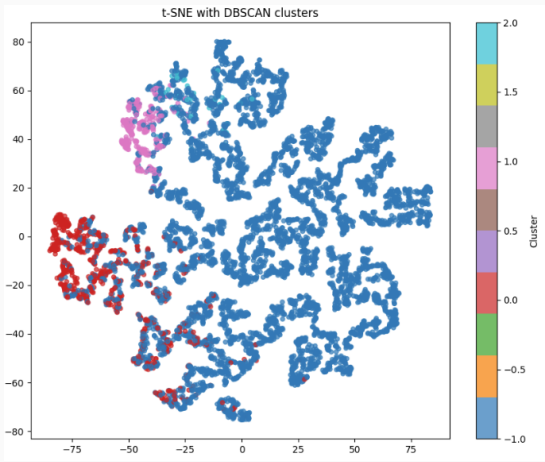


Clustering: Images - t-SNE with clusters

We visualize clusters found in two dimensions using t-SNE. To the left is KMeans, in the middle GMM and to the right is DBSCAN.

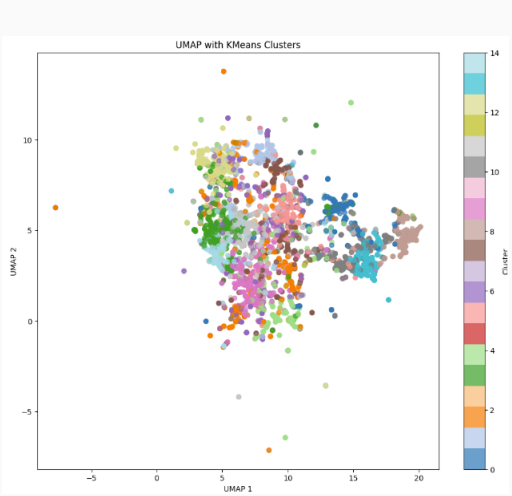


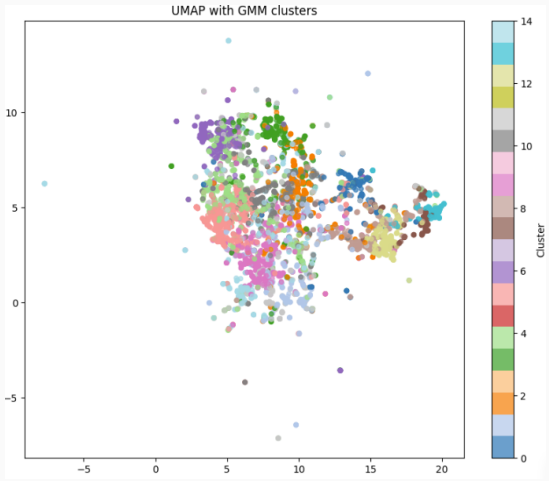


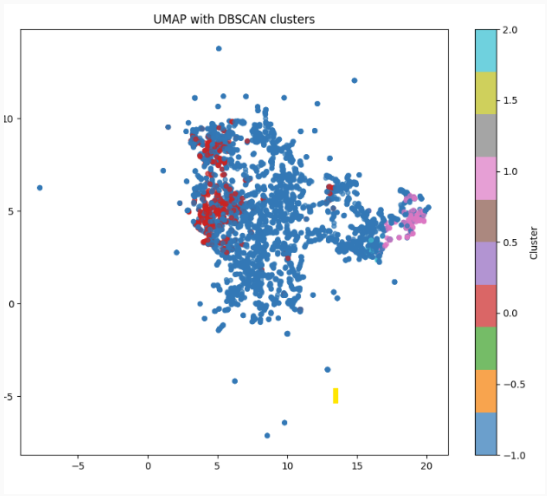


Clustering: Images - UMAP with clusters

We visualize clusters found in two dimensions using UMAP. To the left is KMeans, in the middle GMM and to the right is DBSCAN.







The DBSCAN mostly finds outliers. We choose to only investigate KMeans and GMM clustering in the following.

Clustering: Images - Inspect clusters KMeans and GMM

Clusters sorted by percentage of fraud for KMeans and GMM.

Cluster	KMeans	GMM
1	18.88	16.11
2	7.08	11.35
3	6.60	9.51
4	5.43	5.58
5	4.37	5.14
6	4.26	4.46
7	4.18	3.06
8	3.34	2.63
9	2.33	2.58
10	2.03	1.90
11	1.78	1.56
12	0.93	1.56
13	0.59	1.54
14	0.37	0.22
15	0.23	0.16

Clustering: Images - Inspect clusters KMeans

The KMeans clusters with the most and the least percentage of fraud:

18.9% fraud

0.2% fraud

Clustering: Images - Inspect clusters GMM wheels

The Gaussian mixture model clustering has split photos of wheels into two clusters:

11% fraud

2% fraud

Clustering: Conclusion

Conclusion of clustering: Both the clustering of tabular and image data showed different fraud ratios between clusters:

- **Tabular data:** The clustering of tabular data revealed that 'All perils' base policies in general have high fraud rates, and the 'liability' base policies have lower fraud rates.
- **Image data:** The clustering of images revealed that images of 'total-damaged' cars had a high rate, while images of dents and scrathes had a low rate. Also the GMM was able to capture two subgroups with different fraud ratios, within the images of wheels. These subgroups were not found by Kmeans or DBSCAN or our human eye, suggesting clustering can be used as a tool.

The clustering of images showed the greatest differences between fraud rates, indicating that this data type is prefered for clustering insurance fraud. The different fraud rates suggest that clustering can be used as an exploratory tool for fraud detection, in order to determine which groups of customers and claims the company should be aware of.

Fraud detection

Using tabular data

Problem Statement, Dataset & Strategy

Goal: Detect fraudulent car insurance claims using machine learning applied to tabular data.

Dataset: Vehicle Insurance Fraud Detection

- 15,420 insurance claims with 32 features
- Binary target: `FraudFound` (Yes/No) $\rightarrow$ encoded as 0 (No) and 1 (Yes)
- Class imbalance: only 6% of claims are fraudulent
- Mislabelled data in `Age` variable $\rightarrow$ imputation needed

Features include:

- Policy information (type, deductible, base policy)
- Demographics (age, sex, marital status)
- Accident details (area, fault, police report, witness)
- Vehicle information (make, category, price, age)

Pipeline

1. **Preprocessing**
   - Variable removal, age imputation, scaling, feature encoding
   - Class imbalance handling
   - Train / validation / test split (70% / 15% / 15%)
2. **Initial Model Comparison**
   - Compare models to identify the strongest candidates for tuning. Only the most promising candidates were carried forward to avoid unnecessary computational cost
3. **Hyperparameter Tuning**
   - Random search followed by Bayesian optimisation (Optuna)
4. **Evaluation**
   - AUC (ROC) and confusion matrix on held-out test set
5. **Interpretability**
   - SHAP values and PDP-plots for feature importance and model explainability

Preprocessing: Overview

Pipeline applied before any model training:

1. **Remove irrelevant columns:** ID variables with no predictive power and `AgeOfPolicyHolder`
2. **Feature encoding (if needed):** Ordinal and one-hot encoding of categorical variables
3. **Age imputation:** Replace mislabelled zeros using best regression model (explained later)
4. **Train / validation / test split:** 70% / 15% / 15%, stratified to preserve the fraud rate across all splits
5. **Feature scaling (if needed):** StandardScaler used for neural networks
6. **Class imbalance:** Comparison of strategies (oversampling, weights) and loss functions (LogLoss and Focal loss)

Important: All preprocessing fitted on training data only.

Preprocessing: Removed Variables

Three columns removed before modelling:

Column	Type	Reason
`PolicyNumber`	ID	Arbitrary, no predictive power
`RepNumber`	ID	Arbitrary, no predictive power
`AgeOfPolicyHolder`	Ordinal	Redundant with `Age`

Note on `AgeOfPolicyHolder`:

- This column is a binned version of `Age` (e.g. "31 to 35")
- Including both would introduce close to perfect collinearity
- `Age` (continuous/integer) carries more information than the binned version $\rightarrow$ we keep only `Age`

Preprocessing: Feature Encoding

Three encoding strategies applied:

1. **Ordinal Encoding:** Features with a natural order
   - Categories mapped to integers preserving order
   - Used for `VehiclePrice`, `AgeOfVehicle`, `Days:Policy-Accident` and others
2. **One-Hot Encoding:** Nominal categories with no natural order
   - Each category becomes a binary column, while one category is dropped as the reference (e.g. Sedan for `VehicleCategory`)
   - Used for `VehicleCategory`, `Make`, `PolicyType` and others

`VehicleCategory`	`_Sport`	`_Utility`
Sedan	0	0
Sport	1	0
Utility	0	1

3. **Numeric features:** Left unchanged

Preprocessing: Age Imputation - The Problem

Problem: The `Age` variable contains 320 observations (2% of the dataset) coded as 0. This is an implausible value indicating data entry errors or missing values encoded as zero.

Strategy: Rather than dropping these observations or imputing with a simple median, we train a regression model to predict age from the remaining features.

Models trained:

- Baseline: Linear Regression
- Tree based models: Random Forest, XGBoost, LightGBM
- NN based models: PyTorch and TensorFlow Neural Networks

Data split: 70% train / 15% validation / 15% test, using only observations where `Age` $\neq 0$. All cleaning and encoding were performed before splitting.

Preprocessing: Age Imputation - Evaluation Metric

Evaluation metric: Mean Absolute Error (MAE) defined by

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|,$$

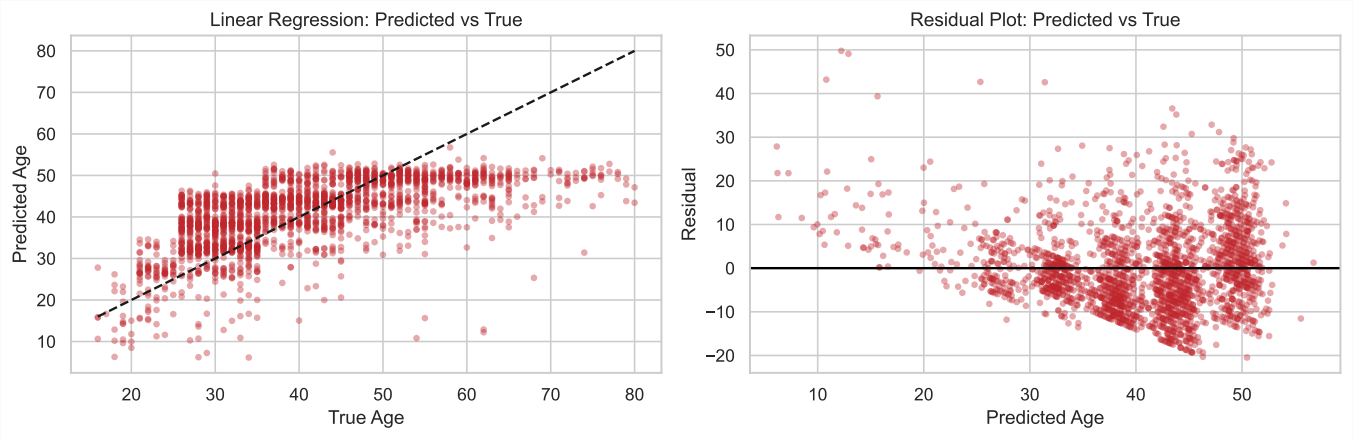
where y_i is the i'th observation and $\hat{y}_i$ that i'th prediction.

The average absolute difference between predicted and true age (measured in years).

Why MAE?

- **Interpretable:** an MAE of 8 means we are off by 8 years on average
- **Robust:** large errors are not disproportionately penalised (unlike MSE)
- **Same unit as target:** directly comparable across models

Preprocessing: Age Imputation - Linear Regression Baseline



- Validation MAE: 7.33 years, $R^2 = 0.384$ (only 38% of variance explained)
- Residual plot shows a downward trend: systematic errors across the age range
- Heteroscedasticity confirms a non-linear relationship
- **Conclusion:** Non-linear ML models are strongly motivated

Preprocessing: Age Imputation - Hyperparameter Tuning Strategy

Two-stage approach:

1. **Random Search:** broad (and fairly quick) exploration of the full parameter space
2. **Bayesian Optimisation** (Optuna)**:** focused refinement in the region identified by random search

Tree model tuned parameters (subset):

- `n_estimators`, `max_depth`
- `learning_rate` (boosted models)
- `subsample`, `colsample_bytree`

Neural networks tuned parameters:

- Number of layers
- Units per layer
- Dropout rate
- Learning rate
- Batch size

Validation strategy: All tuning decisions use the validation set only.

Preprocessing: Age Imputation - Hyperparameter Tuning Results

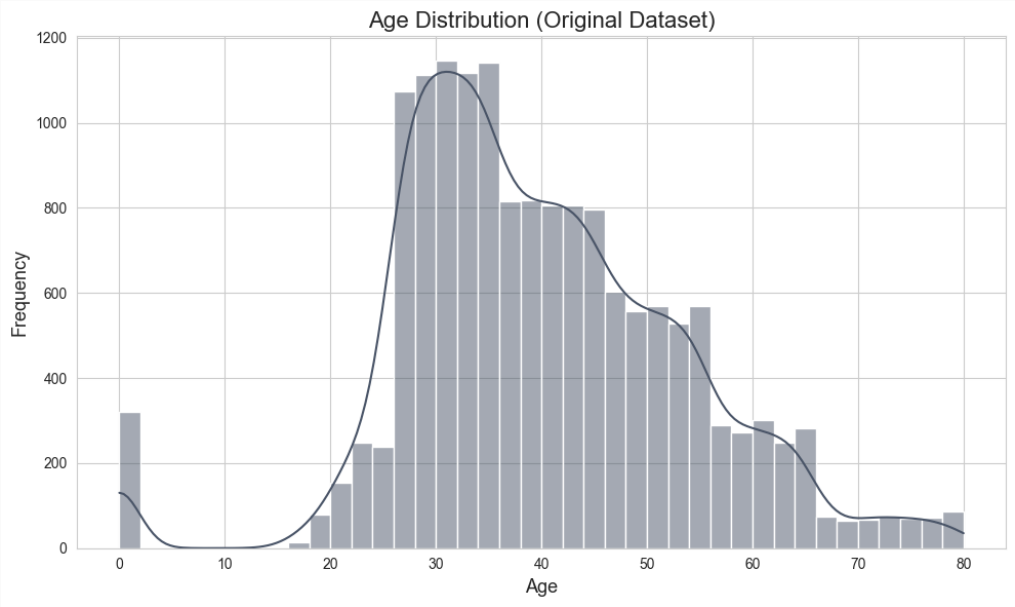
The table below shows the Bayesian optimization results for the best hyperparameter combination for each model.

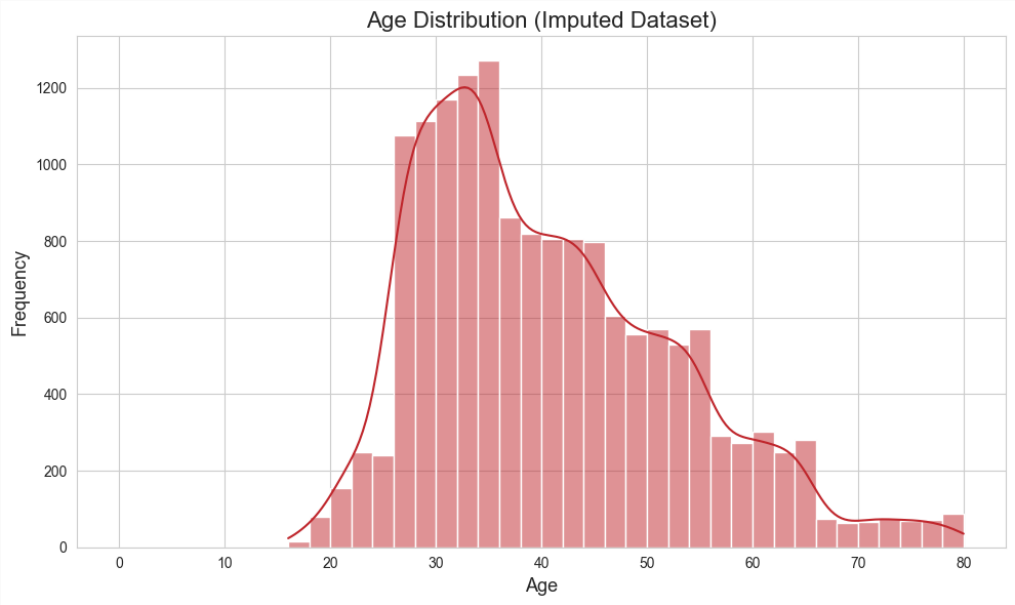
Model	Test MAE
CatBoost	6.445
LightGBM	6.465
XGBoost	6.467
Random Forest	6.496
PyTorch NN	6.522
TensorFlow NN	6.554
Linear Regression	7.330

Key observations:

- All models converge to similar MAE suggesting an information ceiling
- CatBoost (with the best found hyperparameters) achieves the best test MAE of 6.445 and is therefore selected to impute the mislabelled age values

Preprocessing: Age Imputation - Results and Considerations





Key observations:

- Age = 0 spike fully removed after imputation
- Imputed ages follow a plausible distribution centred around 30–40
- Only **2%** of data affected → minimal impact on fraud model
- `Age_imputed` carried forward to fraud classification

Preprocessing: Train / Validation / Test Split & Scaling

Split strategy: 70% train / 15% validation / 15% test, stratified to preserve the fraud rate across all splits.

- **Training set (70%):** model fitting, hyperparameter tuning, scaler fitted here only
- **Validation set (15%):** hyper parameter selection
- **Test set (15%):** final held-out evaluation (never seen during training or tuning)

Once hyperparameters are selected, models are retrained on train + validation combined for the final test set evaluation.

Feature scaling: `StandardScaler` is applied to neural network inputs only. It is fitted on the training set and applied to validation and test sets to prevent data leakage.

Important: The test set is used only once (after all modelling decisions are finalised).

Preprocessing: Class imbalance - Determining what method to use

Problem: Only around 6% of claims are fraudulent. A standard model will predominantly predict the majority class (no fraud).

Four strategies tested:

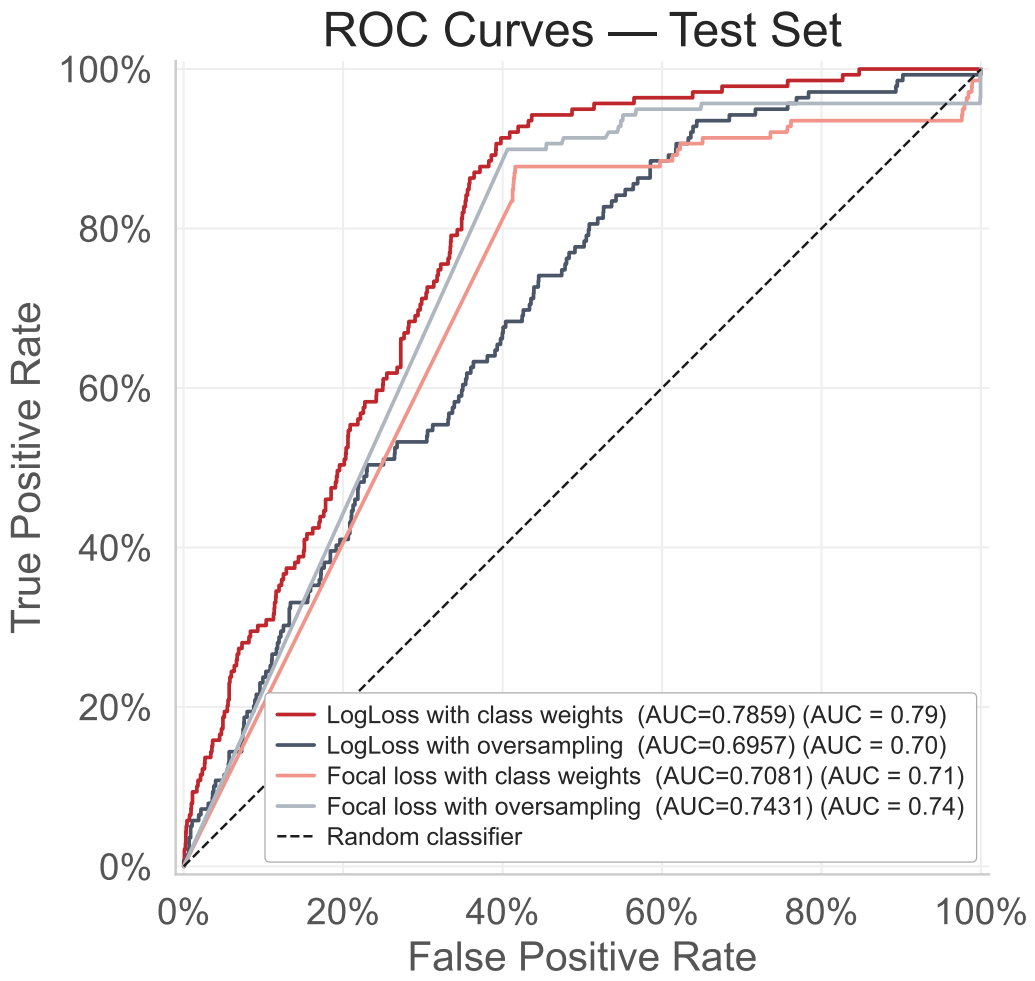
- LogLoss with class weights
- LogLoss with oversampling (fraud cases duplicated x5 to balance classes)
- Focal loss (designed to handle extreme class imbalance) with class weights
- Focal loss with oversampling

Model: All methods use CatBoost with native categorical handling (as we expect this model to perform well). Hyperparameters tuned via a 2-stage search: quick random search followed by a deeper Bayesian optimisation in the suggested region.

Preprocessing: Class Imbalance - Results

Key findings:

- LogLoss with class weights achieves the highest test AUC (0.7859)
- Oversampling improves validation AUC but does not generalise as well to the test set
- Focal loss underperforms relative to LogLoss, suggesting the fraud cases are not particularly hard to classify once reweighted



Based on the above, **LogLoss with class weights** is selected as the imbalance handling strategy for the following model comparisons.

Preprocessing: Conclusion

The following preprocessing steps were applied:

- **Removed variables:** Dropped features with no predictive value
- **Ordinal encoding:** Applied to features with a natural order (e.g. `VehiclePrice`, `AgeOfVehicle`)
- **One-hot encoding:** Applied to nominal features (e.g. `VehicleCategory`, `Make`)
- **Age imputation:** 320 zero-coded ages (2%) replaced using CatBoost regression (Val MAE $= 6.445$)
- **Train / Validation / Test split:** 70% / 15% / 15%, stratified to preserve the fraud rate across all splits
- **Scaling:** `StandardScaler` fitted on training set only, applied to neural network inputs
- **Class imbalance:** LogLoss with class weights was selected after comparison of strategies

From Preprocessing to Modelling

Preprocessing is now complete

The cleaned dataset now consists of 15,420 observations (with no missing values) and 29 features corresponding to 89 encoded features.

Modelling pipeline:

- **Baseline comparison:** five models evaluated with default hyperparameters
- **Hyperparameter tuning:** random search followed by Bayesian optimisation (Optuna)
- **Final evaluation:** best models evaluated once on the held-out test set

Loss function & evaluation metrics:

- **Loss:** log loss with class weights to penalise missed fraud cases
- **Primary metric:** AUC (ROC)
- **Secondary metric:** F1-score

Quick and Dirty: Initial Model Comparison

Linear regression motivates the use of non-linear ML models. Five baseline models trained with default hyperparameters:

- **Tree-based:** XGBoost, CatBoost, LightGBM, Random Forest
- **Neural networks:** PyTorch NN, TensorFlow NN

Validation set performance:

Model	AUC (ROC)	F1
XGBoost	0.859	0.265
CatBoost	0.856	0.289
Random Forest	0.815	0.000
TensorFlow NN	0.803	0.255
LightGBM	0.802	0.000
PyTorch NN	0.787	0.256

Note: All metrics at default threshold (0.5). Random Forest and LightGBM predict no fraud, likely due to class imbalance.

XGBoost and CatBoost highest AUC (ROC). Both outperform neural networks and are carried forward to hyperparameter tuning.

Hyperparameter Tuning: Strategy

Same two-stage strategy as used for age imputation:

1. **Random Search:** broad exploration of the parameter space to identify promising regions
    - Objective: maximise AUC (ROC) on validation set
2. **Bayesian Optimisation (Optuna):** focused refinement within the regions identified by random search
    - Search space narrowed based on random search results
    - Objective: maximise AUC (ROC) on validation set

Selected hyperparameters tuned for tree-based models:

- `n_estimators`, `max_depth`, `learning_rate`, `subsample`, `colsample_bytree` (regularisation and tree structure)
- `gamma`, `reg_alpha`, `reg_lambda` (XGBoost specific)
- `l2_leaf_reg`, `bagging_temperature`, `random_strength` (CatBoost specific)

Hyperparameter Tuning: Results

Example: Random search for XGBoost narrowed `learning_rate` to $[0.013, 0.167]$ and `max_depth` to $[3, 8]$, used for the following Bayesian search.

Bayesian optimisation results (validation AUC (ROC)):

Model	Initial AUC	Tuned AUC
Random Forest	0.815	0.847
LightGBM	0.802	0.861
XGBoost	0.859	0.868
CatBoost	0.856	0.873

- The best hyperparameters found by Bayesian optimisation are used for all follwing models
- All tuned models outperform their initial confirming that hyperparameter tuning is effective
- Models are now ready for final evaluation and comparison on the held-out test set

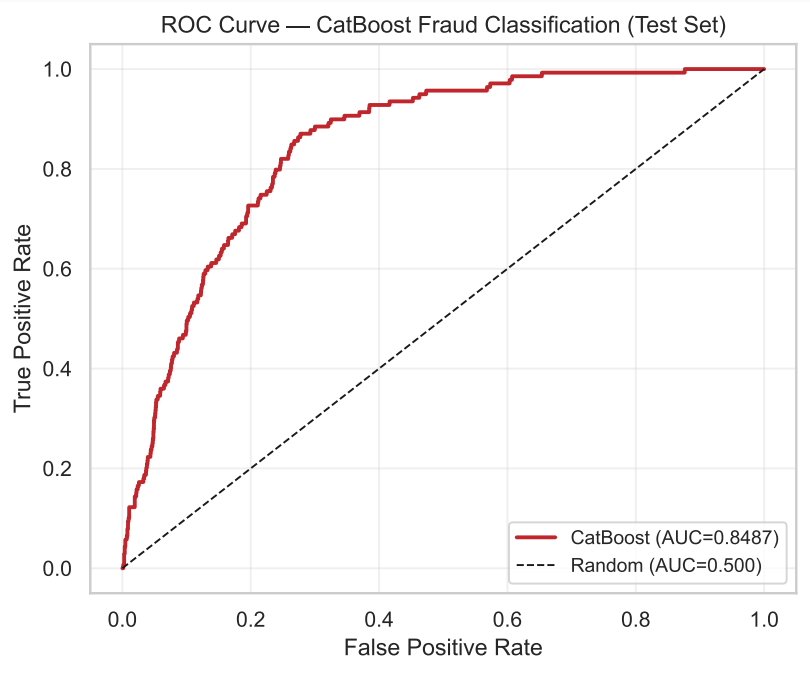
Final Test Set Evaluation

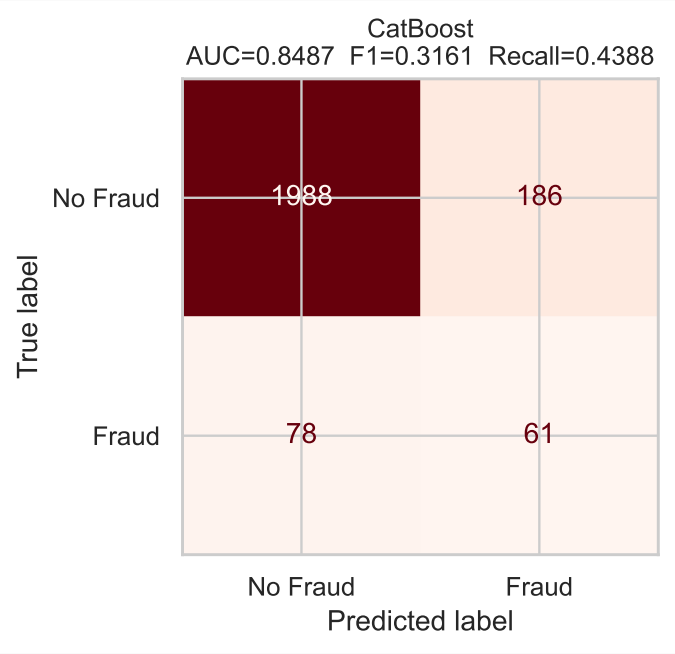
The models were retrained on the train and validation set and evaluated once on the held-out test set. The table below shows the final performance of the tree-based models:

Model	AUC (ROC)	F1
CatBoost	0.849	0.316
XGBoost	0.846	0.297
LightGBM	0.830	0.267
Random Forest	0.828	0.265

- CatBoost achieves the best AUC (0.849) and F1 (0.316) and is selected as the final model for fraud detection
- As expected, neural networks are outperformed by all tree-based models (PyTorch: AUC $= 0.791$, F1 $= 0.238$; TensorFlow: AUC $= 0.789$, F1 $= 0.213$), confirming that boosting methods are better suited for this tabular fraud dataset

Final model: CatBoost





- The AUC (ROC) is 0.849. This is well above the random baseline, indicating strong predictive power
- The figure of the confusion matrix evaluated at the default threshold of 0.5 shows that the model correctly identifies 61 fraud cases but misses 78
- The choice of threshold directly controls the trade-off between missed fraud (false negative) and false alarms (false positive)

Possible Threshold Optimisation

Why optimise the threshold? The default classification threshold of 0.5 is not necessarily optimal for imbalanced fraud detection. It controls the trade-off between:

- **Type I error** (False Positive): flagging a legitimate claim as fraud, which can be costly to investigate
- **Type II error** (False Negative): missing a fraudulent claim, which can result in the company "loosing" money

Strategy: Thresholds from 0.001 to 0.999 were evaluated on the validation set. The value that maximises the F1-score was selected. The optimal threshold was then applied to the test set.

CatBoost result: We found the optimal threshold to be 0.783, yielding AUC 0.849 on the test set.

Note: In practice, the optimal threshold is an administrative decision, since the insurance company must weigh the cost of investigating false alarms against the cost of missing fraud cases.

Explaning the Final Fraud Model: Motivation

Is a model that flags fraud but cannot say why legally and ethically insufficient?

The legal obligation (Forsikringsaftaleloven § 3b)

> **§ 3 b** Selskabets afslag på at tegne en forsikring som begæret og selskabets opsigelse af en forsikringsaftale skal efter anmodning begrundes.
>
> *Stk. 2.* Begrundelsen skal indeholde en henvisning til de relevante retsregler samt en kort redegørelse for, hvorfor forsikringen ikke kan tegnes, eller hvorfor forsikringsaftalen opsiges. Begrundelsen skal efter anmodning være skriftlig.

- Insurers must explain, upon request, why a policy is refused or cancelled
- A model prediction alone does not satisfy this requirement

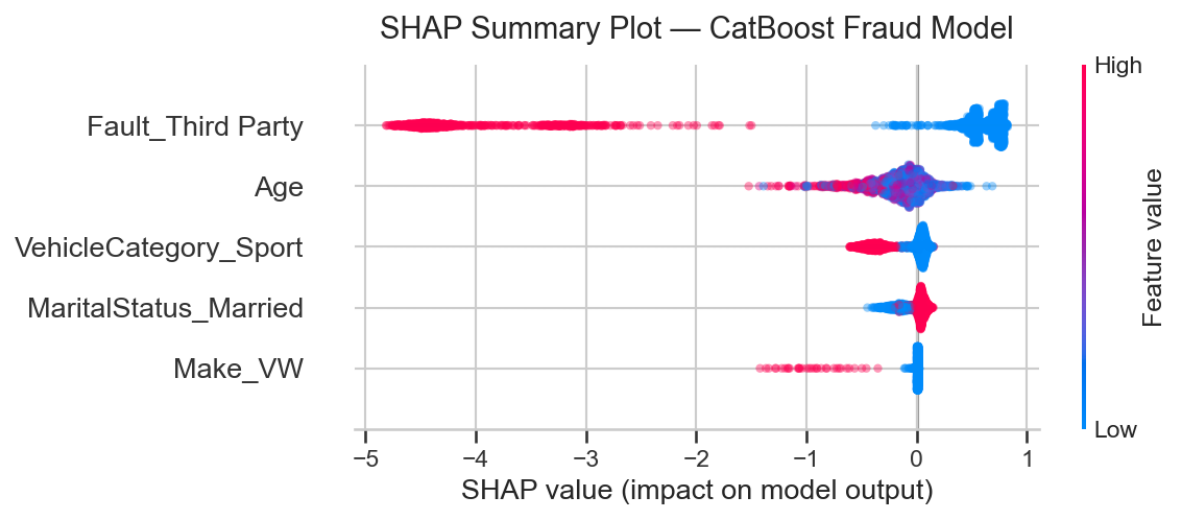
The regulatory horizon (EU AI Act (2024/1689))

- Insurance risk assessment & pricing is explicitly listed as high-risk
- High-risk systems must be interpretable by the humans overseeing them

Explainability bridges ML performance and legal accountability

Explaning the Fraud Model: SHAP Feature Importance

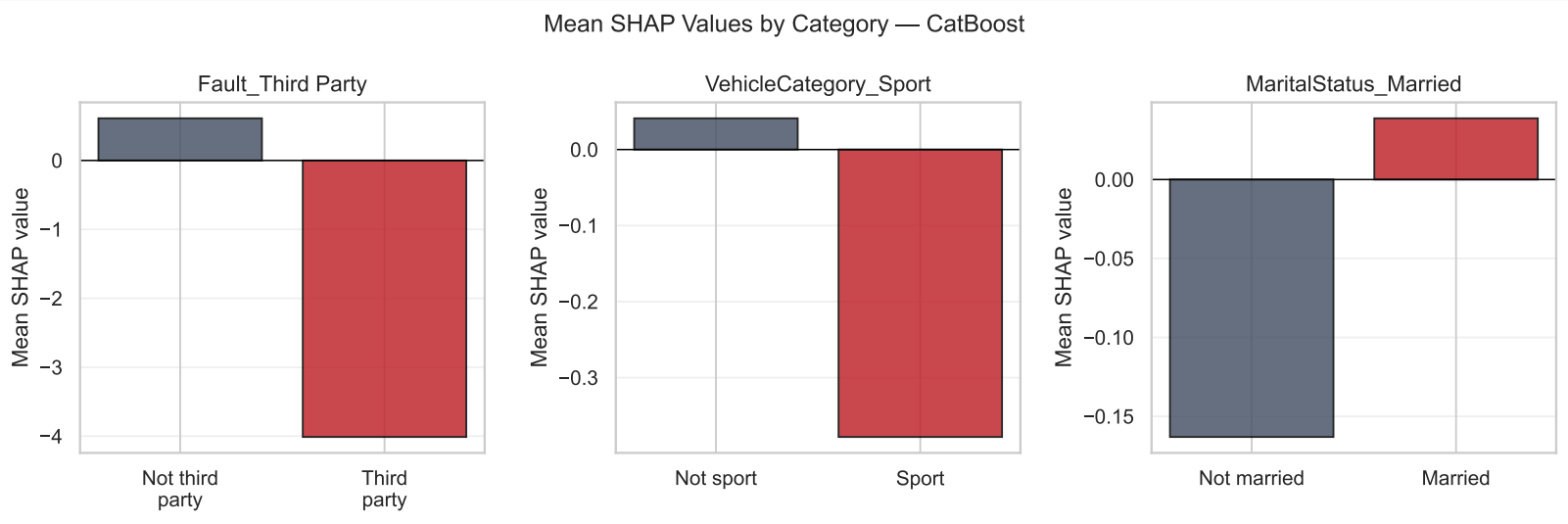
SHAP assigns each feature a contribution score per individual prediction. The plot shows selected features of interest:



- Claims where the third party is at fault (high feature value) are strongly associated with lower fraud risk compared to the policyholder being at fault. This is the most influential feature in the model
- Older policyholders tend to have a lower probability of fraud, suggesting age is a meaningful risk indicator
- Owning a sport vehicle decreases fraud risk compared to a standard passenger car (reference). Owning a Volkswagen is associated with lower risk than owning an Acura (reference), which is a luxury division of Honda
- SHAP allows explanation of individual predictions, making the model explainable under Forsikringsaftaleloven § 3b

Explaning the Fraud Model: SHAP - Categorical Features

Mean SHAP values show the average impact of each category on the predicted fraud probability. A positive value increases fraud risk, while a negative decreases it.



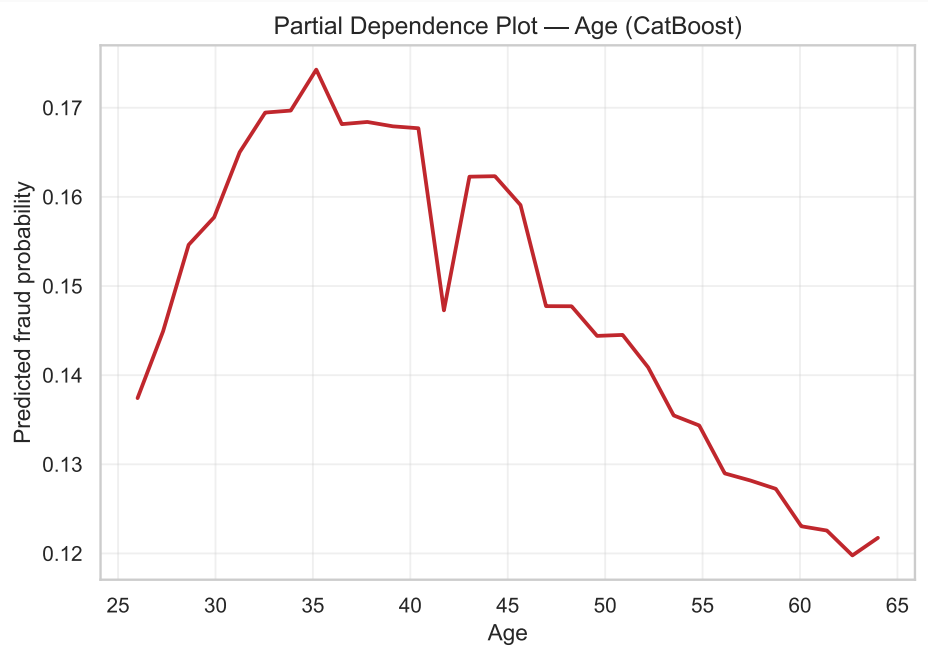
- **Fault:** claims where the policyholder is not the third party are associated with higher fraud risk consistent with the SHAP summary plot
- **VehicleCategory:** sport vehicles are associated with higher fraud risk compared to non-sport vehicles
- **MaritalStatus:** married policyholders are associated with higher fraud risk compared to non-married policyholders
- Note: values represent the mean SHAP contribution. That is, a value of zero represents no contribution relative to the model baseline

Explaning the Fraud Model: PDP of Age

A partial dependence plot (PDP) shows the marginal effect of Age on the predicted fraud probability, holding all other features constant:

$$\hat{\xi}_k(x_k) = \frac{1}{n} \sum_{i=1}^{n} \hat{m}_n(x_k, X_{i,-k})$$

where x_k is the feature value of interest, $X_{i,-k}$ are the observed values of all other features for observation i, and $\hat{m}_n$ is the trained model.



- Fraud risk peaks around age 35 meaning young adults are the highest risk group
- A notable dip around age 40-43 is visible. This could be noise. However, the data is not sparse in this range, and therefore it may reflect a genuine effect where mid-career policyholders are temporarily lower risk
- Risk steadily decreases after 45, confirming the SHAP finding that older policyholders are lower risk
- Age is informative but not the sole driver of fraud

Conclusion: Tabular Fraud Detection

What we learned

- CatBoost was found as the best model with AUC = 0.849 and F1 = 0.316 after hyperparameter tuning (with the default threshold of 0.5)
- The model successfully detects a portion of fraudulent claims but still misses the majority and generates a notable number of false alarms
- Tree-based models consistently outperformed neural networks on this structured tabular dataset
- SHAP analysis revealed interpretable patterns where fault type, age, and vehicle category are among the strongest fraud indicators
- Tabular data alone has an informatinal limit. The model is limited by the information in the claim data

Can images detect fraud that tabular data cannot capture?

In the next section we explore whether visual evidence from claim photos can "outperform" the tabular model and improve fraud detection.

Fraud detection

Using images

Problem Statement

Goal: Detect fraudulent car insurance claims using machine learning applied to labelled images.

Strategy: First we will try to build a CNN from scratch. However, we do not expect this to do particuarly well. Therefore, we will also try using tranfer learning.

Dataset: Car Insurance Fraud Detection

- **Data sample and split:** 6,584 pictures of insurance claims, labelled 'Fraud' or 'Non-fraud'. The data is split in a training set (5,170 images - 78%) and a test set (1,416 images - 22%).
- **Binary target:** Fraud / Non-fraud
- **Class imbalance:** Only 4% of claims are fraudulent.

Original image sizes

Images are ranging from 800x600 pixel to 1500x1125 pixel. The most common resolution is 876x657. In the below, is the distribution of pixels:

Building a Convolutional Neural Network without transfer learning

Pipeline:

1. Preprocess the images
2. Build CNN architecture using Tensorflow Keras.
3. Implement class weights.
4. Train CNN using 5-fold cross validation.
5. Predict on the testset.
6. Evaluate and back to step 2 (or step 1).

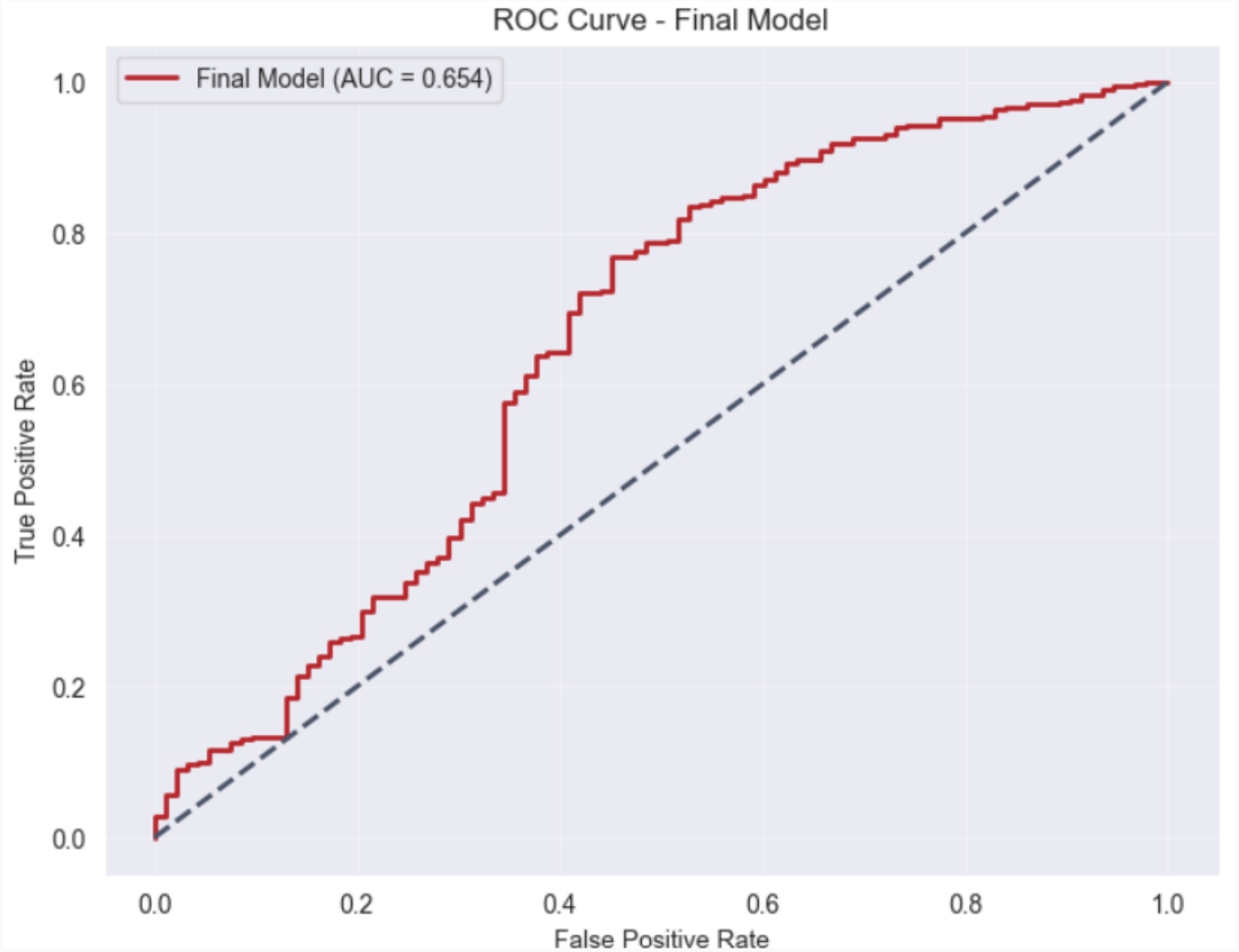
Different methods we tried

The CNN has trouble classifying fraud.
We have tried the following:

- Use (224x224) instead of (150x150) in step 1
- Adding a convolutional layer in step 2 with 256 filters
- Changing the number of filters in step 2 to, 64, 128 and 256.
- Removing a convolutional layer in step 2
- Training in five epoches in step 5
- Making 'lighter' weights in step 3
- Using cross entropy loss function in step 2

The best model

Prediction: The CNN predicts a fraud probability for each image in the testset. In the below plot is a ROC curve for the best model.



The best model: Preprocess the images

Resize: All images are resized to (150x150) pixel.

Input size: All images are in colour RGB and takes input (150, 150, 3) in the later CNN archtecture.

Normalization: The pixel values get normalized
i.e (0, 255) $\rightarrow$ (0, 1)

The best model: Build CNN architecture using Tensorflow Keras

Feature extraction:

- Three Convolution layers: First layer has 32 filters, second layer has 64 filters, third layer has 128 filters.
- Kernel size of (3x3) in all convolution layers
- Activation function ReLu in all convolution layers
- Max Pooling (2x2) after each convolution layer

Classification:

- Regulization: 40% dropout
- Flatten feature in a one-dim vector
- Dense Layer of 128 neurons and output Layer of 1 neuron

Training:

- Optimize using Adam
- Binary Focal Cross-Entropy loss and Accuracy as metric

The best model: Implement class weights

Motivation: Data is imbalanced, hence a model only predicting 'non-fraud' will have a 96% accuarcy.

Solution: Implementing class weights, penalizes false negatives (i.e. predicting 'non-fraud' when it is 'fraud')

Calculation:

$$\text{Weight}_i = \frac{\text{Number of samples}}{\text{Number of classes} \cdot \text{Number of samples in class i}}$$

Used weights:

$\text{Weight}_{Fraud} = 15.20$ and $\text{Weight}_{Non-Fraud} = 0.52$

The best model: Train CNN using 5-fold cross validation

Cross Validation:

Implement StratifiedKFold from Sklearn, with 5 splits on the trainingset. Looping through each fold, 80% is used to train the CNN and 20% is used to validate the CNN on.

Training setup:

- Three epoches per fold
- Class Weights are applied
- A new CNN model is initialized and trained for each fold

Building a CNN without transfer learning: Conclusion

Conclusion: We conclude that the model is able to predict fraud. However, it still makes quite a lot of mistakes. Next, we will see if we can use tranfer learning to improve performance.

Transfer Learning

The next thing we will try is transfer learning, where we used models that are pretrained on millions of other pictures, and then adapt it to work on our case.

- We start by making a "quick and dirty" model to see that the transfer learning works
- We improve our models by making changes one at a time and evaluate the improvement

Building CNN using tranfer learning

Preprocessing

1. **Resizing:** All images standardized and resized to 224×224 pixels
   - Default requirement for most transfer learning networks
   - Balances computational efficiency and feature retention
2. **Class imbalance:** Handled with class weights
3. **Data Split:** The training data is split into 80% training and 20% validation (recall that the test set is already its own). These sets are used for the hyperparamter optimization.
   - Keeping the same class distribution in both set
   - In the hyperparameter optimization cross-validation would be preferred. However, we have only made one split to save computational power.
4. **Optimization & Evaluation:**
   - **Loss Function:** Binary Cross-Entropy
   - **Evaluation metric:** AUC

Transfer learning: The first model

First model - Quick and Dirty

1. **Model:** We use the pretrained CNN ResNet50
   - Keeping all the layers frosen
   - Adding extra layers including output layer
2. **Optimizer:** We use ADAM as optimizer and binary cross entropy as loss function
3. **Hyperparameters:** We use standard parameters without tuning
   - Dropoutrate : 0.5
   - Learningrate: 0.1
   - Epochs : 20

Using the default threshold of 0.5 (which will be used throughout the rest of the improvements) we get the following results

AUC	True Negative	False Positive	False Negative	True Positive
0.90	1211	112	34	59

Transfer learning: Data Augmentation

Data Augmentation

1. **Data Augmentation:** We clone pictures and add different noise factors
   - Zooming in
   - Rotating slightly
   - Chance light and colorscale

   We use the same model again, but this time on more images giving us a slight improvement.

AUC	True Negative	False Positive	False Negative	True Positive
0.91	1231	92	31	62

Transfer learning: Hyperparameters

First improvement - Hyperparameters

1. **BaysianOptimizer** We use BaysianOptimization
2. **Cross Validation:** Ideally, we would do cross validation however this would take to much computational power
3. **Hyperparameters:** The best hyperparameters from the tuning was:
   - Dropoutrate : 0.3
   - Learningrate: 0.001
   - Dense Units : 256

AUC	True Negative	False Positive	False Negative	True Positive
0.93	1253	70	20	73

Transfer learning: Fine tuning

Second model - Fine tuning

1. **Unfreeze** We unfreeze the last layers but keep the first 150 layers frozen to keep the understanding of structures such as lines ect.
2. **Fine tuning** We fine tune the unfrozen layers to learn from our pictures.
3. **Low learningrate** We use learning rate 0.001 in order not to ruin the pretrained model.

We see that this is an improvement of our first model:

AUC	True Negative	False Positive	False Negative	True Positive
0.94	1267	56	18	75

Transfer learning: EfficientNet50

Third improvement - Trying EfficientNet50

1. **EfficienNet50** Instead of the pretrained CNN ResNet50 we use EfficientNet50.
2. **Redo previous steps** We repeat previous steps, i.e. data augmentation, hyperprameter tuning and tuning of last layer of the CNN for pretrained model EfficientNet50.
3. **Choice** The AUC-score is simular. However, we continue with EfficientNet50, since it is computationally faster.

AUC	True Negative	False Positive	False Negative	True Positive
0.94	1269	54	20	73

Grad-CAM

1. **Where does the model look?** We now use Grad-CAM to see which parts of the pictures are weighted highest.
2. **Problem** We see that the model sometimes use background noise such as trees in the background or the ground as information.

Ideas for Solutions

1. **Region of interest:** We used a 'You Only Look Once' (YOLO) algorithm to locate the vehicle and cut out the background.
2. **Center Crop:** We cut away the edges so the model only focus on the center since most noise are in the edges.
3. **Grey Scale:** We remove colors from the picture, such that the color of the car or colorfull items in the background does not confuse the model.

Transfer learning: EfficientNet50

Grad-CAM: Conclusion

1. **Region of interest (RoI):** The algorithm did not manage to indentify the important part of the pictures succesfully.
2. **Center Crop (CC):** Simple, but seemed to be working best
3. **Grey Scale (GS):** Did not seem to impact the model.

Model	AUC	True Negative	False Positive	False Negative	True Positive
RoI	0.94	1278	45	23	70
CC	0.96	1310	13	24	69
GS	0.90	1201	122	27	66

After Center Crop

Before

After

We clearly see that there is a significant improvement and that our

Transfer learning: Focal loss function

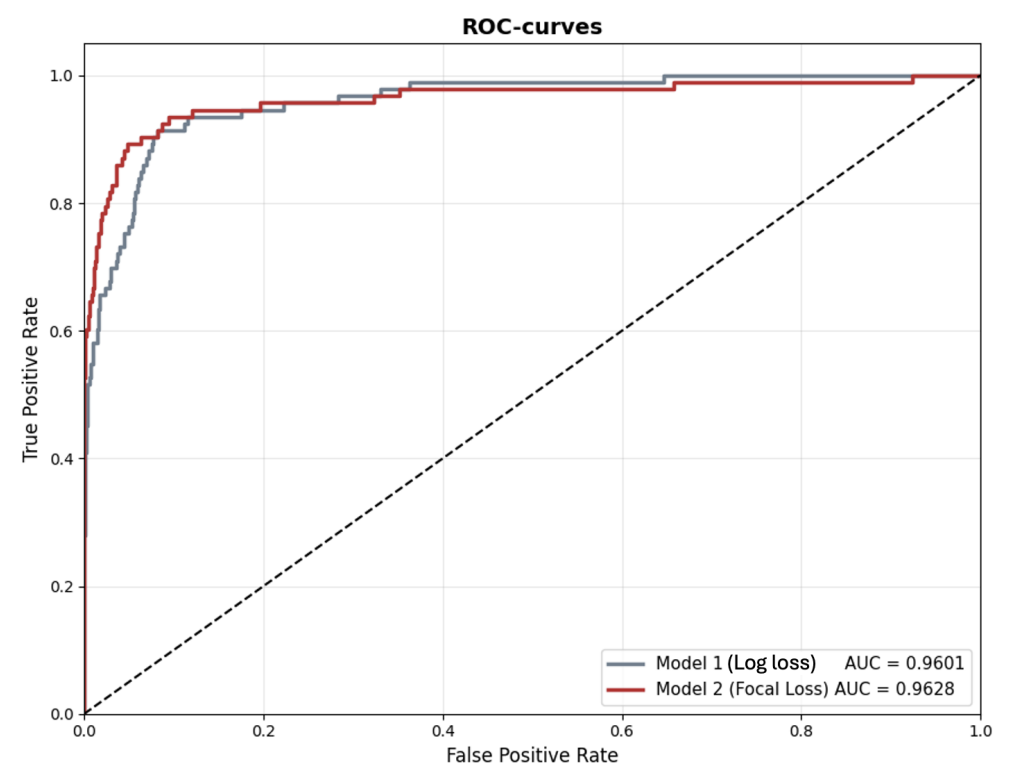
Fourth improvement - Focal loss function

1. **Change of Loss function:** We change from Cross Entropy to Focal loss function.
2. **Focus on False Negatives:** This decreases the number of false negatives but at the cost of false positives (at the default threshold).

AUC	True Negative	False Positive	False Negative	True Positive
0.96	1260	63	11	82

Note: If we force the previous model with Cross Entropy to have only 11 false negatives by changing the threshold, it will have 95 False Positives.

Transfer learning: Choice of threshold



Both models demonstrate excellent predictive performance, but the Focal Loss model (AUC = 0.9628) marginally outperforms the standard LogLoss model (AUC = 0.9601). Notably, Focal Loss achieves a slightly higher True Positive Rate at low False Positive Rates, making it the preferred choice for imbalanced datasets.

Transfer learning: Conclusion

Conclusion

- **Transfer learning outperforms the baseline:** Moving from a custom CNN built from scratch to a pre-trained architecture significantly improved fraud detection accuracy.
- **Significant improvement** We were able to make significant improvements to the model. Especially the Grad-CAM analysis made improvements.

Overall Conclusion

- **Best approach?** The model yielding the best results is the image-based model utilizing transfer learning.
- **Are images better than tabular data?** Since even our worst image-based models seem to capture more information, it indicates that images are superior for detecting fraud.
- **How would we test this?** We could project both images and tabular data into latent spaces of the same dimension and use an identical model architecture to predict on these embeddings. This isolates the informational difference between the two data types.
- **What could we conclude?** This setup would allow us to strictly conclude whether the images contain more predictive information than the tabular data. However, this conclusion would only be valid for these specific dataset.