AI for predicting insurance frauds

Gregori 5410971

Academic Year 2024/25

Evidence and supporting studies show that insurance fraud cost the U.S. economy $308.6 billion annually*, and 20% of the claims are suspected to be fraudulent**.

Problem statement

The designed tool is oriented at helping insurance companies in **detect fraudulent claims**, aiming at reducing the amount of resources needed for checking the deposition made by clients.

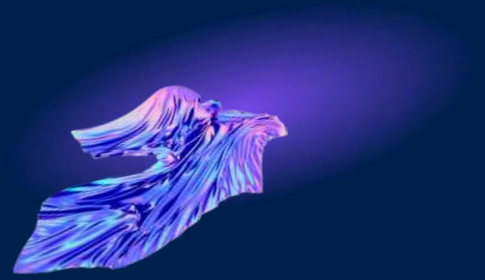
Moreover, the model aims at reducing both false negative and false positive cases, ensuring more accurate fraud detection with fewer unnecessary claim rejections.

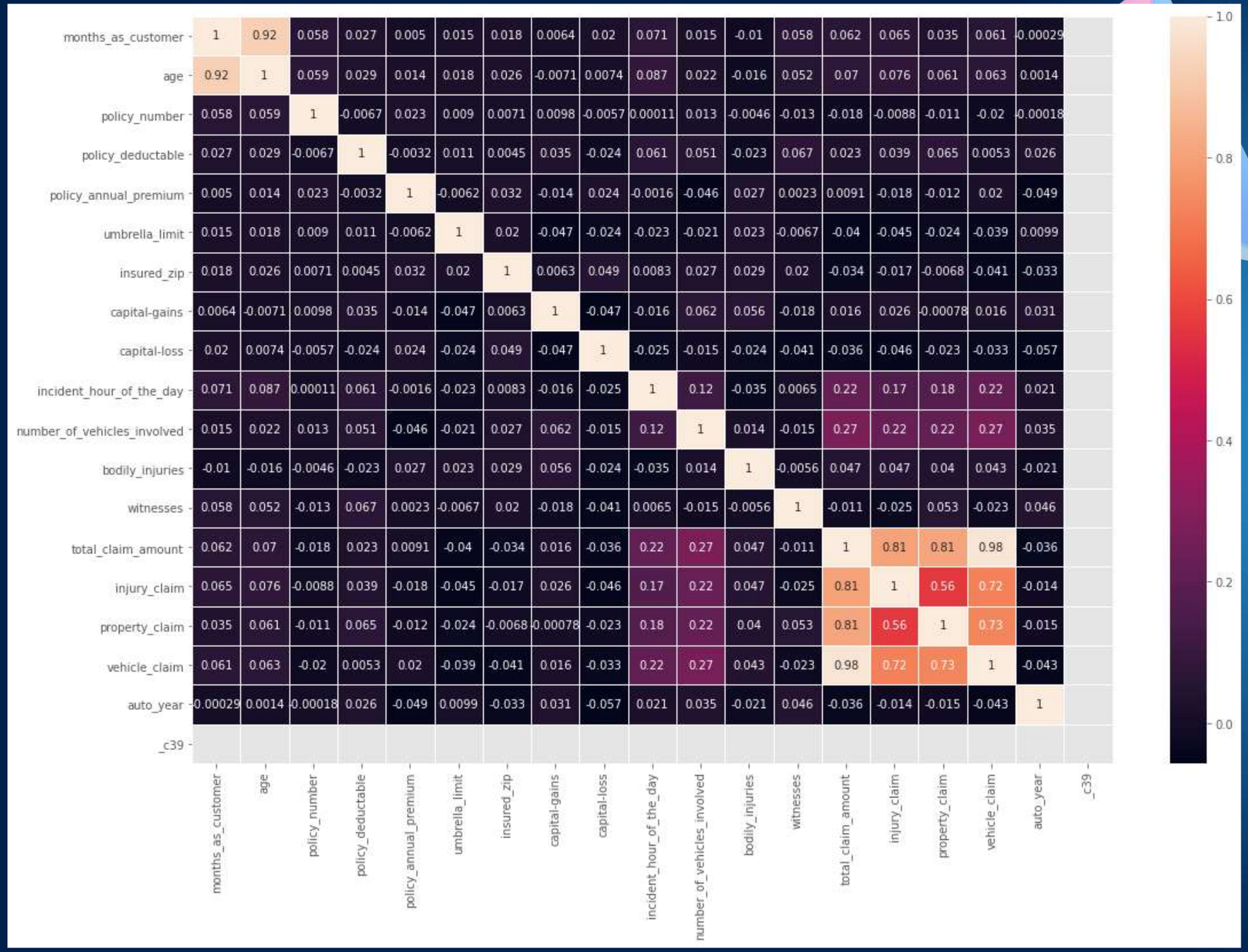
source: * https://www.iii.org/fact-statistic/facts-and-statistics-insurance-fraud
** https://www.friss.com/insurance-fraud-report-2022

As visible from the reported graph, there are some variables that are **highly correlated** with each other, i.e. the age of customers and the time since they are insured.

Another example of visible correlation is the impact that claims regarding, e.g., vehicles have on the total claim (98%).

Since the heatmap reveals that some variables have a high correlation coefficient, we can analyze the data from the claims presented by customers and **assess the likelihood of fraud**.





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

Data gathering

Which data do we need and how to get them

There is no evidence of a perfect correlation (= 1) between any kind of variable and the claim being a fraud.

Therefore, it is necessary to collect **different types of information** in order for the model to predict, with the best accuracy possible, whether there is an ongoing fraudulent action.

Some variables show less to no linking to the truthfulness of the statement provided.

For the model to work, we need to gather data that shows signs of **differentiability** regarding the compliance of the claim.

Some of the **parameters** that represent a differential factors, hence are worth considering, are the following:
-the **insurance history** of both the vehicle and the subject of the claim,
-the **matching** between the damage declared and the reported incident dynamics,
-the **relationship** between the parties
-the **insurance data** in general (was the policy activated shortly before the claim? Does the customer do frequently policy changes or upgrades?).

The sources of data gathering are **both internal and external**. Some information comes internally from the insurance company, i.e. from internal databases.

Also public and external information has to be gathered and consulted, e.g. the police report of the accident or government vehicle records.

Database of choice

The data collected for fraud prediction includes **both structured and unstructured information**.

To fully leverage the strengths of each data type, the model integrates **two different types of databases**, ensuring optimal utilization of the available features.

Structured data are stored in a **relational database**, allowing the model to detect insights and guaranteeing referential integrity.

Nevertheless, since the aim of the model is also identifying connections between data that are unstructured, there is also a **blob storage**, more suitable for unstructured data and characterized by great scalability.

It is important to underline that to train the model effectively, we need to consolidate data from both databases into a **single unified database**.

This will ensure that both structured and unstructured data are properly integrated, allowing the AI model to leverage all available features for accurate fraud detection.

The tree-based model and why we chose it

The model we chose is a **tree-based model**. Trees are great at capturing complex relationships (e.g. non-linear relationships and interactions between variables) and are easy to interpret and to explain.

Moreover, having in the database both categorical and numerical data, tree-based models are suitable since they natively support both types of variable without requiring encoding.

Our **target variable** is whether the claim is a fraud (labelled with 1) or not (labelled with 0).

In order to train the set we should separate the features (that can be seen as the inputs of the model) from our target variable, that represents the **outcome** we would like to acquire.

```
[4]  #defining the features (x) and the target variable (y)
     X = df.drop(columns=['fraud_label', 'claim_id', 'client_id', 'policy_id'])
     y = df['fraud_label']

     #dividing the dataset in training (70%) and testing (30%)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

     #verifying the fair distribution of the classes
     print("Class distribution in the training set:")
     print(y_train.value_counts(normalize=True))

     Class distribution in the training set:
     fraud_label
     0    0.893714
     1    0.106286
     Name: proportion, dtype: float64
```

This piece of code and all the following are extracted from:
https://colab.research.google.com/drive/1Oo7LfY7TsKYplKwkqnvebVXZos4f0CUz#scrollTo=cXVhipRmVP9e

High variance: a strong limit of tree-based models

Tree models are characterized by low bias (so they are capable of capturing data well) but a **large variance**, meaning that the models are very sensitive to changes in the dataset.

The model therefore adapts too much to the training data, causing what is called '**overfitting**'.

Simplifying, there is the chance that the model fits optimally the training data, but the performances on new data are poor.

In order to achieve the best results possible, we should find the **right balance** between bias and variance to obtain the right level of generalization.

To reduce variance, it should be used the **Random Forest algorithm**.
It is an ensemble training method that combines multiple decisions trees to improve accuracy and diminish overfitting.

Via **bootstrap sampling**, Random Forest builds multiple decision trees using different random subsets of data, and trains each decision tree independently from its subset.

To train each tree independently (on different features) this method uses **bagging**.
Then it **aggregates the prediction** from each tree, for classification (using the majority of the results) or for regression (computing the average of tree prediction).

Random forest

```
#definying the hyperparameters to test
param_grid = {
    'n_estimators': [50,70],
    'max_depth': [5,7, 9],
    'min_samples_split': [5, 10, 15],
    'min_samples_leaf': [2, 5, 10],
    'max_features': ['sqrt', 'log2'],
    'bootstrap': [True, False]
}

#creating the model with balanced classes
rf = RandomForestClassifier(random_state=42, class_weight="balanced")

# Grid Search to find the best hyperparameters
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)

#training the model with the best hyperparameters
grid_search.fit(X_train, y_train)

#best hyperparameters found
print("Best hyperparameters:", grid_search.best_params_)

#optimized model
best_rf = grid_search.best_estimator_
```

```
Fitting 5 folds for each of 216 candidates, totalling 1080 fits
Best hyperparameters: {'bootstrap': True, 'max_depth': 9, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 70}
```

The desired outcomes

Since we are talking about a classification model, the desired outcome is a prevision about whether an insurance claim is fraudulent. The primary output is a **boolean value** that returns 0 (no fraud) or 1 (fraud).

```
[ ]  y_pred = best_rf. predict (X_test)
     print(y_pred [:50])
     #this is the output of the model, revealing that among the first 50 cases 5 cases are identified as frauds (value 1)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 0 0]
```

```
from sklearn.metrics import accuracy_score

#prediction on the test set
y_pred = best_rf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Model accuracy: {accuracy:.4f}")
```

```
Model accuracy: 0.8573
```

It is important to underline that, when **checking the accuracy** of the model, the value returned should not be 1, otherwise, it's a clear sign of overfitting. A well-generalized model should produce probability distributions that reflect real uncertainty, rather than making extreme, overconfident predictions.

FP and FN

An intrinsic limit of every AI solution, including the one presented in this project, is the presence of cases of False Positive (FP) and False Negative (FN).

These are two different error that needs to be balanced in different ways, accordingly to the scope of the AI solution.

Are called **False Positive** the cases in which a legitimate claim is identified as a fraud, causing additional costs for the insurance company and frustration for the customer.

The **precision** of the system is the metric that measures the model's ability to avoid false positives, ensuring that flagged fraudulent cases are actually fraud.
With an higher precision the system is more conservative.

A **False Negative** occurs when the model incorrectly classifies a fraudulent claim as legitimate, leading to financial losses for the company.

The **sensitivity** (recall) is the parameter that measures the ability of the system to detect fraud cases, making, if increased, the system more aggressive.

It is hence fundamental, for a well-performing AI solution, to **balance the trade-off** between precision and sensitivity.

Feasibility of the AI solution

Being much of the **data** needed for the solution to predict frauds internal to the insurance company, the data collection is feasible.
On the other hand, also external sources are needed, and this might lead to difficulties in the integration process.

A potential issue arising from the adoption of the solution is the initial **resistance to change**.
This can be addressed communicating with the end-users and providing adequate training in order to allow the employees to integrate the solution into their workflow.

The implementation of the system is expensive and resource consuming, but in the medium term proves to be **cost-effective**.
In fact, the initial investment is one-shot and it enables long-term savings by reducing the time and resources needed for manual fraud detection.

Moreover, the designed solution is **in compliance** with the regulatory framework and also shows **high verifiability**, making the model reliable and minimizing the risk of biased or incorrect decisions.
According to the EU AI Act*, the model can be considered a **high-risk AI system**. Therefore, both the provider and the supplier must meet stringent requirements:
- ensure **system accuracy**, robustness and cyber security;
- guarantee the **human oversight** over the outputs;
- protect the **data quality** and governance
- provide comprehensive **technical documentation** and **transparent information** to the deployers.

* sources:
https://artificialintelligenceact.eu/article/6/
https://artificialintelligenceact.eu/annex/3/

From a business perspective...

Fraud detection software are **not yet widely adopted**: a survey showed that, whereas 100% of the insurance professionals have mechanisms to identify potentially fraudulent claims, only the 62% of them uses a fraud detection software*.

Surely these solutions present **defects** (i.e. the scarcity of IT resources or the difficult of measuring important metrics as the Return On Investment).
Moreover, there might be a generalized skepticism regarding Artificial Intelligence, especially in non-innovative traditional companies.

On the contrary, a fraud detection software can lead to **multiple and increasing benefits**.
Among many advantages, the most remarkable ones are:
- increased analytics,
- improved loss ratio,
- better investigation efficiency,
- capability of staying ahead of fraud schemes.

*source: https://www.friss.com/insurance-fraud-report-2022

Deployment and roadmap

Phase 1: Model Development and Training (month 1-4)

During the first months of the project, there is the collecting and analyzing of historical insurance data, to perform the fundamental step of exploratory analysis.
Define, moreover, the evaluation metrics of the models and optimize the hyperparameters to balance accuracy, precision and recall.
Eventually, validate the models via cross-validation and the predefined metrics.

Phase 2: System Integration and Testing (UAT, month 5-6)

Develop and integrate the AI model into the traditional business processes of the insurance company. Ensure the compliance with the AI regulatory framework and collect users' feedbacks to improve and verify the user acceptance.

Phase 3: Deployment and Monitoring (month 7+)

Full adoption of the AI fraud detection system. After the deployment, continuous monitoring of the performance are necessary. Implement feedback mechanisms to refine predictions and improve efficiency over time.