

MASTER THESIS

Title: Automatic Machine Learning for Insurance: H2O Experiment

Author: Samuel Valle Nofuentes

Advisor: Dr. Salvador Torra Porrás

Academic year: 2020-2021

Faculty of Economics and Business

Universitat de Barcelona

Master thesis

Master in Actuarial and Financial Sciences

**Automatic Machine
Learning for Insurance: H2O
Experiment**

Author: Samuel Valle Nofuentes

Advisor: Dr. Salvador Torra Porras

The content of this document is the sole responsibility of the author, who declares that he has not incurred plagiarism and that all references to other authors have been expressed in the text.

Summary

This thesis provides an introduction of machine learning (ML), shows the implication that ML has on the insurance sector and takes a special consideration to the H2O ensemble modelling approach for the insurance claim fraud detection binary classification. The aim of this thesis is to study the H2O Automatic ML potential and compare the results generated with traditional algorithms such as lineal perceptron, Logistic Regression, multilayer perceptron, support vector machine and decision tree. Using H2O web interface or R programming, not only the most efficient ML algorithms are obtained with no effort but also provide better modelling metrics than traditional methods.

Key words

Insurance, fraud, AutoML, binary classification, machine learning.

Acknowledgments

I take advantage of this section to thank my parents for the trust given, the support of my wife, the wisdom provided by the professors of the master's degree in actuarial and financial sciences at the Universitat de Barcelona, as well as the perseverance of my advisor who had the idea of developing this theme.

Contents

1. Introduction	1
1.1. Motivation.....	1
1.2. Objectives	1
1.3. Structure.....	1
2. The insurance industry	2
2.1. Data and Risk.....	2
2.2. Machine Learning	3
3. Conventional Machine Learning.....	5
3.1. Brief History	5
3.2. ML basics.....	5
3.2.1. Supervised learning.....	6
3.2.2. Unsupervised learning.....	6
3.2.3. Reinforcement learning	7
3.2.4. Ensemble methodology	7
3.3. Insurance practices.....	7
4. Automatic Machine Learning.....	11
4.1. AutoML Basics	11
4.2. AutoML systems available	12
4.3. H2O's AutoML.....	12
5. Case: Fraud detection.....	14
5.1. Data analysis with R	14
5.2. H2O AutoML with R.....	17
5.3. H2O AutoML with Flow	23
5.4. Linear model - Perceptron with R.....	24
5.5. Linear model - Logistic regression with R	24
5.6. Neural Network – Multilayer perceptron with R.....	27
5.7. Support Vector Machine with R	29
5.8. Decision trees with R.....	30
6. Results	32
7. Conclusions	34
8. Annex A - Images and Outputs	35
9. Bibliography.....	53
10. Annex B - Project's R code.....	59

1. Introduction

1.1. Motivation

In insurance, data scientist and actuaries, the former developing models to process data and the latter working with existent models to quantify risks, know how to make good use of data. Conventional curve-fitting models work as a reference approach for risk analysis and prediction. Although the assumptions in which these models rely on may find relationships between variables, these only provide divided and limited targeting productive capacity.

Nowadays, enhancements in computer processing power, evolution of hosted service providers over the Internet and large amount of data available and stored by firms have caused the development of new techniques based on ML.

Insurers using ML are already obtaining beneficial results such as quicker claim settlement, more targeted sales and fraud detection, among other advantages. However, it is still too soon to widely apply ML in many procedures due to data availability and its poor quality. But most importantly, the performance of ML models is sensitive to an excess of design and expert judgment configuration, being a significant barrier for inexperienced users. Automatic ML (AutoML) has the objective of making these decisions automatically and choosing the best ML method based on the data used.

1.2. Objectives

Several tech companies do currently offer expensive AutoML services. The H2O open-source platform is used in this master thesis. The H2O platform allows building ML models without any technical cost through AutoML methods, with R programming or with its own web platform, that require very few parameters to make it work.

This paper intends to clarify if AutoML functionality is applicable to motivated but inexperienced users, also considering if its outcome is more promising than conventional methods in order to become an accessible solution for the insurance sector.

1.3. Structure

This paper is organised as follows: section 2 covers the current insurance industry situation with respect to data needed to asses risks, section 3 briefly introduces the conventional ML world and section 4 explains the AutoML capabilities with the particularity of H2O AutoML. To finalise, section 5 details the binary classification practical case on fraud detection with car insurance claim data, section 6 shows the results of the practical case and section 7 provides the conclusions of this master thesis.

2. The insurance industry

2.1. Data and Risk

Even in the most remote cases, insurers have needed, need and will need data to be able to predict future claims in exchange for setting an appropriate price to cover that risk. Technology evolves and the insurance sector with it. An example of this fact comes from Beam (Globe Newswire, 2021), a North American dental insurer that generates concern in some of its customers by deciding to send them, as part of a package of benefits included in their policy, a connected electric toothbrush. Clients must use the device in combination with an app installed on their smartphones which transmits the data of their oral hygiene habits to the company. Beam company informs that data collected are not shared but the public opinion might believe that the tooth brushing style is too private even to be shared with the insurer. Well, in insurance, as long as that helps to adjust the price down, time will end up saying if the market accepts such data transferring.

People here in Europe may keep calm with respect to user data protection as the European Commission is already looking into the Artificial intelligence (AI) legal framework to address fundamental rights and safety (European Commission, 2021). But, what about black boxes and sensors in vehicles collecting GPS positions and places visited? According to Research and Markets (2021), the insurance telematic market is already implemented in Asia-Pacific countries, such as Japan and China, Europe, the United Kingdom, North America and many other regions in the world. Consequently, tracking driving habits prevents false insurance claims and drives growth in the insurance sector.

The insurance industry seems to push customers towards a more connected world with the unique purpose of obtaining data to assess risk more accurately. Insurers protect themselves from the risks they back by setting provisions. However, they are not protected against fraud risk and need to learn how to mitigate such offence. The COVID 19 pandemic, which human beings are currently living worldwide, has triggered the number of fraud cases in the Spain's insurance business to record levels over the last decade (AXA, 2021), reaching 24000 fraud cases detected in 2020 and avoiding paying 67 million euros. Furthermore, the Cooperative Investigation between Insurance Entities and Pension Funds (ICEA, 2021) informs that the average amount of fraudulent claims avoided by insurers was 2287 euros in 2020.

The strengthening of the company's detection capabilities of fraudulent claims has been increased with investment in resources and specific technological tools. In a blink of an eye, computing power has increased exponentially and there is a wealth of data available to get processed. One of the protagonist of this thesis, ML, has been able to take advantage of the situation with Big data, removing the focus on knowledge to give it to data-driven processes. However, it is not optional but essential for some insurers to embrace digital technology in order to cope with the growing volatility of environmental catastrophes and damages. Complex risks to insure, such as climate change, requires more sophisticated Property and Casualty risk models, enabled by AI and analytics (Accenture, 2021).

2.2. Machine Learning

ML, a subset of AI, focuses on the ability of machines to receive a set of data and learn by themselves, adapting algorithms as they learn more about the information they process. Its applicability has primarily been limited in the past due to the need for large volume of big data and computational power to deliver robust results. Nowadays, shortage of computer system investment and qualified data scientist limit the development of this technology. According to Malhotra, R and Sharma, S (2018), many insurers process between 10% and 15% of the data they have access to, most of which is structured data they house in traditional databases. That means they are not only failing to unlock value from their structured data, but also overlooking the valuable insights hidden in their unstructured data. Analysing unstructured or semi-structured data and using it to drive better business decisions requires advanced data science techniques. Emerging data analytics technologies centred on ML bring order and purpose to semi-structured data so that it can be more effectively mined for business insights.

As evidenced from a recent London based financial technology website article (Tech Bullion PR, 2021), insurance firms can automate various manual activities with AI and ML, driving higher customer loyalty and providing their services quicker.

According to INESE (2021), a pioneering solution developed by IBM and MAPFRE based on AI named Verbatims will change the experience and communication with the insured. Insurance firms turn to AI companies to provide specialised solutions, including records management, virtual agents and intelligent analytical systems. These firms will use specialised tools to achieve lower operational costs and better consumer service with faster claim processing, customised pricing and fraud identification.

The research from a UK financial sector provider of market intelligence (Juniper Research, 2018) found that the introduction of AI in the insurance claim process will generate significant cost savings. Juniper forecast that across property, health, life and motor insurance, the annual cost savings will exceed 1.2 billion dollars by 2023.

According to TechBullion PR (2021), the benefits of AI in the insurance sector are:

- **Faster claim processing:** AI can handle insurance claim data in an easier, simpler and faster way, generating a wider data analysis, which becomes in a quicker policyholder service to gain customer loyalty and retention.
- **Fraud detection:** AI enables real-time claim fraud detection that human adjuster cannot provide, and hence, reducing claim settlements and costs.
- **Loss prevention:** The integration of data from multiple sources, such as tablets, mobile phones, cars, home appliances, health trackers, fingerprint readers and other telematics devices, makes more data available to insurance industry nowadays. The

greater the use of such devices the higher the volume of customer data stored, which allows insurance companies to assess and model claims behaviour and risk levels more accurately, avoiding excessive costs and increasing profitability. In addition, insurance firms can use data for risk evaluation during the underwriting process and preventive promotion.

- **Better customer experience:** Automatic virtual assistants interact with policyholders over the phone and through online applications, setting solutions to clients' complaints and collecting additional information, accelerating incident management, providing customer advice, predicting customer leakage and calculating insurance price instantly.

The whole world seems to be adapting to the new Big Data era, as reported by Google Trends (2021) which informs that the demand for AI for insurance has tripled in the United States since 2012.

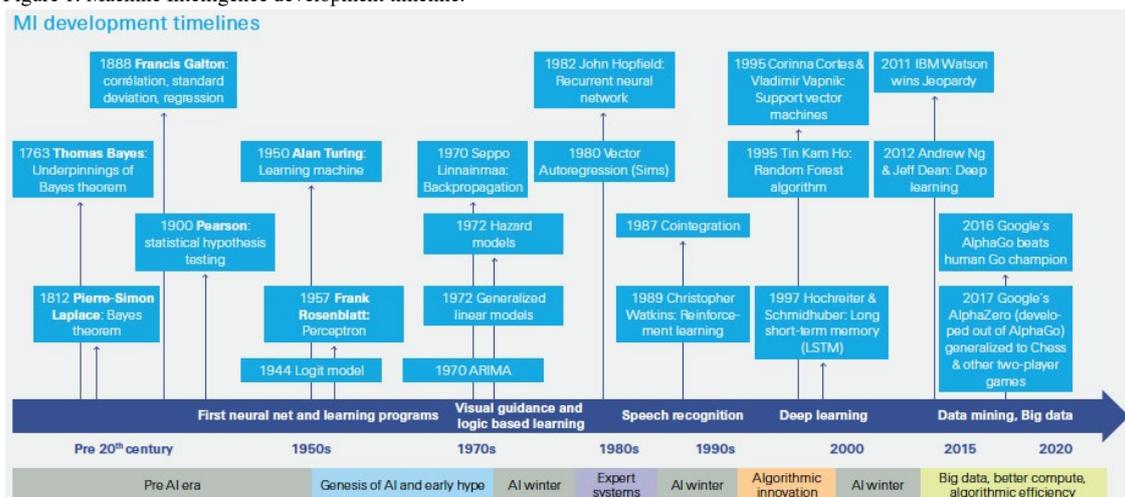
In addition, according to the research for the Accenture Technology Vision 2018 (Accenture, 2018) details that more than 90 percent of insurance firms are using, plan to use or are considering using machine learning or AI in the claims or underwriting process.

3. Conventional Machine Learning

3.1. Brief History

From the Turing test in 1950 to the 21st century object and speech recognition, AI and ML have evolved radically over the last 70 years. According to Gartner (2016) ML is among the technologies at the peak of inflated expectations, looking forward to high degree of competitive advantage over the coming 10 years. A brief timeline with the machine intelligence history can be seen in Figure 1:

Figure 1: Machine Intelligence development timeline.



Source: Swiss Re Institute (2020)

3.2. ML basics

ML is a scientific discipline from the AI field that studies the computer capacity to learn from large data sets without being programmed explicitly. This automatic learning methodology uses algorithms that receive and analyse input data in order to identify and learn from patterns found in such data. Human observation can easily miss these patterns and associations and, due to the analysis of complex, inter-related and non-linear relationships among the input data variables, automatic decisions or predictions can be made by ML once algorithms are trained with data. After supplying data to predictive models, the algorithms will forecast based on data provided. As new and more diverse data is fed into ML algorithms, these learn and optimise their operations to improve performance, developing intelligence over time.

Similarly, pre-schoolers sometimes learn new vocabulary through trial and error. The mistake of recognising a horse when it is actually a giraffe will be corrected by parents or teachers. Consequently, their brains will learn to identify giraffes after generating patterns, for instance, with the animal's physical features such as height and skin colour.

As there is no one way to solve a problem, depending on the nature of the challenge being addressed, there are different approaches based on the data available. The image in Annex A.1 shows a classification diagram of the most relevant ML algorithms. As described by

the diagram, there are four types of ML methods: supervised, unsupervised, reinforcement and ensemble methods (Burger, S.V. 2018).

3.2.1. Supervised learning

The machine learns under the presence of a supervisor. Data labelled is provided to algorithms, i.e. a set of known data that includes the desired inputs and outputs. Consequently, the labelled data act as supervisor. The algorithm must find a method to determine how to get from inputs to outputs by identifying patterns in the data set. The machine learns from the observations, makes predictions and, if needed, is corrected by the labelled data with the correct answer. This process continues until the algorithm reaches a high level of precision.

These algorithms are easy-to-explain models that are used to solve math tasks, such as exploration of patterns with numbers, determine closeness of data points or estimate the trend of a vector. When data is categorised or numerical, supervised learning is applied to predict a category (classification method) or to predict a number (regression method).

Neural Networks are models inspired by the behaviour of biological neurons and brain connections allowing models to identify patterns and solve problems. A subset of neural network algorithms is found in the supervised learning type. Neurons or nodes are connected to each other forming a neural network consisting of node layers, including an input layer, zero hidden layers or more and an output layer. Neurons are connected to each other with associated weights and thresholds applicable to the data sent to other neurons. As human brains, neural networks have learning power but rely on training data to learn and enhance performance over time. One of the fastest growing technologies in the world according to Pal, R (2020) is Deep Learning, a deep neural network with many hidden layers and many nodes in every hidden layer to train complex data that improve the efficiency and accuracy of predictions.

Supervised algorithms are the most used in real-life assignments as they usually tend to learn quicker with a supervisor, as long as data quality utilised and modelling approach chosen are favourable. According to Wakefield, K (2021), 7 out of the 10 most common and popular machine learning algorithms belong to the supervised group.

3.2.2. Unsupervised learning

Algorithms study data to identify patterns and correlations but neither the output is included in the data nor wrong predictions can be corrected, i.e. no labelled data is used. Therefore, the machine is provided with data without being categorised, but it learns to recognise that there are different patterns, i.e. types of outputs, although it will not know what they are named.

For instance, after being able to recognise groups A, B and C, the machine will connect to external databases, such as the internet, and will look for pattern similarities with respect to the data initially provided. The machine will end up naming the groups A, B

and C, providing the correct output, because it will find them on the external database without anyone having told it. Thus, the more data evaluated, the higher the ability to make decisions with the corresponding gradual improvement.

This method is used to divide by similarity (clustering method), find hidden dependencies (dimension reduction method) and identify sequences (association method). A subset of Neural Networks algorithms is found in the unsupervised learning type, such as SOM Neural Network models (Hainaut, D., 2018).

3.2.3. Reinforcement learning

Reinforcement learning is a type of ML algorithm that aims to develop a system, called an agent, that wants to improve its efficiency by performing a task based on interaction with its environment. In order to accomplish its commitment, the agent receives rewards that allow to adapt its behaviour. Perhaps, this type of algorithm could be considered as a sort of supervised algorithm since the agent is going to receive a label, i.e. the reward. However, this label obtained is not the absolute truth associated with its behaviour, but only an indicator of how well or badly it performed its action. (Sutton, R. S. and Barto, A. G., 2018).

For instance, if an algorithm is trained to play chess efficiently, the fact that the algorithm wins a game and receives a positive reward does not indicate that the pieces movements made were correct. It simply indicates that the pieces movements were suitable in the specific environment of that game. As it receives rewards, the agent must develop the correct strategy, named politics, that leads the agent to obtain positive rewards in all possible situations. A subset of Neural Networks algorithms is also found in reinforced learning type.

3.2.4. Ensemble methodology

This method is set by multiple models that each of them produces different predictions. The predictions from the different models are combined to obtain a single prediction with a better predictive performance than the individual predictions. As each model works differently, their combination tend to compensate their individual errors.

3.3. Insurance practices

The objective of ML is to predict results based on incoming data. Accordingly, the insurance sector stores large datasets, makes the corresponding selection of features or independent variables that can explain the dependent variable and selects the algorithms that better fits the data.

Datasets can be obtained manually (containing less errors, taking more time to be collected and being expensive) o automatically (containing different formats with incomplete content, being free/cheap and accessible). The larger the datasets the better, considering that it should be complete, well formatted and diverse. However, the best algorithm will perform badly with a wrong dataset.

The selection of the ML method can be considered an art. Supervised and unsupervised learning is applied when sample data sets are available with clear variables, Reinforcement learning works on interacting with an environment, or Ensemble methods are implemented when complex data is available with unclear variables.

The selection of the features or variables in the dataset that form hidden patterns to generate predictions may not be easy to find, trying the model several times until some results make sense. The formula and/or equation/s that produce the prediction with the variables of the dataset, the algorithm, may be easy to find, since many are available. However, the equation providing the most accurate results may take not only additional time to be found, i.e. after applying the corresponding fine-tuning modifying variables and parameters, but also expert judgment. The solution to any problem can come from different approaches, as several algorithms can fit, but the key is to decide which algorithm fits better. The methodology utilised impacts on the performance, accuracy, and dimension of the model.

Classical econometrics has an economic base that specifies the exogenous variables to be used in a model created with an economic reasoning supported by economic theories through a progressive study of the problem. Instead, ML is created to find strong correlations in variables that allow predictability, becoming a much more operational work with almost no need for a theoretical basis that explains the sense of using the exogenous variables utilised, at the expense of not being able to contrast parameters and making a more difficult interpretation of these parameters. ML searches the best goodness of fit for the model used considering causal inference, which allows using more sophisticated algorithms to improve such goodness of fit and perform more automatic modelling, saving time.

Under the supervised algorithm group, classification methods are used in insurance industry for similar document search, policy handwriting recognition and fraud detection. Additionally, regression methods are used for demand and sales volume analysis, insurance pricing any number-time correlations.

Labelled data may be expensive to obtain or may not even exist, and then, unsupervised algorithms are the only method applicable when there is no other available option. However, as these models are not limited to labels set, they can rapidly adapt to new and emerging patterns, for example, with dishonest behaviour. As informed by Kapetanvasileiou, G. (2019), a New Zealand health insurer used unsupervised learning methods to identify cases where practitioners were deliberately overcharging patients for a particular procedure or providing unnecessary treatment for certain diagnoses. Clustering methods are used in the insurance industry on client segmentation and identification of irregular conducts. Dimension reduction methods handle risk management, advice system and fake image detection. Additionally, Association learning

can cope with modelling regarding sales prediction, insurance products acquired simultaneously and study patterns in web page navigation.

Reinforcement learning algorithms follow the action-and-reward approach, i.e. reduce mistakes instead of forecasting. Similarly, Krashennnikova, E. et al (2019), describe how to successfully apply Reinforcement learning to model the problem of renewal price optimisation. After offering renewal prices to all the portfolio's policyholders, the states of the insurer will change (high renewal price may represent one customer less, which could be a bad decision by lowering revenue or a good decision by increasing revenue) and become a succession of situations (states), decisions (actions), and utilities (rewards) received.

The largest technology companies use the Neural Network approach as a little increase in algorithm accuracy represents additional millions in revenues. They can take the place of any model mentioned in this paper and are used for identification of objects, customer speech recognition, translation and accident image processing. However, many other applications are being investigated. According to Fernandez-Arjona, L. (2019), Neural network can be an approximation for solvency calculations when using Monte Carlo simulations to model options and guarantees for life businesses, considering future discounted cash flows simulated with Economic Scenario Generators. Similarly, Amir Hejazi, S and Jackson, K. R. (2016), propose a neural network implementation of the spatial interpolation technique, providing superior accuracy with neural network approach rather than Monte Carlo simulations on variable annuity modelling.

Finally, Ensemble methods cope with any supervised and unsupervised learning algorithm, on object detection and search systems. Like Neural Networks, Ensemble methods nowadays are one of the most used models in ML industry and research, providing favourable results such as fraud detection in health insurance (Kunickaitè, R. et al, 2020). Several approaches are used to build ensembles, such as Voting, Boosting, Stacking and Bagging.

Voting method consists of multiple ML models trained to provide their predictions individually, which have a vote associated. The final prediction is what the majority of the models vote. An application of voting is considered in aviation incident risk prediction (Zhang, X. and Mahadevan, S., 2018).

The *Bagging* method also combines multiple ML models but, unlike Voting, the way to get the errors to compensate with each other, each model is trained with subsets of the training set. These subsets are formed by randomly choosing samples from the training set. An example of bagging is used by Knighton, J. et al (2020) in predicting flood insurance claims with hydrologic and socioeconomic demographics.

In *Boosting*, each model tries to fix the errors of the previous models. In binary classification, the first model tries to learn the relationship between the input variables

and the outputs. As the first model will make some errors, the second one will try to reduce the errors of the first model. Su, X and Bai, M (2020) show how a gradient boosting model is able to capture the non-linear dependence between the claim frequency and severity using French car insurance claim data.

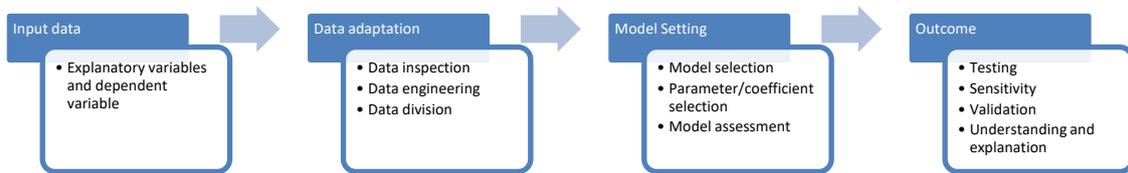
Stacking approach basically combines the predictions of base models, i.e. the output of multiple models is used as the input of multiple models. For instance, Prakash, G. and Jyothi, R. N. (2019) deal with cost reduction and enhancement in nature of care by establishing preventative interventions for risk patients for hospitals and insurers. With clinical data and evaluating risk associated, the use of stacked ensemble machine learning techniques obtain higher predictive accuracy than traditional models and also identify high-risk patients with low amount of data.

4. Automatic Machine Learning

4.1. AutoML Basics

During the last years, conventional ML has made a huge progress in allowing models to learn without being programmed for it. However, ML modelling still requires human expertise to be set. The conventional ML process is shown in Figure 2:

Figure 2: Conventional Machine Learning process.



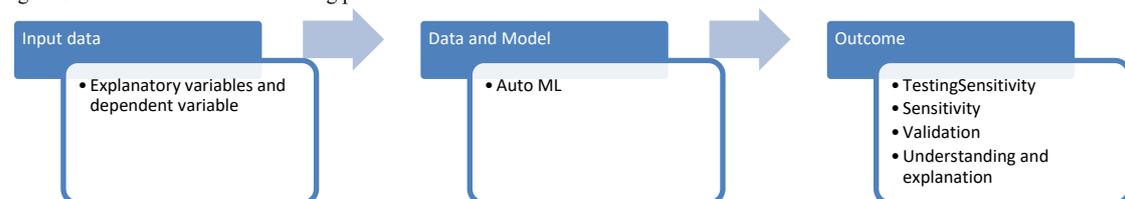
Source: own elaboration from Nilsson, N. J. (1998)

On the other hand, AutoML approach has the aim of making the modelling more automatic by going from inputs to outputs quicker, not only by dealing with data preparation and model selection tasks more efficiently but also providing the best ML method based on the data used.

In general, using AutoML might be considered an advantage in modelling due to the shortage of actuaries and data scientist with ML knowledge and experience. Moreover, large input data sets can be used in its raw state with AutoML, i.e with no additional preparation is needed, and the process increases efficiency by allowing data scientist to focus on the problem rather than the model and/or parameter selection, reducing human error .

The AutoML modelling process can be seen in Figure 3:

Figure 3: Automatic Machine Learning process.



Source: own elaboration from Hutter, F. et al (2019)

Consequently, the future of the conventional ML process may be AutoML. However, there are several disadvantages too: models are more complex and may be difficult to explain, high run time (if performed in-house) and high IT resources are needed to process (if performed in-house).

Generally, AutoML is mainly used to perform predictions with classification tasks, regression tasks and time-series forecasting. AutoML supports ensemble modelling by default. As previously mentioned, ensemble methods enhance ML outputs and predictive performance by mixing numerous models as opposed to utilising single models.

According to Boral, S. (2019), AutoML was mentioned for the first time in 2014 at a workshop in the University of Freiburg, Germany. The organisers considered the idea of developing a machine learning method that would not require any expert judgement to configure it.

4.2. AutoML systems available

The following section covers a non-exhaustive list of AutoML toolkits currently available:

- Auto-sklearn is an open-source AutoML tool created by Feurer, M. et al. (2019) implemented in Python, built around scikit-learn library.
- AutoML-Tables is a commercial AutoML tool created by Google (2021) on Google Cloud Platform
- AutoGluon is an open-source AutoML tool created by Amazon (AutoGluon, 2021)
- AutoML-azure is a commercial AutoML tool created by Microsoft (2021)
- Auto-WEKA is an open-source AutoML tool created by Thornton, C. et al (2016)
- H2O AutoML is an open-source AutoML tool created by H2O (2021)

4.3. H2O's AutoML

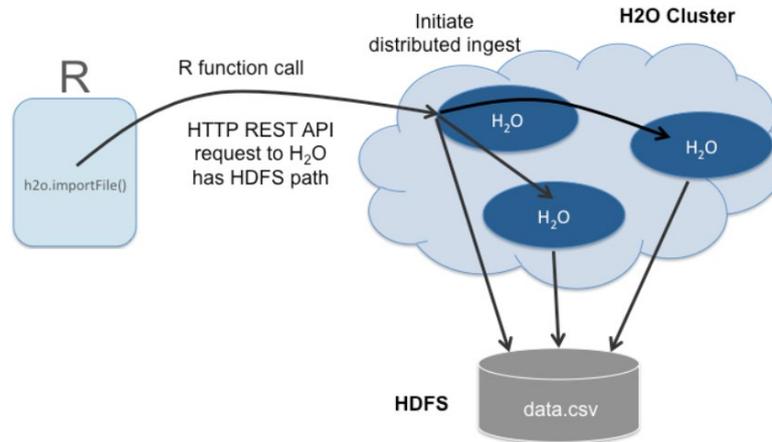
H2O is a company founded in 2012 with headquarters in Mountain View, California, United States of America. The aim of the founders, who are Java-engineers, was to build a platform that could work on big data sets in a distributed manner, had a large amount of ML algorithms available to work with, and provided data science functionalities.

H2O is also an open-source platform (Apache 2.0 Licensed), created by the H2O company, programmed in Java programming language with several Application Programming Interfaces (API) available such as R or Python, including a web interface for users that may not know programming. The platform is a distributed machine learning system, which distributes the data across multiple nodes and clusters (needed for very large amount of data) and can easily deploy models to production level with its high-performance Java code through models programmed in R or Python programming languages. Basically, H2O system is a library with a group of unsupervised and supervised ML algorithms such as Gradient Boosting Machine (GBM), Random Forest (RF), Deep Neural Network (DNN), Generalised Linear Model (GLM), stacked ensembles, among others (H2O, 2021).

H2O is considered a distributing computing system due to its H2O cluster and its H2O frame. Any R or Python script on H2O will start setting the H2O cluster which is a Java process that sets a block of memory where model, data and computations are going to be processed. The cluster has no size limit, being able to run from a laptop, any physical machine or physical cluster, and has multiple nodes with shared memory model, seeing each node only some rows of the data. A clustering algorithm tries to find out inherent groups in the dataset, for instance grouping insureds by driving preferences. The system faces an association problem which tries to discover rules that explain the data, for

example insureds that drive red vehicles tend to drive over the speed limit once a month. Figure 4 shows a H2O Cluster when interacting with R:

Figure 4: Representation of H2O Cluster interacting with R.



Source: H2O (2021) H2O Architecture

Models are strong in some areas and weak in others, but correctly combined hidden benefits may appear. Ensembles often out-perform individual models. In this case, stacking ensemble approach is applied by H2O following approaches of Wolpert, D. (1992) and Breiman, L. (1996) to learn what the best combination of the base models generates the best outcome.

The H2O’s AutoML process takes the following steps (H2O, 2021):

- Basic data pre-processing: parsing or analysing data structure, variable structure identification, handling missing values, filling NAs, splitting data internally into groups based on specific criteria and transformation of original dataset features into new and more predictive ones for better model performance
- Trains a random grid of GBMs, DNNs, GLMs, among other models, using a particular hyper-parameter space.
- Individual models are tuned using cross-validation in order to avoid overfitting.
- Two stacked ensembles are trained: “all models” ensemble and a lightweight “best of family” ensemble. The stacking method uses a second-level “metalearning” algorithm to find the optimal combination of base models.
- Returns a “leaderboard” of all models, i.e. a ranking of all models based on cross-validation performance ranked by a default or specified matrix (a binary classification problem would be ranked by Area Under the Curve or AUC by default).

As previously mentioned, H2O allows using its AutoML functionalities through a web-based interactive environment named Flow, especially designed for users who do not know coding. The image of Annex A.2 shows the H2O Flow interface and, in the next section, a practical case is generated with it.

5. Case: Fraud detection

This section intends to demonstrate that the use of AutoML is worthwhile by comparing H2O capabilities to predict car insurance claim fraud detection with several ML techniques.

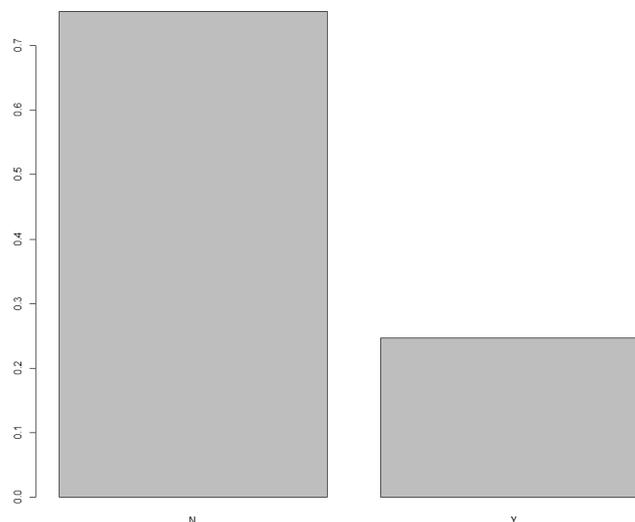
5.1. Data analysis with R

A supervised binary classification problem on fraud detection is considered under this practical case in R studio with R programming language. United States (US) car insurance claim data from 2015 is sourced from Kaggle (Sharma, R., 2019) with 1000 observations and 39 variables or claim features which may help to detect fraudulent claims. In particular, policy data come from three states: Ohio, Indiana and Illinois. A brief description of the meaning of the 39 variables is shown in the Annex A.3 although the original variable name is very descriptive.

One of the most common problems data analysts face when working with data occurs when doing the cleaning/adapting analysis on the raw data obtained, specially the fact of dealing with NA, Unspecified and/or missing values. Deletion of the observations is a solution in these cases. In this project it is better considered to keep data rather than to discard it, as small datasets are not desired for the analysis and working with the total number of claims is preferable.

Analysing the structure of the raw data obtained (see Annex A.4), the reader can observe the common error with zip code variable, “insured_zip”, which is recognised as integer instead of a character to become a categorical variable. As previously mentioned, data content does not normally come in the perfect format to work with. For instance, “policy_csl” variable is recognised with the character class but, ideally, it should be considered an integer. In addition, data content is not fully complete as it contains “?” values in several variables. An histogram of the dependent variable is shown below:

Figure 5: Histogram of “fraud_reported” variable.



Source: own elaboration with R

The bar plot depicts that approximately 25% (Y variable) of the car insurance claims reported are fraudulent and 75% (N variable) are not fraudulent.

The H2O AutoML function does not need any data preparation before modelling. However, traditional ML approaches applied in this thesis may need some modifications in order to proceed with modelling. The Annex A.5 presents the summary of the variables. Some particularities of the data inform that:

- Insureds have been between 0 and 479 months as customers, with an average of 204 months.
- Insureds are between 19 and 64 years old, with an average age of 39 years old.
- Policy annual premium is between 433 US dollars and 2048 US dollars, with an average of 1256 US dollars.
- The umbrella insurance policy variable contains a minimum value of -1000000 US dollars. As this is a clear data error, it is changed to 0.
- Policyholder's capital gains over the last year are between 0 and 100500 US dollars.
- Policyholder's capital losses over the last year are between 0 and 111100 US dollars.

Moreover, variables that may be considered affecting the determination of a fraudulent claim are injury claim, property claim and vehicle claim. The descriptive statistics of these numeric variables (see annex A.6) indicate that:

Injury claim variable shows the following statistics:

- it contains 1000 observations (as previously informed)
- the mean is 7433 US dollars and the standard deviation is 4881 US dollars
- the median is 6775 which is the value below which are 50% of the observations
- 7235 is the value of the trimmed mean, a robust estimator of central tendency that describes the location of a data set without undue influence of extreme values.
- 5493 is the median absolute deviation (from the median) which is the median of the set comprising the absolute values of the differences between the median and each data value.
- the minimum injury claim amount is 0 and the maximum is 21450 US dollars
- 0.26 is the skewness or the degree of distortion from the symmetrical bell curve (or the normal distribution), and so it measures the lack of symmetry in data distribution. As the skewness is between -0.5 and +0.5, the distribution is approximately symmetric (see annex A.7)
- kurtosis is -0.77, which is used to describe the extreme values in one versus the other tail. As kurtosis is lower than 0 the data variable is categorised as platykurtic, i.e. a distribution that has a flat peak and has more dispersed scores with lighter tails.
- 154.35 is the standard error or the standard deviation of the data set divided by the square root of the number of observations. Standard error of the mean is a measure that estimates how close a calculated mean is likely to be to the true mean of the population.

Property claim variable shows the following statistics:

- it contains 1000 observations

- the mean is 7400 US dollars and the standard deviation of 4825 US dollars
- the median is 6750 and 7165 is the value of the trimmed mean
- 4878 is the median absolute deviation (from the median)
- the minimum claim value is 0 and the maximum claim value is 23670
- 0.38 is the skewness, which indicates that the distribution is approximately symmetric (see annex A.8)
- kurtosis is -0.39, thus the variable data is categorised as platykurtic
- the standard error is 152.57.

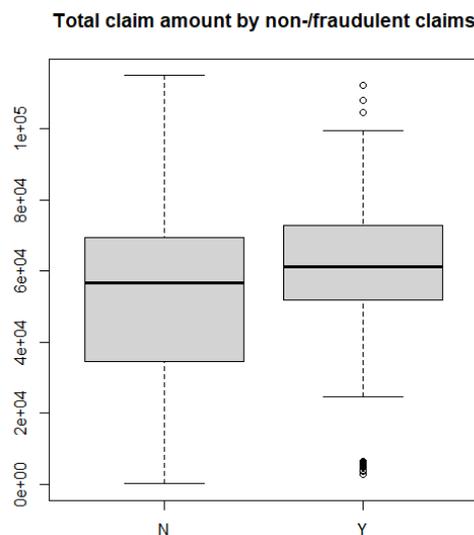
Vehicle claim variable shows the following statistics:

- it contains 1000 observations
- the mean is 37929 US dollars and the standard deviation is 18886 US dollars (both statistics are the largest in comparison to the property and injury claim variables previously analysed)
- the median is 42100 and 38944 is the value of the trimmed mean
- 14589 is the median absolute deviation (from the median)
- the minimum claim value is 70 and the maximum claim value is 79560
- -0.62 is the skewness, which indicates that the distribution is moderately negative, i.e. the left-hand tail is normally longer than the right-hand tail (see annex A.9)
- kurtosis is -0.46, thus the variable data is categorised as platykurtic
- the standard error is 597.24.

Descriptive analysis segmenting by the “fraud_reported” variable (see annex A.10) is also performed but no numerical variables seems to provide shocking clues on what makes the claim to be or not to be fraudulent. The most relevant findings are:

- Fraudulent reported claims have on average 30% increase of the umbrella insurance policy limit
- Fraudulent reported claims have on average 20% higher total claim amounts. This statement can be appreciated from the following graph (Y = fraudulent and N = non-fraudulent claims):

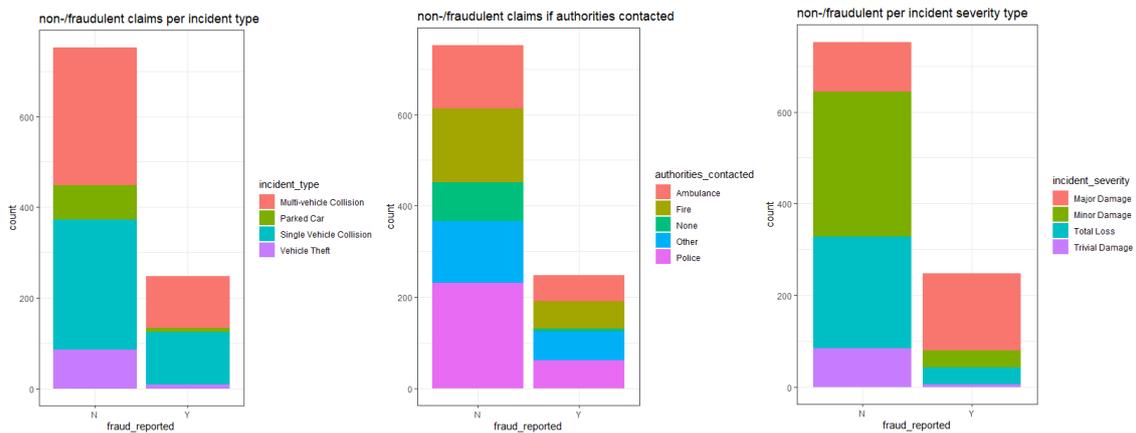
Figure 6: Total Claim amount by non-fraudulent claims.



Source: own elaboration with R

Additionally, analysis between categorical variables is produced which visually explains some sort of distinctions between non-/fraudulent claims. The distinguishing characteristics can be seen with the following bar plots with respect to accident type, authorities contacted and incident severity variables when counting the non-/fraudulent claims under each variable:

Figure 7: Non-/fraudulent claims by incident type, authorities contacted and incident severity.



Source: own elaboration with R

Fraudulent claims seem to occur more frequently with multi-vehicle collision and single vehicle collision, generally some kind of authority is contacted when the incident occurs, and major damages are suffered.

At this point the preliminary analysis of the data set is performed and the behaviour of the non-/fraudulent claims with respect to each predictor has been studied. Hence, it is time to determine the possible models to consider in the estimation phase.

5.2. H2O AutoML with R

The H2O AutoML full potential is shown at first and results are later compared with other modelling approaches. No modification of the original data is applied, with the exception of the error found on the umbrella insurance policy variable which contains a minimum value of -1000000 US dollars and is changed to 0.

Once the personal computer is connected to the H2O cluster (see Annex A.11), R Studio informs that 0.97 GB of H2O cluster memory are assigned to this project, and the H2O data frame can then be created in R memory. All H2O models are stored in H2O cluster memory. Hence, to get data into H2O Java memory, data are written in the local hard disk and then the H2O cluster reads them.

A brief description of the H2O data frame is shown in the Annex A.12. As expected, 39 variables are shown under column “Label”. The column “Type” automatically indicates the type of variables found which are “int” (integer), “string”, real (real number) and

“enum” (categorical/factor variables). No missing values are found in the dataset, but it contains many zeros in several variables. The largest “sigma” (standard deviation) is found in the umbrella insurance policy variable. “Cardinality” column automatically detects the number of levels of the categorical variables.

In H2O AutoML, unique features do not positively contribute to the modelling experience due to overfitting, such as incident date (60 levels). Additionally, no data leakage is considered if non-predictive variables are removed, such as policy number and policy bind date. Consequently, the previously mentioned variables are not considered in the modelling process. However, the fact of data content having “?” as value in some variables is not an issue for H2O AutoML.

AutoML function of H2O library automatically builds large number of models with the aim of finding the best model with no previous knowledge of the data. There is no need to divide the data set into train data and test data since H2O functionality does it automatically. Nevertheless, data are divided (90% training data set and 10% test data set)¹ in order to be compared to other modelling approaches in equal circumstances. Since the response column, “fraud_reported” variable, is identified as categorical variable, H2O automatically recognises the binary classification problem. Otherwise, if the response column was encoded with numbers, it would perform a regression.

Once dependent and independent variables are identified, two AutoML functions are set with a particular specification, although many others could be established (LeDell, E et al, 2020):

- Run the data over a maximum number of models, in particular 10 models
- Run the data during a time set, in particular 600 seconds

The time these functions take depends on the size of the data set and the resources given to process. Few lines of code make AutoML efficient to data scientist. Users leverage H2O to select the most promising algorithms by just taking a look at the data. Consequently, the user can spend time up front in the process analysing the problem. The following image shows the leaderboards of the two AutoML functions set:

Figure 8a: 10-model AutoML function leaderboard.

```
> print(lb10mod, n=nrow(lb10mod))
      model_id      auc  logloss  aucpr mean_per_class_error  rmse  mse
1      GBM_5_AutoML_20210504_081342 0.8718062 0.3617501 0.6681112 0.1398532 0.3329736 0.1108714
2  StackedEnsemble_AllModels_AutoML_20210504_081342 0.8690357 0.3618537 0.6537988 0.1390982 0.3357654 0.1127384
3      GBM_2_AutoML_20210504_081342 0.8684154 0.3874082 0.6649496 0.1707216 0.3441624 0.1184478
4      GBM_1_AutoML_20210504_081342 0.8682841 0.3928251 0.7010725 0.1590754 0.3476190 0.1208390
5      GBM_3_AutoML_20210504_081342 0.8674273 0.3904430 0.6399746 0.1612353 0.3434122 0.1179319
6  StackedEnsemble_BestOfFamily_AutoML_20210504_081342 0.8672566 0.3623195 0.6514763 0.1391114 0.3355080 0.1125656
7      GBM_4_AutoML_20210504_081342 0.8654020 0.3966743 0.6504421 0.1700848 0.3503018 0.1227114
8  GBM_grid_1_AutoML_20210504_081342_model_1 0.8649162 0.3835127 0.6307613 0.1751989 0.3455527 0.1194067
9      GLM_1_AutoML_20210504_081342 0.8533652 0.4031324 0.6116159 0.1715685 0.3547374 0.1258386
10     DRF_1_AutoML_20210504_081342 0.8438559 0.4438435 0.5775008 0.1582285 0.3585428 0.1285529
11     XRT_1_AutoML_20210504_081342 0.8212068 0.4940978 0.5508169 0.2269439 0.3846948 0.1479901
12  DeepLearning_1_AutoML_20210504_081342 0.7753998 0.5529930 0.5317847 0.2755245 0.4060462 0.1648735

[12 rows x 7 columns]
```

Source: own elaboration with R

¹ Although other desired proportion may be chosen, a division between training, validation and test data set may be considered more correct for modelling purposes with the disadvantage of dividing further the data and reducing the number of observations in the corresponding data sets.

Figure 8b: 600-second AutoML function leaderboard.

```
> print(1b600sec, n=row(1b10mod))
  model_id      auc  logloss  aucpr mean_per_class_error  rmse  mse
1 GBM_grid__1_AutoML_20210504_081538_model_15 0.8781348 0.4051963 0.6960356      0.1596991 0.3464263 0.1200112
2 GBM_2_AutoML_20210504_081538 0.8745831 0.3899473 0.6642134      0.1714897 0.3456668 0.1194855
3 GBM_grid__1_AutoML_20210504_081538_model_12 0.8734277 0.3685403 0.6686595      0.1604803 0.3399785 0.1155854
4 GBM_grid__1_AutoML_20210504_081538_model_24 0.8724791 0.3714731 0.6847860      0.1670649 0.3414147 0.1165640
5 GBM_grid__1_AutoML_20210504_081538_model_8 0.8711201 0.4464811 0.6935085      0.1818492 0.3606634 0.1300781
6 GBM_grid__1_AutoML_20210504_081538_model_16 0.8708838 0.3639406 0.6462297      0.1442911 0.3354304 0.1125136
7 GBM_grid__1_AutoML_20210504_081538_model_18 0.8696233 0.4009606 0.6612140      0.1678462 0.3489574 0.1217713
8 GBM_5_AutoML_20210504_081538 0.8694657 0.3670064 0.6401794      0.1472322 0.3371090 0.1136425
9 GBM_grid__1_AutoML_20210504_081538_model_14 0.8683760 0.4101087 0.6849571      0.1707610 0.3524380 0.1242125
10 GBM_grid__1_AutoML_20210504_081538_model_17 0.8682447 0.3778254 0.6439936      0.1552874 0.3435110 0.1179998
11 GBM_3_AutoML_20210504_081538 0.8681856 0.3947457 0.6883820      0.1701111 0.3460824 0.1197730
12 GBM_grid__1_AutoML_20210504_081538_model_2 0.8674240 0.3866492 0.6747669      0.1626533 0.3443827 0.1185995

[42 rows x 7 columns]
```

Source: own elaboration with R

Figure 8a shows 10-model AutoML function leaderboard and figure 8b shows the 600-second AutoML function leaderboard. Both leaderboards show the 12 top models with highest AUC. The highest AUC is 0.87, indicating that the top models have 87% probability of distinguishing between fraudulent claims and non-fraudulent claims. Since no specification was requested to score models, the leaderboard uses the cross-validation metric. A default performance metric for each ML task (binary classification or multiclass regression) is specified internally and the leaderboard is sorted by that metric. For this binary classification problem, the default ranking metric is the AUC.

The 10-model AutoML function leaderboard provides 12 models (of which two stacked ensembles) and the 600-second AutoML function leaderboard provides 42 models. The leaderboards provide the `model_id` variable (first column) which specifies the parameters of the models, and the remaining variables are metrics that represent the cross-validated performance of each of the models. The top model of the 10-model AutoML function is named “GBM_5_AutoML_20210504_081342” and the top model of 600-second AutoML function also obtains another GBM model under the name of “GBM_grid__1_AutoML_20210504_081538_model_15”. The chosen algorithm is a GBM in both cases, a ML technique that create a predictive model by chaining together a set of weak decision trees as a prediction basis to create a more robust classifier, allowing arbitrary optimization of a differentiable loss function.

As this is a binary classification problem, we have the following metrics: AUC, logarithmic loss, Area Under Precision Recall Curve (AUCPR), mean per class error, Root Mean Squared Error (RMSE) and Mean Squared Error (MSE). While AUC indicates how well a model classifies a binary target, the logarithmic loss metric is a probability value between 0 and 1, and it studies how close the predicted values generated by the model are to the actual target value, i.e. it evaluates the correctness of a classifier by penalising false classifications. Consequently, the higher the logarithmic loss value is, the higher divergence exists between the predicted probability and the actual value. For instance, the model “GBM_5_AutoML_20210504_081342” shows a logarithmic loss of 0.3617501 which is the lowest among the models selected.

As previously indicated, AUC (Ballings, M. and Van den Poel, D., 2015) metric considers how well the model can differentiate between true fraudulent claims and false fraudulent claims, with a value of 1 indicating the perfect classifier and a value of 1/2 indicating a

poor one. H2O uses the trapezoidal rule to approximate the area under the ROC curve, and consequently, AUC is usually not the best metric for an imbalanced binary target because a high number of true non-fraudulent claims may cause the AUC to look overestimated, considering H2O more appropriate to use in these cases the AUCPR metric. 0.6681112 is the AUCPR value of the previous mentioned model “GBM_5_AutoML_20210504_081342”, which is among the highest values of the models selected.

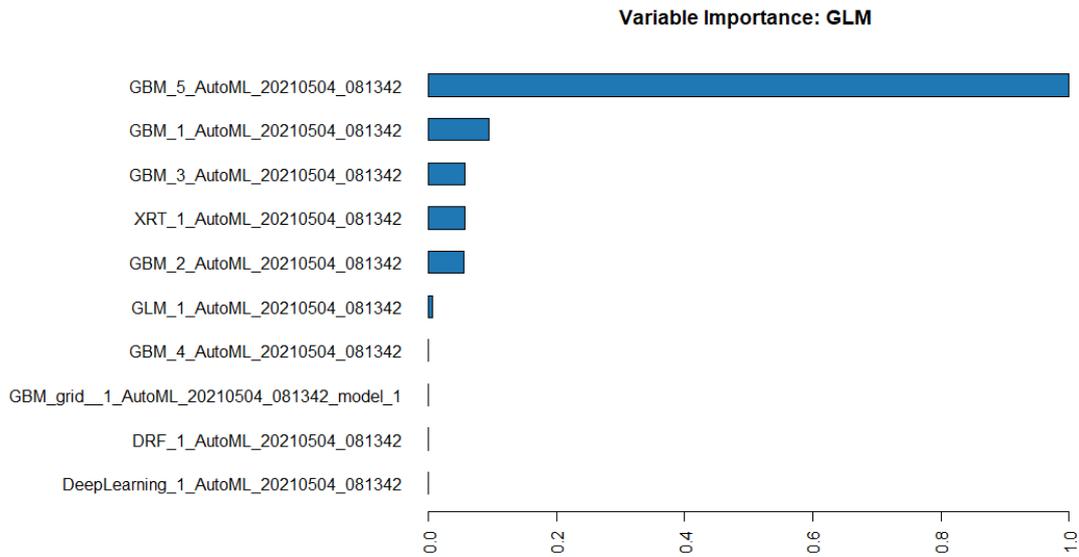
The “mean per class error” metric works for problems of classifying instances into one of three or more classes, i.e. multiclass classification, rather than binary classification as this is the case in this project. This metric calculates the average of the errors of each class in the multi-class data set and considers misclassification of the data across the classes. Consequently, the lower the value of “mean per class error”, the better the model, and actually, the mentioned “GBM_5_AutoML_20210504_081342” model obtains one of the lowest value (0.1398532).

The MSE metric calculates the average of the squares of the errors. The errors are the distances from the points to the regression line and these are squared to remove any negative sign. The RMSE metric calculates the standard deviation of the residuals or prediction errors. Therefore, the lower the MSE and RMSE, the better, and in fact, the “GBM_5_AutoML_20210504_081342” model results show 0.3329736 and 0.1108714, respectively, which are the lowest of the given outputs. Anyway, these are not good metrics for binary classification but for regression problems.

At the leaderboard of the 10-model AutoML function, Stacked Ensemble models are found in the second and sixth position. These models almost always outperform a single model (not the case in this practical case), and in fact, typically the StackedEnsemble_Allmodels wins and the StackedEnsemble_BestOfFamily remains a bit behind, as it can be seen in figure 8. The Stacked Ensemble "All models" is an ensemble of all the individual models in that AutoML run and Stacked Ensemble “Best of family” is a subgroup of models selected automatically by the H2O algorithm.

The image found in the Annex A.13 shows the variable importance for the metalearning algorithm inside of the “All models” stack ensemble model. Meta learner is the name of the algorithm which its goal is to take the predictions from the base models and predict the outcome. The features in the meta learner are actually predictions from the base model, representing each of the models that are in the stack ensemble. The meta learner is a model stored inside the stack ensemble and, as shown by the results, it corresponds to a Generalised Linear Model (GLM) in this case. In addition, the default metalearner algorithm is a GLM binomial logit elastic net, parameters $\alpha = 0.5$ and $\lambda = 0.000165$, and the variable importance of the metalearner is provided by the standardised coefficient magnitudes of the GLM (Annex A.14) which can be seen in more detail in figure 9:

Figure 9: Variable importance of the metalearner.

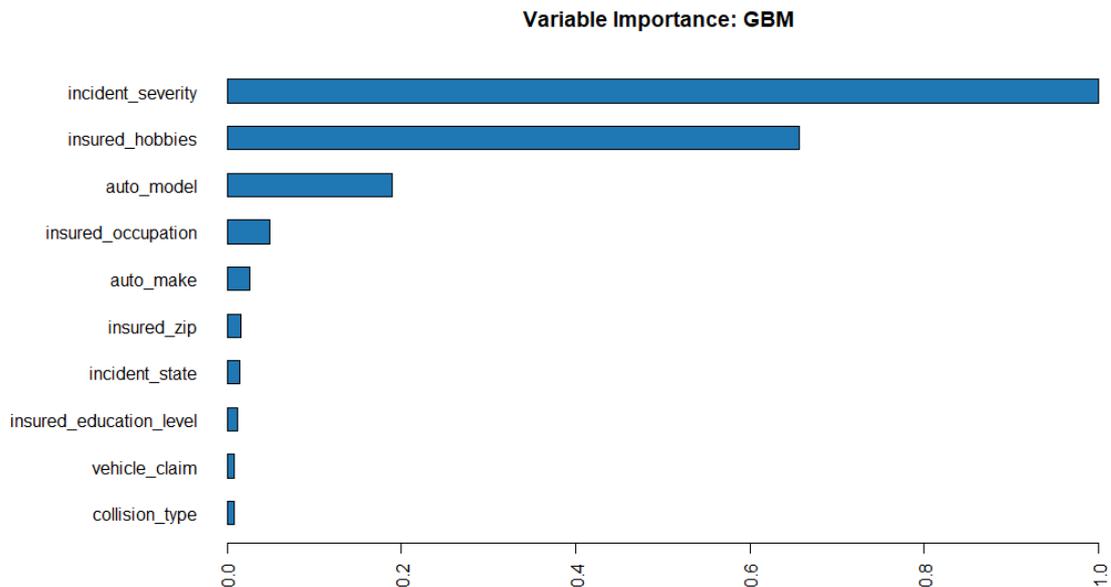


Source: own elaboration with R

Consequently, the most important base learner to the ensemble is the GBM_5_AutoML_20210504_081342. The models are extremely randomised trees (XRT), which is a variant of random forest, distributed random forest (DRF), GBM, Deep Learning and Generalised Linear model (GLM). From the sixth model included in the metalearner, the models are not doing well, which was already seen in the leaderboard. The goal with the metalearner is to ignore any model that is not useful.

Going back to the best model found (GBM_5_AutoML_20210504_081342), a Gradient Boosting model, the following image indicates that the “incident_severity” variable and “insured_hobbies” variable are by far the most predictive variables to determine fraudulent claims, followed by the “auto_model” variable:

Figure 10: Variable importance with GBM.



Source: own elaboration with R

It is very interesting to see that 2 or 3 variables are above the remaining ones for predicting. “Incident_severity” is one of the categorical variables previously analysed where distinctions between non-/fraudulent claims are found. For further information on the relative importance of other variables see Annex A.15.

All the parameters selected for the top model are found in Annex A.16, for instance “ntrees” indicates that 46 trees were created. This is not a large number of trees and it is better to stop the modelling before having too many trees and overfit the data. If desired, it is possible to use this model found with the AutoML function and change some parameters to beat the result just given in order to try to obtain a better model.

The following image shows the confusion matrix of the best model found with the train data set:

Figure 11: Confusion matrix.

```
> h2o.confusionMatrix(gbm)
Confusion Matrix (vertical: actual; across: predicted) for max f1 @ threshold = 0.521421549387896:
      N   Y  Error  Rate
N    639  35 0.051929 =35/674
Y     44 182 0.194690 =44/226
Totals 683 217 0.087778 =79/900
```

Source: own elaboration with R

The matrix shows that only 44 out of 226 claims would be considered non-fraudulent claims by the model although they actually are fraudulent claims. In addition, 35 out of 674 claims would be considered as fraudulent claims by the model but in reality they are not fraudulent claims. These results mean that the model would be mistaken 8.7% of the predictions made. To finish this section, given the GBM fitted model and the 100-observation test data set in which to look for variables with which to predict, R returns an H2O data frame object with probabilities and default predictions (for the full list of predicted values see Annex A.17):

Figure 12: GBM Prediction.

predict	N	Y
N:81	Min. :0.1466	Min. :0.009876
Y:19	1st Qu.:0.5537	1st Qu.:0.037713
	Median :0.9291	Median :0.070875
	Mean :0.7672	Mean :0.232760
	3rd Qu.:0.9623	3rd Qu.:0.446290
	Max. :0.9901	Max. :0.853372

Source: own elaboration with R

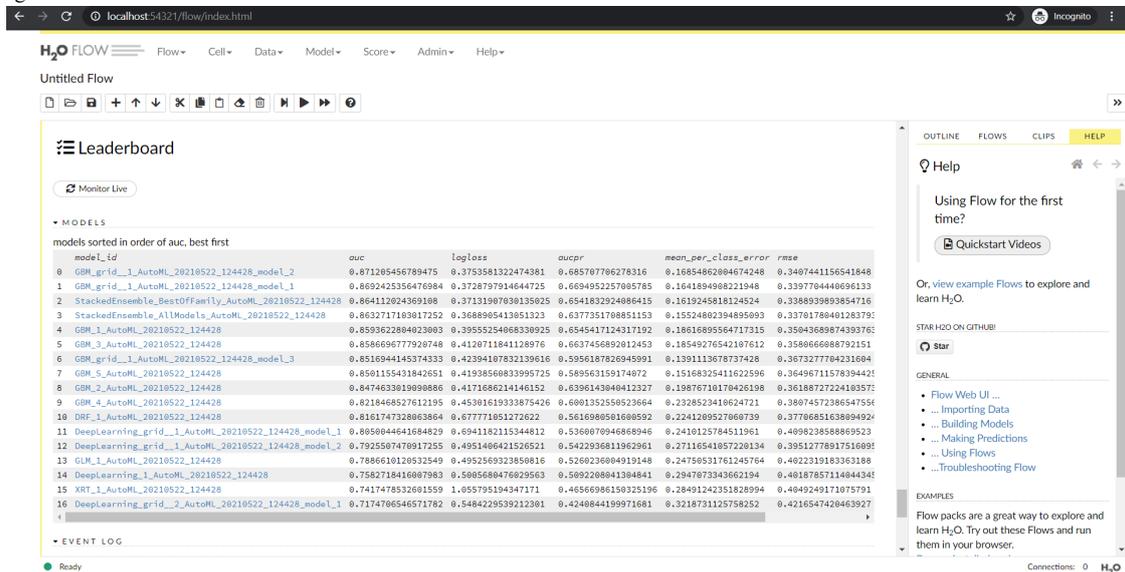
The model predicted 81 non-fraudulent claims and 19 fraudulent claims, and the test data set contains 79 non-fraudulent claims and 21 fraudulent claims. However, 9 claims were considered fraudulent and they were not, and 11 claims were not considered fraudulent, but they were. Hence, 20% of the times the model would be mistaken. After performing the selection of the model and the prediction with the H2O library in R, H2O Flow interface is shown.

5.3. H2O AutoML with Flow

The potential of the H2O AutoML function capabilities is applied with H2O Flow in this section. The aim is to demonstrate that the process carried out with RStudio can also be applied by simply downloading a file and using a web explorer to connect to the H2O Flow platform (H2O, 2021), without trying to replicate the results obtained in RStudio, i.e. this is a sample case study.

The same data set previously analysed is utilised and Annex A.18 shows the uploading process of the data set. The image in Annex A.19 shows how H2O Flow automatically detects the dataset variables, the picture in Annex A.20 visualises how to set the AutoML function and the image in Annex A.21 shows how to deselect explanatory variables from the dataset. The leaderboard of the corresponding run is shown below:

Figure 13: H2O Flow leaderboard.



Source: own elaboration with H2O Flow

As can be seen, the leaderboard informs that a GBM model, in this case under the name of GBM_grid__1_AutoML_20210522_124428_model_2, is chosen as the best model, as previously seen with RStudio. The identical output obtained when using Rstudio is not replicated by H2O Flow due to using different model seed and slightly different configuration. The image in Annex A.22 shows the confusion matrix which shows that only 22 out of 226 claims would be considered non-fraudulent claims by the model although they actually are fraudulent claims. In addition, 10 out of 674 claims would be considered as fraudulent claims by the model but in reality they are not fraudulent claims. These results mean that the model would be mistaken 3.6% of the predictions made.

Predictions are performed with the test data set and the model predicted 80 non-fraudulent claims and 20 fraudulent claims, although the test data set contains 79 non-fraudulent claims and 21 fraudulent claims. However, 8 claims were considered fraudulent and they were not, and 10 claims were not considered fraudulent, but they were.

After presenting the capabilities of H2O Flow potential, the process is repeated according to more widespread modelling methodologies that could handle binary classification problems such as artificial neural network, logistic regression and support vector machine according to PureAI Editors (2020). Additionally, decision tree classification algorithm is also tested as it an approach quick to use and easy to interpret (Chakure, A., 2019). Many other approaches could have been used for the purpose of this thesis, but the mentioned approaches are considered the most conventional and easy-to-use ML approaches.

5.4. Linear model - Perceptron with R

The perceptron is the simple neuron model, consisting of a basic unit of inference in the form of a linear discriminator with no hidden layer, from which an algorithm is developed, capable of generating a criterion to select a sub-group from a larger group of components. Thus, it serves to classify linearly separable problems.

The same data set previously analysed is desired to be used. However, Neuralnet package cannot handle characters or categorical inputs. Hence, beforehand, these variables should be either changed to dummy variables or eliminated. The solution is to proceed with both options since some variables have too many levels, such as “auto_model” and “incident_location”, but the remaining ones are changed to dummy values, to finally remain with 34 variables in total. Best practices recommend that the better the data, the better the model. Nonetheless, data preparation is a must workload when using Neuralnet.

Regrettably, after adding/removing variables and adjusting some Neuralnet function parameters, such as adding reps (repetitions), increasing the stepmax to give more time to the model to learn/converge or increasing the threshold to allow earlier stop for convergence, the model did not converge. One of the main problems in classification occurs when the algorithm never converges on updating the weights while it is being trained. It normally occurs when the classes are not perfectly linearly separable.

5.5. Linear model - Logistic regression with R

The drawback of linear regression is that models are sensitive to outliers and assume normally distributed errors when performing hypothesis test. Regardless, an extension of linear regression models are the Generalised Linear Models (Annette, D., 1990) in which Logistic regression is part of (Agresti, A., 2015). The model estimates a probability (between 0 and 1) and, with the default threshold² (set to 0.5), it performs the binary classification which the observation belongs to, with the predictors provided.

The same data set previously analysed is desired to be used, also eliminating the error found in “umbrella_limit” variable. However, GLM function cannot handle character type for the dependent variable. Hence, beforehand, this variable is changed to factor

² For the purpose of this thesis, threshold and parameter model settings tend to be left as default by the model function specifications as long as the model works, i.e. the minimum model specifications are manually set. Nevertheless, thresholds and parameters may be amended following data set utilised, increasing the model manipulation and expert judgement required for modelling.

variable. Additionally, “maxit” parameter from GLM function is set 100 in order for the model to converge. The parameter indicates the maximal number of iteratively reweighted least squares iterations and, by being incremented, it helps to find the maximum likelihood estimate and converge.

The GLM model output (see Annex A.23) show nearly all coefficients’ p-values with value of 1 (“collision_typeSide Collision” variable provides a “NA”). Coefficient’s p-value tests the null hypothesis that the particular coefficient is equal to zero, and hence, it has no effect on the model. For instance, a p-value lower than 0.05 indicates that the null hypothesis can be rejected with 95% confidence. In other words, coefficients with low p-values are likely to be meaningful additions to the GLM model since movements in the independent variables are related to movements in the dependent variable, in this case, “fraud_reported” variable. Consequently, none of the coefficients are statistically significant to the model and care should be taken to discard them, as keeping them could contribute to overfitting. Furthermore, the Akaike Information Criterion (AIC) is 1800, a measure of the relative quality of the statistical model. AIC manages a trade-off between the model’s goodness of fit and its complexity, by penalising the number of parameters in order to avoid overfitting.

Even making the model to converge does not guarantees a valuable predictive model. After several attempts, variables are carefully selected in order to obtain a meaningful model with an AIC of 642.49 (the lowest found) by using only two categorical variables from the original data set: “insured_hobbies” and “incident_severity” (distinctions between non-/fraudulent claims were found when analysing the data).

Figure 14: GLM coefficients.

```
> glm1 <- glm(fraud_reported ~ insured_hobbies+incident_severity, data = glmtrain, family = binomial)
> summary(glm1)

Call:
glm(formula = fraud_reported ~ insured_hobbies + incident_severity,
     family = binomial, data = glmtrain)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0833 -0.3937 -0.2688  0.1316  2.8941

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)         1.16430    0.47762   2.438  0.0148 *
insured_hobbiesbasketball -1.08216    0.72328  -1.496  0.1346
insured_hobbiesboard-games  0.03986    0.64084   0.062  0.9504
insured_hobbiesbungee-jumping -1.21138    0.67752  -1.788  0.0738 .
insured_hobbiescamping    -1.73775    0.71392  -2.434  0.0149 *
insured_hobbieschess       3.58035    0.63413   5.646 1.64e-08 ***
insured_hobbiescross-fit   3.00462    0.65788   4.567 4.94e-06 ***
insured_hobbiesdancing    -1.18420    0.73880  -1.603  0.1090
insured_hobbiesexercise   -1.28608    0.64003  -2.009  0.0445 *
insured_hobbiesgolf       -1.13487    0.69812  -1.626  0.1040
insured_hobbieshiking     -0.46880    0.64031  -0.732  0.4641
insured_hobbieskayaking   -2.00505    0.81066  -2.473  0.0134 *
insured_hobbiesmovies     -0.95798    0.65495  -1.463  0.1436
insured_hobbiespaintball  -1.00653    0.61738  -1.630  0.1030
insured_hobbiespolo       -0.32825    0.64029  -0.513  0.6082
insured_hobbiesreading    -0.36525    0.60055  -0.608  0.5431
insured_hobbieskydiving   -0.39438    0.64916  -0.608  0.5435
insured_hobbiesleeping    -1.52901    0.67434  -2.267  0.0234 *
insured_hobbiesvideo-games -0.54002    0.67644  -0.798  0.4247
insured_hobbiesyachting   -0.33528    0.61646  -0.544  0.5865
incident_severityMinor Damage -3.31759    0.28934 -11.466 < 2e-16 ***
incident_severityTotal Loss -3.33181    0.31186 -10.684 < 2e-16 ***
incident_severityTrivial Damage -3.87925    0.55418  -7.000 2.56e-12 ***

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1014.39  on 899  degrees of freedom
Residual deviance: 596.49  on 877  degrees of freedom
AIC: 642.49

Number of Fisher Scoring iterations: 6
```

Source: own elaboration with R

The previous image shows the coefficients' estimate values, its standard errors, the z-statistics (estimate value divided by the standard error) and the p-values. For this new model, "maxit" parameter is not needed. The multiple logistic regression used finds one of the most significant coefficients the "insured_hobbieschess" coefficient. This coefficient has a value of 3.58, indicating that if chess is the hobby of the insured, there is associated an increase in the probability of reporting fraudulent claims.

The following image shows the confusion matrix of the best model found with the train data set:

Figure 15: GLM Confusion matrix.

```
> confusionMatrix(glmtrain$fraud_reported,predict.train)
Confusion Matrix and Statistics

          Reference
Prediction N    Y
   N      610   64
   Y       58  168

      Accuracy : 0.8644
      95% CI   : (0.8403, 0.8861)
 No Information Rate : 0.7422
 P-Value [Acc > NIR] : <2e-16

      Kappa : 0.6427

 Mcnemar's Test P-value : 0.6508

      Sensitivity : 0.9132
      Specificity : 0.7241
   Pos Pred Value : 0.9050
   Neg Pred Value : 0.7434
      Prevalence : 0.7422
   Detection Rate : 0.6778
 Detection Prevalence : 0.7489
   Balanced Accuracy : 0.8187

 'Positive' class : N
```

Source: own elaboration with R

The matrix shows that only 58 out of 226 claims would be considered non-fraudulent claims by the model although they actually are fraudulent claims. In addition, 64 out of 674 claims would be considered as fraudulent claims by the model but in reality they are not fraudulent claims. These results mean that the model would be mistaken 13.6% of the predictions made.

To finish this section, given the GLM fitted model and the 100-observation test data set in which to look for variables with which to predict, predictions are performed (see Annex A.24). The model predicted 80 non-fraudulent claims and 20 fraudulent claims, and the test data set contains 79 non-fraudulent claims and 21 fraudulent claims. However, 7 claims were considered fraudulent and they were not, and 8 claims were not considered fraudulent, but they were. Additionally, the model accuracy is 85%, which measures the proportion of observations that have been correctly predicted. After performing the selection of the model and the prediction with the GLM, the process will be repeated with a multilayer perceptron model.

5.6. Neural Network – Multilayer perceptron with R

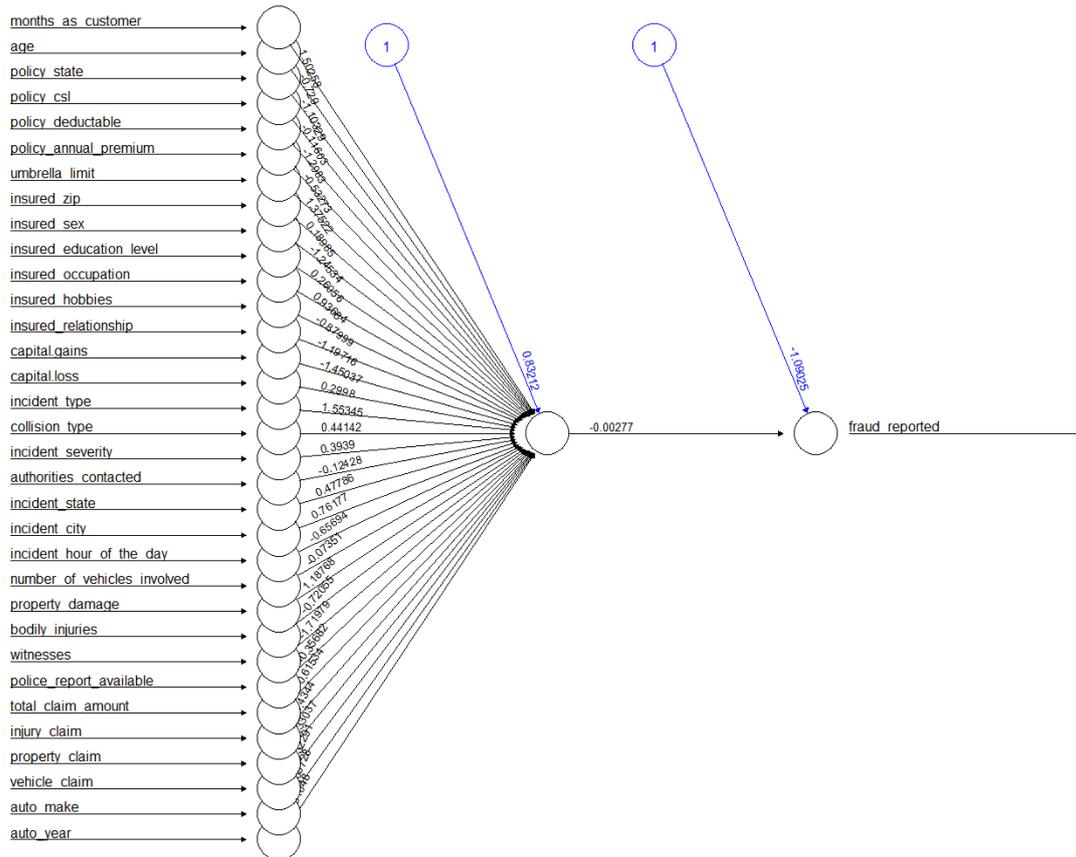
Unlike perceptron, the multilayer perceptron is a neural network formed by at least 3 layers of nodes (input layer, a hidden layer and an output layer) which has the ability to solve problems that are not linearly separable, which is the main limitation of the perceptron. Hidden layers use backpropagation to optimise the weights of the input variables with the aim of improving the predictive power of the model.

The same data set previously analysed is desired to be used. As previously seen, Neuralnet package cannot handle characters or categorical inputs. Hence, beforehand, these variables should be either changed to dummy variables or eliminated. The solution is to proceed with both options since some variables have too many levels, such as “auto_model” and “incident_location”, but the remaining ones are changed to dummy values, to finally remain with 34 variables in total. Best practices recommend that the better the data, the better the model. Nonetheless, data preparation is a must workload when using Neuralnet. The model setting is specified with parameter “linear.output” as FALSE since non-linear implications are considered to be affecting the independent variable (fraudulent claim) by the predictors utilised.

The number of hidden layers to set in the neural network is not a straightforward decision³. Trial and error approach is the only way to compare how accurate the predictions are by changing the number of hidden layers. Too many hidden layers and neurons may overfit the data and accuracy may sometimes improve without any hidden layers at all. For the purpose of this project, no specific number of hidden layer is set, letting the model set the default number of hidden layers, i.e. 1 hidden layer with one neuron. The following graph represents the neural network set:

³ According to Heaton, J. (2017) There are many rule-of-thumb methods for determining an acceptable number of neurons to use in the hidden layers, such as: the number of hidden neurons should be between the size of the input layer and the size of the output layer, 2/3 the size of the input layer plus the size of the output layer or less than twice the size of the input layer.

Figure 16: Neural network.



Source: own elaboration with R

The independent variables are shown in the input layer (column of nodes from the left side). Their values are multiplied by the appropriate weights (which are shown in the plot next to the nodes) and sent to the next layer, the hidden layer (central column). The hidden layer, with one node, gets a bias value input to it (labelled as 1) for the neuron to represent any separating hyperplane. The same process is performed for the output layer (right column) which also has one node. For the detail of the weights between inputs, hidden layer and output, and the error of the model see Annex A.25. As previously seen for other models, the following image shows the confusion matrix with the train data set:

Figure 17: Multilayer perceptron confusion matrix.

	prediction	
actual	0	1
0	674	0
1	0	226

Source: own elaboration with R

Surprisingly, the model does not predict fraudulent claims at all. These results mean that the model would be mistaken 25.1% of the predictions made.

To finish this section, given the neural network fitted model and the 100-observation test data set in which to look for variables with which to predict, predictions are performed (see Annex A.26).

The model predicted 100 non-fraudulent claims and 0 fraudulent claims (threshold is set to 0.5), but the test data set contains 79 non-fraudulent claims and 21 fraudulent claims. Therefore, 21 claims are not considered fraudulent (and they are fraudulent) and the model correctly predicts 79% of the observations. Although the model seems to work, results are not promising since no fraudulent claims are detected. Model parameters should be adjusted, or a better selection of input variables should be performed, which means further preliminary analysis. After performing the selection of the model and the prediction with the multilayer perceptron model, the process will be repeated with a Support Vector Machine model.

5.7. Support Vector Machine with R

This algorithm can be considered an extension of the perceptron algorithm. The optimisation goal of the Support Vector Machine (SVM) is to establish a decision line that separates the classes by maximising the margin between this line and the sample points (or support vectors) close to this hyperplane. According to Kharel, S. (2020), perceptron detains after classifying data rightly while Support Vector Machine (SVM) detains after it finds the best plane. SVM procedure provides the maximal distance between data points of both classes, providing reinforcement and then classifying with more reliability future data points.

The same data set previously analysed is desired to be used. Similarly seen in other algorithms, “svm” function of “e1071” package (Meyer, D. et al 2021) cannot handle characters or categorical inputs for the dependent variable. Hence, beforehand, this variable should be changed to 1 (fraudulent claim) and 0 (non-fraudulent claim). Finally, the most promising model found, after specifying several parameters of the model such as the linear Kernel function, obtains 479 support vectors (see Annex A.27) with the following confusion matrix with the train data set:

Figure 18: SVM confusion matrix.

```
> print(confmatrixlin)
      Actual
Predicted 0  1
0 667  10
1   7 216
```

Source: own elaboration with R

The matrix shows that only 10 out of 226 claims would be considered non-fraudulent claims by the model although they actually are fraudulent claims. In addition, 7 out of 674 claims would be considered as fraudulent claims by the model but in reality they are not fraudulent claims. These results mean that the model would be mistaken 1.9% of the predictions made.

To finish this section, given the SVM fitted model and the 100-observation test data set in which to look for variables with which to predict, predictions are desired to be

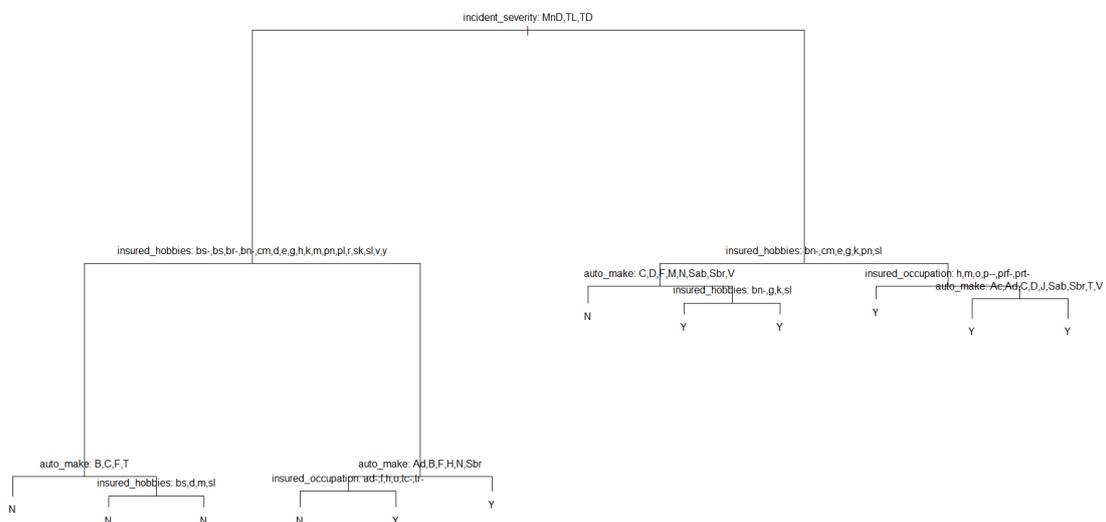
performed. However, as can be seen in Annex A.28), due to fact that the number of levels in the “auto_model” variable differs between the train and the test data sets (39 for the former and 36 for the latter), R indicates an error. Consequently, the SVM model is not comfortable making predictions with the test data set and the exercise must finish at this point. After trying the SVM model, the process will be repeated with a decision tree model.

5.8. Decision trees with R

The decision tree is a scheme of multiple branches nested in the form of a tree in such a way that following the branches of the tree a prediction is obtained. Decision tree algorithm drills down the data set by asking questions until the classification meets the properties required in the particular branch, showing robustness when dealing with outliers.

The same data set previously analysed is desired to be used. Similarly seen in other algorithms, “tree” function of “tree” package (Ripley, B., 2019) not only is unable to handle characters but also the categorical factors cannot exceed 32 levels. Hence, beforehand, characters should be changed to categorical factors and “auto_model” and “incident_location” variables should be discarded. The summary of the decision tree model (see Annex A.29) shows the relevant variables utilised for dividing the data which minimises the deviance rate. The decision tree is shown below:

Figure 19: Decision tree.



Source: own elaboration with R

The tree show how it performs the classification depending on the variables (an observations) shown in the branches, to go all the way down to the leaf nodes consisting of the fraudulent claim response value of Y (fraudulent) or N (non-fraudulent).

As previously seen for other models, the following image shows the confusion matrix with the train data set:

Figure 20: Decision tree confusion matrix.

```
> dtreepred<-predict(dtrees, newdata = dttrain,type = "class")
> with(dttrain,table(dtreepred,fraud_reported))
      fraud_reported
dtreepred  N    Y
N      623  48
Y       51 178
```

Source: own elaboration with R

The matrix shows that 48 out of 226 claims would be considered non-fraudulent claims by the model although they actually are fraudulent claims. In addition, 51 out of 674 claims would be considered as fraudulent claims by the model but in reality they are not fraudulent claims. These results mean that the model would be mistaken 11% of the predictions made.

Given the Decision Tree fitted model and the 100-observation test data set in which to look for variables with which to predict, predictions are performed (see Annex A.30). The model predicted 76 non-fraudulent claims and 24 fraudulent claims, but the test data set contains 79 non-fraudulent claims and 21 fraudulent claims. Additionally, 6 claims are not considered fraudulent (and they are fraudulent) and 9 claims are considered fraudulent (but they are not fraudulent). Consequently, the model mistakes 15% of the predictions made with test data. This is actually a good result if the data adjustment and the discarding of two variables would not be considered.

6. Results

All models considered to be able to predict the non-/fraudulent claims were challenged in the previous section. The most relevant metrics⁴ considered for this binary classification problem are shown in the following table with the models that provided results:

Figure 21: Result comparison table.

Platform and function	Models	Explanatory variables	AUC	Logarithmic loss	AUCPR	False negative rate (Training data set)	False positive rate (Training data set)	False negative rate (Test data set)	False positive rate (Test data set)
R Studio with H2O AutoML function	GBM_5_AutoML_20210504_081342	35	0.872	0.362	0.668	44/226 = 19%	35/674 = 5%	11/21 = 52%	9/79 = 11%
H2O Flow with H2O AutoML function	GBM_grid_1_AutoML_20210522_124428_model_2	35	0.871	0.375	0.686	22/226 = 10%	10/674 = 1%	10/21 = 48%	8/79 = 10%
R Studio with GLM function	Linear model - Logistic regression	2	0.824	4.682	0.125	58/226 = 26%	64/674 = 9%	8/21 = 38%	7/79 = 9%
R Studio with Neuralnet function	Multilayer perceptron	33	0.500	0.564	0.000	226/226 = 100%	0/674 = 0%	21/21 = 100%	0/79 = 0%
R Studio with SVM function	Support Vector Machine	35	0.973	0.652	0.027	10/226 = 4%	7/674 = 1%	Unable	Unable
R Studio with Tree function	Decision Tree	33	0.856	3.799	0.109	48/226 = 21%	51/674 = 8%	6/21 = 29%	9/79 = 11%

Source: own elaboration

Metric values are filled with dark green colour (best outcome), medium green colour (second best outcome) and light green colour (third best outcome). As previously detailed, the metric definitions are the following:

- **Area Under the Curve (AUC):** probability value that measures the ability to of the classifier to distinguish between fraudulent claims and non-fraudulent claims.
- **Logarithmic loss:** value that indicates how close the predicted values generated by the model are to the actual target value, i.e. it evaluates the correctness of a classifier by penalising false classifications.
- **Area Under the Precision-Recall Curve (AUCPR):** performance metric that allows the visualisation of performance that represents a trade-off between the true positive rate and the positive predictive value.
- **Confusion matrix N (error):** percentage of claims that would be considered non-fraudulent claims by the model although they actually are fraudulent claims.
- **Confusion matrix Y (error):** percentage of claims that would be considered as fraudulent claims by the model but in reality they are not fraudulent claims.
- **False positive rate** is the proportion of all non-fraudulent claims observed that model informs that are modelled as fraudulent claims.
- **False negative rate** is the proportion of all fraudulent claims observed that the model informs that are modelled as non-fraudulent claims.

Two models are considered in the previous grid although they should not be. The multilayer perceptron model does predict no fraudulent claims at all. In addition, the logistic regression model only uses two categorical variables of the original data set after manually selecting the model with the lowest AIC metric and more coefficients statistically significant. Although the Support Vector Machines seems to be a model with

⁴ Several metrics and statistics are available in academia. The metrics mentioned in this thesis are not considered an exhaustive list of all the available metrics but just a subset of the most adequate ones for a binary classification problem.

good metrics, due to fact that the number of levels in the “auto_model” variable differs between the train and the test data sets, R Studio indicates an error and no predictions can be made with the test data set, being forced to modify or discard data once again to make it work.

Assuming that all the above metrics are considered equally important, the below table represents points⁵ given to each model considering the outcome obtained:

Figure 22: Result comparison table.

Platform and function	Models	Explanatory variables	Points
R Studio with H2O AutoML function	GBM_5_AutoML_20210504_081342	35	29
H2O Flow with H2O AutoML function	GBM_grid_1_AutoML_20210522_124428_model_2	35	32
R Studio with GLM function	Linear model - Logistic regression	2	20
R Studio with Neuralnet function	Multilayer perceptron	33	21
R Studio with SVM function	Support Vector Machine	35	24
R Studio with Tree function	Decision Tree	33	22

Source: own elaboration

Following the colour convention of the table of Figure 21, Figure 22 shows that the GBM models automatically selected by the AutoML function obtained the highest points with the ranking performed. The H2O-Flow GBM model provided the highest AUCPR value and R-Studio GBM model obtained the lowest Logarithmic loss value. Additionally, the GBM models obtained with the AutoML function are not only the most comfortable models to apply with the raw data set utilised but also the models with the best metrics as seen in Figure 21. AutoML function identifies variable structure automatically, with no data feature manipulation, and has no need to discard observations with data content not fully complete or unspecified content, such as “?” symbol. The analysis performed with different ML approaches reflects the convenience, the flexibility and the accuracy of the H2O AutoML function over other approaches.

⁵ Given the 6 final models, 6 points are given to the best metric and 1 point is given to the worst metric, considering the better the metric the higher the point.

7. Conclusions

According to Swiss Re Institute (2020), ML offers potential value to insurers and data scientist trying to leverage big data to better make predictions and obtain conclusions from data. Business leaders are also realising that the evolution of their organisation cannot be further understood through consultation but with hidden patterns and anomalies buried in data, improving human being capabilities. The advantage of ML is that several models can be used to predict results, but the real trick is to ensure that data scientists and actuaries apply the correct algorithms, consider the most appropriate data (correct and well-structured) and use the models that offer the best performance. So, models can continuously be trained to learn from results by learning from data. Automating this modelling process, training, testing and validating models allows users to generate accurate predictions to drive business evolution. Moreover, data can nowadays be stored in an efficient and cost-effective way, both locally and in the cloud, considering network reliability and speed discard any limitation related to handling large amounts of data.

The topic of this master thesis was decided to be on AutoML capabilities because it may help to drive the insurance sector and academics of the insurance environment towards the comfort zone on this matter. The rule of thumb may indicate that the more complex the data utilised, the more complex the algorithm should be, but there are so many models to try with their particular parameterisation and no much time to try them all⁶. This paper presents the current situation of the ML in the insurance sector, the capabilities of the ML to meet the needs of the insurance sector to evolve and also shows how a practical case is easy to be applied to demonstrate and justify that AutoML obtains good results with little effort in a binary classification practical case, especially with intuitive and easy-to-use web interfaces such as H2O Flow. Users could leverage H2O to select the most promising algorithms by just taking a look at the data and apply the ensemble techniques automatically. Consequently, the user can spend time up front in the process analysing the problem. In addition, H2O AutoML models can be taken as good-performer prototypes to be manually enhanced later on amending their parameterisation if desired.

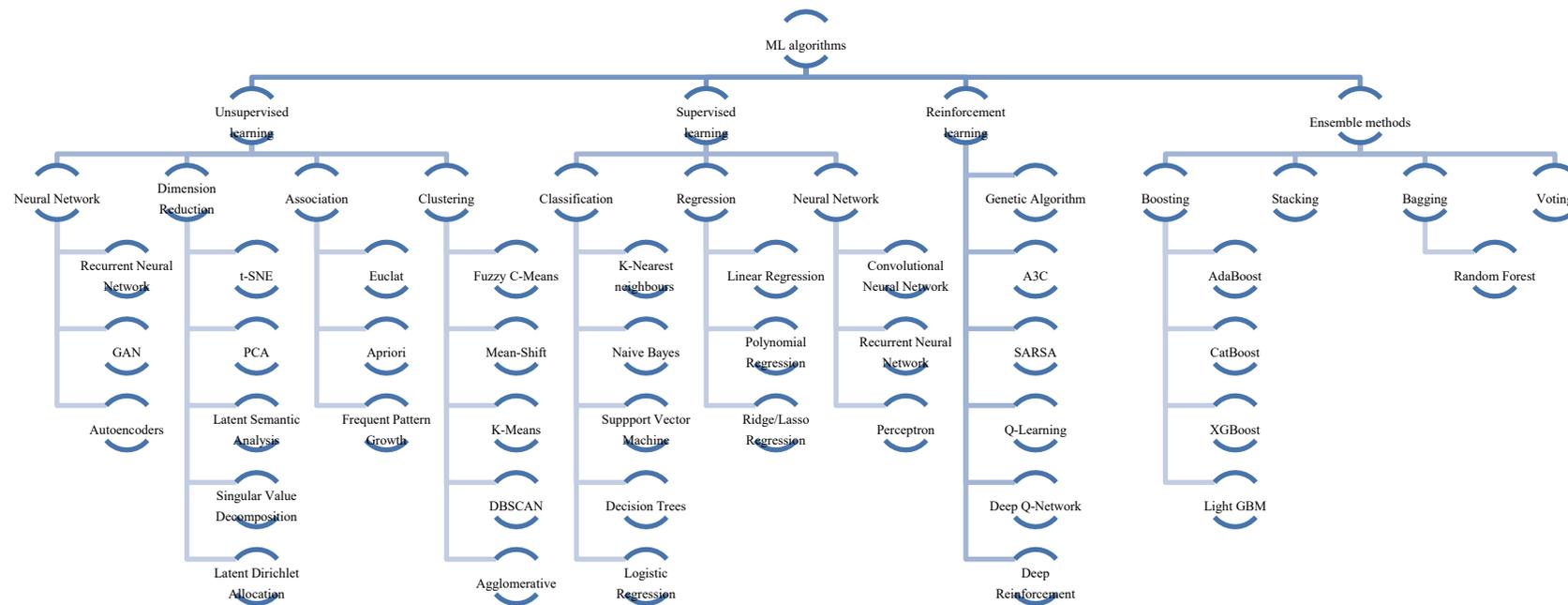
The practical case of this thesis demonstrates that users of ML do not need Big Data to use ML techniques but, it is true that with Big Data the precision of ML algorithms can be improved. Given the promising results by automatically selecting the GBM model for the binary classification problem, multi-class classification and regression process may be the next challenge to test AutoML's capabilities. Particularities of this practical case indicate that AutoML may mean finding the best model in the least amount of time and effort, but it cannot be forgotten that the best algorithm will perform badly with a wrong dataset. In addition, issues faced during the implementation of the practical case are: impossibility to install H2O requirements in Python, impossibility to connect to H2O's server from Macintosh operating systems, impossibility to converge a perceptron model and additional data manipulation and model setting when not using AutoML function.

⁶ It is acknowledged that several type of models with different parameterisation may be tested for the purpose considered in this thesis. Nevertheless, limitation on time and number of pages to detail this practical case are considered a constraint to obtain further conclusions. Therefore, an additional practical case may continue the comparison initiate with this paper.

8. Annex A - Images and Outputs

Annex A.1. Type of Machine Learning algorithms.

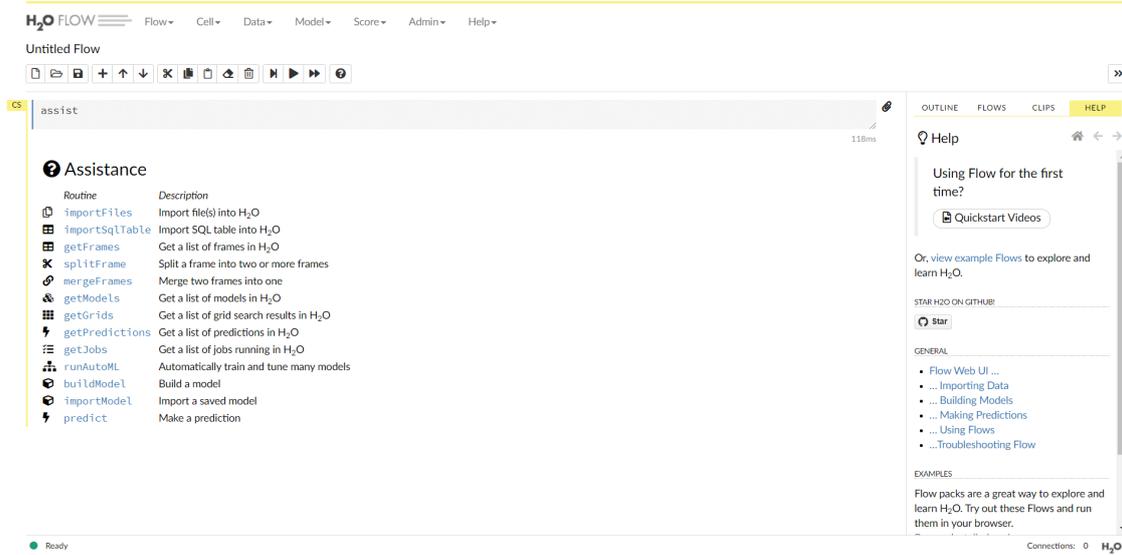
Figure 23: Type of ML algorithms.



Source: own elaboration from Wakefield, K. (2021), Nilsson, N. J. (1998) and Burger, S.V. (2018)

Annex A.2. H2O Flow interface

Figure 24: H2O Flow interface.



Source: H2O (2019) Scalable AutoML in H2O

Annex A.3. Description of variables

Figure 25: Description of variables.

VARIABLE	DESCRIPTION
months_as_customer	months as policyholder
Age	policyholder's years of age
policy_number	policy number
policy_bind_date	date when the insurance policy coverage is in place
policy_state	state of the United States of America where the insurance policy was signed
policy_csl	combined single limit of the insurance policy
policy_deductable	deductible amount of the insurance policy
policy_annual_premium	annual insurance premium in dollars
umbrella_limit	umbrella insurance policy
insured_zip	zip code where the policyholder lives
insured_sex	insured's sex
insured_education_level	level of education of the insured
insured_occupation	occupation of the insured
insured_hobbies	hobbies of the insured
insured_relationship	family situation of the insured
capital-gains	policyholder's capital gains over the last year
capital-loss	policyholder's capital loss over the last year
incident_date	incident date
incident_type	incident type description
collision_type	collision type description
incident_severity	incident severity description
authorities_contacted	authorities contacted at the time of the incident
incident_state	state where the incident occurred
incident_city	city where the incident occurred
incident_location	address where the incident occurred
incident_hour_of_the_day	hour of the day when the incident occurred with 24-hour clock
number_of_vehicles_involved	number of vehicles involved in the incident
property_damage	yes-no indicator of property damage
bodily_injuries	number of bodily injuries
witnesses	number of witnesses of the incident
police_report_available	yes-no indicator of police report
total_claim_amount	total claim amount in dollars
injury_claim	claim amount related to bodily injuries
property_claim	claim amount related to property
vehicle_claim	claim amount related to vehicle damages
auto_make	vehicle make

auto_model	vehicle model
auto_year	year of vehicle registration
fraud_reported	Y-N indicator of fraudulent claim reported

Source: own elaboration from Sharma, R. (2019)

Annex A.4. Data structure

Figure 26: Data structure.

```
> str(rawdata)
'data.frame': 1000 obs. of 39 variables:
 $ months_as_customer : int 328 228 134 256 228 256 137 165 27 212 ...
 $ age : int 48 42 29 41 44 39 34 37 33 42 ...
 $ policy_number : int 521585 342868 687698 227811 367455 104594 413978 429027 485665 636550 ...
 $ policy_bind_date : chr "17-10-2014" "27-06-2006" "06-09-2000" "25-05-1990" ...
 $ policy_state : chr "OH" "IN" "OH" ...
 $ policy_cs1 : chr "250/500" "250/500" "100/300" "250/500" ...
 $ policy_deductable : int 1000 2000 2000 2000 1000 1000 1000 500 500 ...
 $ policy_annual_premium : num 1407 1197 1413 1416 1584 ...
 $ umbrella_limit : int 0 5000000 5000000 6000000 6000000 0 0 0 0 ...
 $ insured_zip : int 466132 468176 430632 608117 610706 478456 441716 603195 601734 600983 ...
 $ insured_sex : chr "MALE" "MALE" "FEMALE" "FEMALE" ...
 $ insured_education_level : chr "MD" "MD" "PhD" "PhD" ...
 $ insured_occupation : chr "craft-repair" "machine-op-inspct" "sales" "armed-forces" ...
 $ insured_hobbies : chr "sleeping" "reading" "board-games" "board-games" ...
 $ insured_relationship : chr "husband" "other-relative" "own-child" "unmarried" ...
 $ capital_gains : int 53300 0 35100 48900 66000 0 0 0 0 ...
 $ capital_loss : int 0 0 0 -62400 -46000 0 -77000 0 0 -39300 ...
 $ incident_date : chr "25-01-2015" "21-01-2015" "22-02-2015" "10-01-2015" ...
 $ incident_type : chr "single vehicle collision" "vehicle theft" "Multi-vehicle collision" "Single vehicle collision" ...
 $ collision_type : chr "side collision" "2" "Rear collision" "Front collision" ...
 $ incident_severity : chr "Major Damage" "Minor Damage" "Minor Damage" "Major Damage" ...
 $ authorities_contacted : chr "PC" "Police" "Police" "Police" ...
 $ incident_state : chr "SC" "VA" "NY" "OH" ...
 $ incident_city : chr "Columbus" "Riverwood" "Columbus" "Arlington" ...
 $ incident_location : chr "9935 4th Drive" "6608 MLK Hwy" "7121 Francis Lane" "6956 Maple Drive" ...
 $ incident_hour_of_the_day : int 5 8 7 5 20 19 0 23 21 14 ...
 $ number_of_vehicles_involved : int 1 1 3 1 1 3 3 3 1 1 ...
 $ property_damage : chr "YES" "2" "NO" "2" ...
 $ bodily_injuries : int 1 0 2 1 0 0 0 2 1 2 ...
 $ witnesses : int 2 0 3 2 1 2 0 2 1 1 ...
 $ police_report_available : chr "YES" "2" "NO" "NO" ...
 $ total_claim_amount : int 71610 5070 34650 63400 6500 64100 78650 51590 27700 42300 ...
 $ injury_claim : int 6510 780 7700 6340 1300 6410 21450 9380 2770 4700 ...
 $ property_claim : int 13020 780 3850 6340 650 6410 7150 9380 2770 4700 ...
 $ vehicle_claim : int 52080 3510 23100 50720 4550 51280 50050 32830 22160 32900 ...
 $ auto_make : chr "Saab" "Mercedes" "dodge" "Chevrolet" ...
 $ auto_model : chr "92x" "E400" "RAM" "Tahoe" ...
 $ auto_year : int 2004 2007 2007 2014 2009 2003 2012 2015 2012 1996 ...
 $ fraud_reported : chr "Y" "Y" "N" "Y" ...
```

Source: own elaboration with R

Annex A.5. Data summary

Figure 27: Data summary.

```
> summary(rawdata)
 months_as_customer  age  policy_number  policy_bind_date  policy_state  policy_cs1  policy_deductable  policy_annual_premium
Min. : 0.0  Min. :19.00  Min. :100804  Length:1000  Length:1000  Length:1000  Length:1000  Min. : 500  Min. : 433.3
1st Qu.:115.8  1st Qu.:32.00  1st Qu.:335980  Class :character  Class :character  Class :character  Class :character  1st Qu.: 500  1st Qu.:1089.6
Median :199.5  Median :38.00  Median :533135  Mode :character  Mode :character  Mode :character  Mode :character  Median :1000  Median :1237.2
Mean :204.0  Mean :38.95  Mean :546239  Mode :character  Mode :character  Mode :character  Mode :character  Mean :1136  Mean :1256.4
3rd Qu.:276.2  3rd Qu.:44.00  3rd Qu.:759100  Mode :character  Mode :character  Mode :character  Mode :character  3rd Qu.:2000  3rd Qu.:1415.7
Max. :479.0  Max. :64.00  Max. :999435  Mode :character  Mode :character  Mode :character  Mode :character  Max. :2000  Max. :2047.6
 umbrella_limit  insured_zip  insured_sex  insured_education_level  insured_occupation  insured_hobbies  insured_relationship
Min. : -10000000  Min. :430104  Length:1000  Length:1000  Length:1000  Length:1000  Length:1000
1st Qu.: 0  1st Qu.:448405  Class :character  Class :character  Class :character  Class :character  Class :character
Median : 0  Median :466446  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character
Mean : 1101000  Mean :501215  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character
3rd Qu.: 0  3rd Qu.:603251  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character
Max. :10000000  Max. :620962  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character
 capital_gains  capital_loss  incident_date  incident_type  collision_type  incident_severity  authorities_contacted  incident_state
Min. : 0  Min. : -111100  Length:1000  Length:1000  Length:1000  Length:1000  Length:1000  Length:1000
1st Qu.: 0  1st Qu.: -51500  Class :character  Class :character  Class :character  Class :character  Class :character  Class :character
Median : 0  Median : -23250  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character
Mean : 25126  Mean : -26794  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character
3rd Qu.: 51025  3rd Qu.: 0  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character
Max. :100500  Max. : 0  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character
 incident_city  incident_location  incident_hour_of_the_day  number_of_vehicles_involved  property_damage  bodily_injuries  witnesses
Length:1000  Length:1000  Min. : 0.00  Min. :1.000  Length:1000  Min. :0.000  Min. :0.000
Class :character  Class :character  1st Qu.: 6.00  1st Qu.:1.000  Class :character  1st Qu.:0.000  1st Qu.:1.000
Mode :character  Mode :character  Median :12.00  Median :1.000  Mode :character  Median :1.000  Median :1.000
Mean :11.64  Mean :1.839  Mean :11.64  Mean :1.839  Mean :0.992  Mean :1.487
3rd Qu.:17.00  3rd Qu.:3.000  3rd Qu.:17.00  3rd Qu.:3.000  3rd Qu.:2.000  3rd Qu.:2.000
Max. :23.00  Max. :4.000  Max. :23.00  Max. :4.000  Max. :2.000  Max. :3.000
 police_report_available  total_claim_amount  injury_claim  property_claim  vehicle_claim  auto_make  auto_model  auto_year
Length:1000  Min. : 100  Min. : 0  Min. : 0  Min. : 70  Length:1000  Length:1000  Min. :1995
Class :character  1st Qu.: 41813  1st Qu.: 4295  1st Qu.: 4445  1st Qu.:30293  Class :character  Class :character  1st Qu.:2000
Mode :character  Median : 58055  Median : 6775  Median : 6750  Median :42100  Mode :character  Mode :character  Median :2005
Mean : 52762  Mean : 7433  Mean : 7400  Mean :37929  Mean :0.992  Mean :1.487
3rd Qu.: 70593  3rd Qu.:11305  3rd Qu.:10885  3rd Qu.:50823  3rd Qu.:2.000  3rd Qu.:2.000
Max. :114920  Max. :21450  Max. :23670  Max. :79560  Max. :2.000  Max. :3.000
 fraud_reported
Length:1000
Class :character
Mode :character
```

Source: own elaboration with R

Annex A.6. Descriptive statistics for numeric variables

Figure 28: Descriptive statistics for numeric variables.

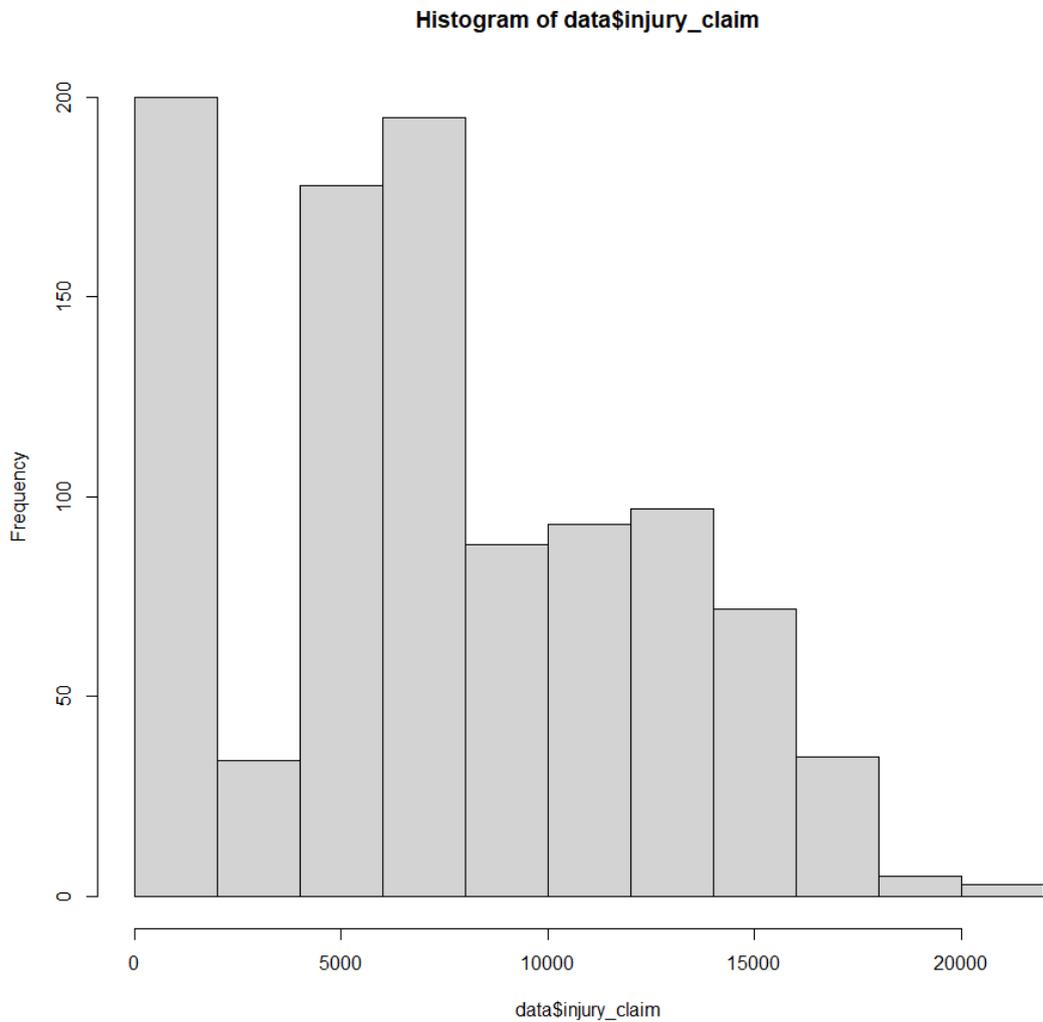
```
> numeric_cols <- unlist(lapply(data, is.numeric))
> describe(data[numeric_cols])
```

vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
months_as_customer	1 1000	203.95	115.11	199.5	198.16	119.35	0.00	479.00	479.00	0.36	-0.49	3.64
age	2 1000	38.95	9.14	38.0	38.36	8.90	19.00	64.00	45.00	0.48	-0.27	0.29
policy_number	3 1000	546238.65	257063.01	533135.0	545338.32	312790.05	100804.00	999435.00	898631.00	0.04	-1.14	8129.05
policy_deductable	4 1000	1136.00	611.86	1000.0	1107.50	741.30	500.00	2000.00	1500.00	0.48	-1.38	19.35
policy_annual_premium	5 1000	1256.41	244.17	1257.2	1255.97	243.53	433.33	2047.59	1614.26	0.00	0.06	7.72
umbrella_limit	6 1000	1102000.00	2296708.70	0.0	542500.00	0.00	0.00	10000000.00	10000000.00	1.80	1.77	72628.31
insured_zip	7 1000	501214.49	71701.61	466445.5	495162.84	32381.47	430104.00	620962.00	190858.00	0.81	-1.19	2267.40
capital_gains	8 1000	25126.10	27872.19	0.0	22236.88	0.00	0.00	100500.00	100500.00	0.48	-1.28	881.40
capital_loss	9 1000	-26793.70	28104.10	-23250.0	-24170.38	34470.45	-111100.00	0.00	111100.00	-0.39	-1.32	888.73
incident_hour_of_the_day	10 1000	11.64	6.95	12.0	11.67	8.90	0.00	23.00	23.00	-0.04	-1.20	0.22
number_of_vehicles_involved	11 1000	1.84	1.02	1.0	1.76	0.00	1.00	4.00	3.00	0.50	-1.50	0.03
bodily_injuries	12 1000	0.99	0.82	1.0	0.99	1.48	0.00	2.00	2.00	0.01	-1.51	0.03
witnesses	13 1000	1.49	1.11	1.0	1.48	1.48	0.00	3.00	3.00	0.02	-1.35	0.04
total_claim_amount	14 1000	52761.94	26401.53	58055.0	54093.43	20541.42	100.00	114920.00	114820.00	-0.59	-0.46	834.89
injury_claim	15 1000	7433.42	4880.95	6775.0	7234.66	5493.03	0.00	21450.00	21450.00	0.26	-0.77	154.35
property_claim	16 1000	7399.57	4824.73	6750.0	7164.84	4877.75	0.00	23670.00	23670.00	0.38	-0.39	152.57
vehicle_claim	17 1000	37928.95	18886.25	42100.0	38944.29	14588.78	70.00	79560.00	79490.00	-0.62	-0.46	597.24
auto_year	18 1000	2005.10	6.02	2005.0	2005.14	7.41	1995.00	2015.00	20.00	-0.05	-1.18	0.19

Source: own elaboration with R

Annex A.7. Injury claim variable histogram

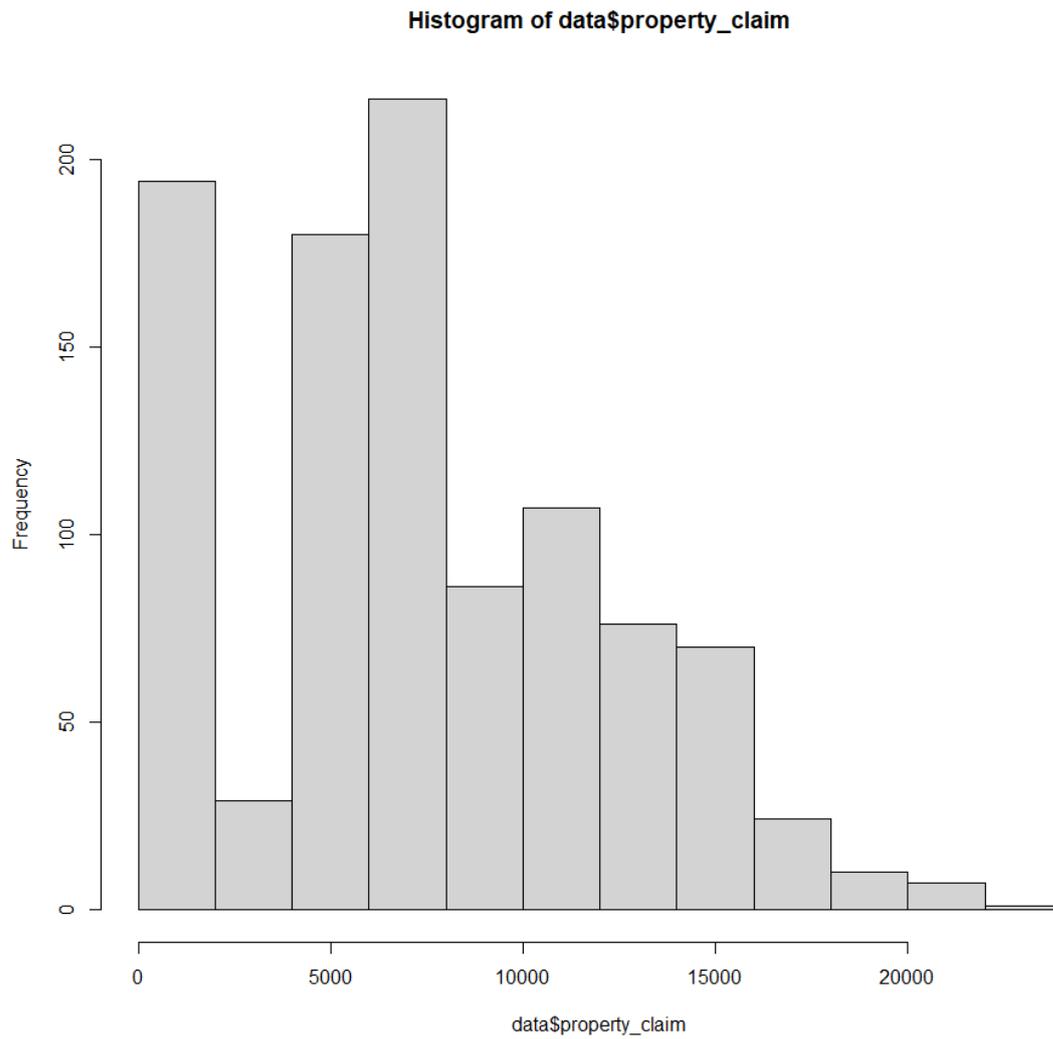
Figure 29: Injury claim variable histogram.



Source: own elaboration with R

Annex A.8. Property claim variable histogram

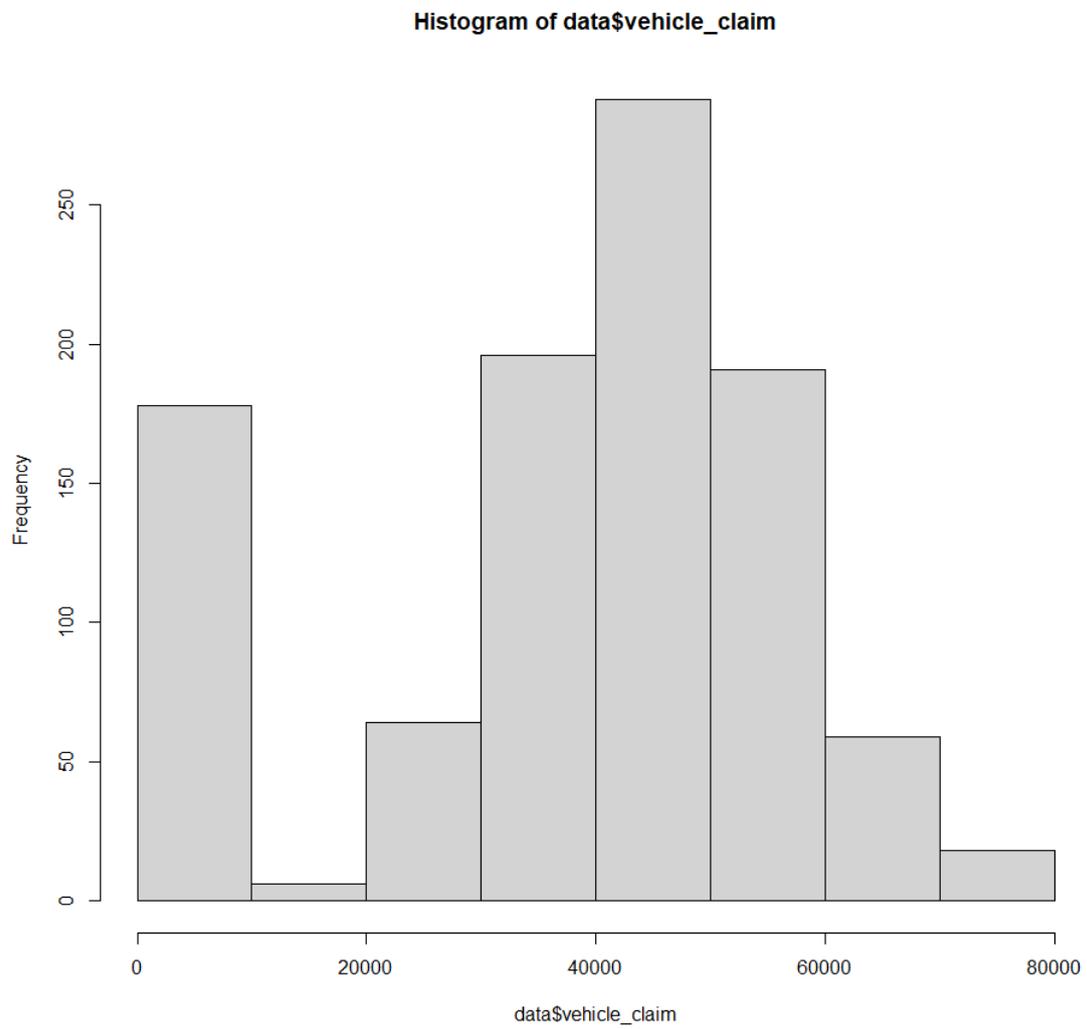
Figure 30: Property claim variable histogram.



Source: own elaboration with R

Annex A.9. Vehicle claim variable histogram

Figure 31: Vehicle claim variable histogram.



Source: own elaboration with R

Annex A.10. Descriptive analysis segmenting by Y-N fraud reported variable

Figure 32: Descriptive analysis segmenting by Y-N fraud reported variable.

```
> describeBy(data[numeric_cols], group=data$Fraud_reported)
```

Descriptive statistics by group

group:		vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
group: N	months_as_customer	1	753	202.60	113.57	200.00	196.72	118.61	0.00	479.00	479.00	0.37	-0.45	4.14
	age	2	753	38.88	8.97	38.00	38.32	8.90	20.00	64.00	44.00	0.47	-0.23	0.33
	policy_number	3	753	550571.30	257323.90	543610.00	551215.84	317326.81	100804.00	998865.00	898061.00	-0.01	-1.16	9377.40
	policy_deductable	4	753	1130.81	606.77	1000.00	1101.16	741.30	500.00	2000.00	1500.00	0.50	-1.34	22.11
	policy_annual_premium	5	753	1258.43	241.25	1253.12	1258.30	241.80	433.33	2047.59	1614.26	0.00	0.02	8.79
	umbrella_limit	6	753	1025232.40	2224297.82	0.00	466003.32	0.00	0.00	10000000.00	10000000.00	1.89	2.07	81057.92
	insured_zip	7	753	500419.54	72123.98	465674.00	494241.71	33109.42	430104.00	620962.00	190858.00	0.82	-1.19	2628.34
	capital_gains	8	753	25432.01	27918.46	0.00	22667.00	0.00	0.00	100500.00	100500.00	0.45	-1.32	1017.41
	capital_loss	9	753	-26554.58	28280.49	-20400.00	-23833.50	30245.04	-111100.00	0.00	111100.00	-0.42	-1.29	1030.60
	incident_hour_of_the_day	10	753	11.63	6.98	12.00	11.65	8.90	0.00	23.00	23.00	-0.02	-1.21	0.25
	number_of_vehicles_involved	11	753	1.81	1.01	1.00	1.73	0.00	1.00	4.00	3.00	0.54	-1.48	0.04
	bodily_injuries	12	753	0.98	0.82	1.00	0.97	1.48	0.00	2.00	2.00	0.04	-1.50	0.03
	witnesses	13	753	1.46	1.12	1.00	1.44	1.48	0.00	3.00	3.00	0.07	-1.37	0.04
	total_claim_amount	14	753	50288.61	27575.19	56520.00	51104.16	23128.56	100.00	114920.00	114820.00	-0.47	-0.80	1004.90
	injury_claim	15	753	7179.23	4961.20	6620.00	6925.26	6049.01	0.00	21450.00	21450.00	0.30	-0.82	180.80
	property_claim	16	753	7018.88	4828.92	6560.00	6743.81	5544.92	0.00	23670.00	23670.00	0.39	-0.51	175.98
	vehicle_claim	17	753	36090.49	19698.05	41220.00	36751.46	16397.56	70.00	79560.00	79490.00	-0.49	-0.81	717.84
	auto_year	18	753	2005.08	6.00	2005.00	2005.10	7.41	1995.00	2015.00	20.00	-0.03	-1.18	0.22

group:	Y													
group: Y	months_as_customer	1	247	208.08	119.82	199.00	202.94	123.06	3.00	478.00	475.00	0.33	-0.65	7.62
	age	2	247	39.14	9.65	38.00	38.54	8.90	19.00	63.00	44.00	0.49	-0.42	0.61
	policy_number	3	247	533030.21	256334.03	516959.00	527366.52	293227.15	104594.00	999435.00	894841.00	0.19	-1.06	16310.14
	policy_deductable	4	247	1151.82	628.12	1000.00	1128.14	741.30	500.00	2000.00	1500.00	0.41	-1.50	39.97
	policy_annual_premium	5	247	1250.24	253.26	1269.64	1248.67	248.80	484.67	1935.85	1451.18	0.03	0.13	16.11
	umbrella_limit	6	247	1336032.39	2494798.91	0.00	809045.23	0.00	0.00	10000000.00	10000000.00	1.55	0.95	158740.25
	insured_zip	7	247	503637.96	70487.50	469853.00	498235.83	28941.83	430141.00	620819.00	190678.00	0.80	-1.22	4485.01
	capital_gains	8	247	24193.52	27766.25	0.00	21065.33	0.00	0.00	91900.00	91900.00	0.56	-1.17	1766.72
	capital_loss	9	247	-27522.67	27603.23	-30200.00	-25277.89	44774.52	-91200.00	0.00	91200.00	-0.29	-1.42	1756.33
	incident_hour_of_the_day	10	247	11.70	6.89	12.00	11.74	8.90	0.00	23.00	23.00	-0.09	-1.17	0.44
	number_of_vehicles_involved	11	247	1.93	1.05	1.00	1.86	0.00	1.00	4.00	3.00	0.37	-1.57	0.07
	bodily_injuries	12	247	1.04	0.83	1.00	1.05	1.48	0.00	2.00	2.00	-0.08	-1.55	0.05
	witnesses	13	247	1.58	1.07	2.00	1.60	1.48	0.00	3.00	3.00	-0.12	-1.23	0.07
	total_claim_amount	14	247	60302.11	20746.28	61290.00	61933.32	15404.21	2860.00	112320.00	109460.00	-0.81	1.31	1320.05
	injury_claim	15	247	8208.34	4550.31	7240.00	8171.21	4180.93	0.00	20700.00	20700.00	0.22	-0.57	289.53
	property_claim	16	247	8560.12	4631.74	7440.00	8419.40	3914.06	0.00	21810.00	21810.00	0.44	-0.03	294.71
	vehicle_claim	17	247	43533.64	14849.39	44800.00	44687.54	11000.89	2080.00	77760.00	75680.00	-0.83	1.40	944.84
	auto_year	18	247	2005.19	6.07	2006.00	2005.25	7.41	1995.00	2015.00	20.00	-0.09	-1.17	0.39

Source: own elaboration with R

Annex A.11. Connection to H2O Cluster

Figure 33: Connection to H2O Cluster.

```
> localH2O = h2o.init()

H2O is not running yet, starting it now...

Note: In case of errors look at the following log files:
C:\Users\svalle\AppData\Local\Temp\RtmpYJTU9d\file2af060557a0/h2o_svalle_started_from_r.out
C:\Users\svalle\AppData\Local\Temp\RtmpYJTU9d\file2af06a92309/h2o_svalle_started_from_r.err

java version "1.8.0_201"
Java(TM) SE Runtime Environment (build 1.8.0_201-b09)
Java HotSpot(TM) Client VM (build 25.201-b09, mixed mode)

Starting H2O JVM and connecting: . Connection successful!

R is connected to the H2O cluster:
H2O cluster uptime: 6 seconds 518 milliseconds
H2O cluster timezone: Europe/Paris
H2O data parsing timezone: UTC
H2O cluster version: 3.32.0.1
H2O cluster version age: 6 months and 25 days !!!
H2O cluster name: H2O_started_from_R_svalle_hmz709
H2O cluster total nodes: 1
H2O cluster total memory: 0.97 GB
H2O cluster total cores: 4
H2O cluster allowed cores: 4
H2O cluster healthy: TRUE
H2O connection ip: localhost
H2O connection port: 54321
H2O connection proxy: NA
H2O Internal security: FALSE
H2O API Extensions: Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4
R Version: R version 4.0.4 (2021-02-15)
```

Source: own elaboration with R

Annex A.12. H2O data frame description

Figure 34: H2O data frame description.

```
> h2o.describe(h2odata)
```

	Label	Type	Missing	Zeros	PosInf	NegInf	Min	Max	Mean	Sigma	Cardinality
1	months_as_customer	int	0	1	0	0	0.00	479.00	203.954	1.151132e+02	NA
2	age	int	0	0	0	0	19.00	64.00	38.948	9.140287e+00	NA
3	policy_number	int	0	0	0	0	100804.00	999435.00	546238.648	2.570630e+05	NA
4	policy_bind_date	string	0	0	0	0	NaN	NaN	NA	NA	NA
5	policy_state	enum	0	338	0	0	0.00	2.00	NA	NA	3
6	policy_csl	enum	0	349	0	0	0.00	2.00	NA	NA	3
7	policy_deductable	int	0	0	0	0	500.00	2000.00	1136.000	6.118647e+02	NA
8	policy_annual_premium	real	0	0	0	0	433.33	2047.59	1256.406	2.441674e+02	NA
9	umbrella_limit	int	0	799	0	0	0.00	1000000.00	1102000.000	2.296709e+06	NA
10	insured_zip	int	0	0	0	0	430104.00	620962.00	501214.488	7.170161e+04	NA
11	insured_sex	enum	0	537	0	0	0.00	1.00	0.463	4.988786e-01	2
12	insured_education_level	enum	0	145	0	0	0.00	6.00	NA	NA	7
13	insured_occupation	enum	0	65	0	0	0.00	13.00	NA	NA	14
14	insured_hobbies	enum	0	49	0	0	0.00	19.00	NA	NA	20
15	insured_relationship	enum	0	170	0	0	0.00	5.00	NA	NA	6
16	capital-gains	int	0	508	0	0	0.00	100500.00	25126.100	2.787219e+04	NA
17	capital-loss	int	0	475	0	0	-111100.00	0.00	-26793.700	2.810410e+04	NA
18	incident_date	enum	0	19	0	0	0.00	59.00	NA	NA	60
19	incident_type	enum	0	419	0	0	0.00	3.00	NA	NA	4
20	collision_type	enum	0	178	0	0	0.00	3.00	NA	NA	4
21	incident_severity	enum	0	276	0	0	0.00	3.00	NA	NA	4
22	authorities_contacted	enum	0	196	0	0	0.00	4.00	NA	NA	5
23	incident_state	enum	0	110	0	0	0.00	6.00	NA	NA	7
24	incident_city	enum	0	152	0	0	0.00	6.00	NA	NA	7
25	incident_location	string	0	0	0	0	NaN	NaN	NA	NA	NA
26	incident_hour_of_the_day	int	0	52	0	0	0.00	23.00	11.644	6.951373e+00	NA
27	number_of_vehicles_involved	int	0	0	0	0	1.00	4.00	1.839	1.018880e+00	NA
28	property_damage	enum	0	360	0	0	0.00	2.00	NA	NA	3
29	bodily_injuries	int	0	340	0	0	0.00	2.00	0.992	8.201272e-01	NA
30	witnesses	int	0	249	0	0	0.00	3.00	1.487	1.111335e+00	NA
31	police_report_available	enum	0	343	0	0	0.00	2.00	NA	NA	3
32	total_claim_amount	int	0	0	0	0	100.00	114920.00	52761.940	2.640153e+04	NA
33	injury_claim	int	0	25	0	0	0.00	21450.00	7433.420	4.880952e+03	NA
34	property_claim	int	0	19	0	0	0.00	23670.00	7399.570	4.824726e+03	NA
35	vehicle_claim	int	0	0	0	0	70.00	79560.00	37928.950	1.888625e+04	NA
36	auto_make	enum	0	68	0	0	0.00	13.00	NA	NA	14
37	auto_model	enum	0	18	0	0	0.00	38.00	NA	NA	39
38	auto_year	int	0	0	0	0	1995.00	2015.00	2005.103	6.015861e+00	NA
39	fraud_reported	enum	0	753	0	0	0.00	1.00	0.247	4.314825e-01	2

Source: own elaboration with R

Annex A.13. Metalearner and base model importance

Figure 35: Metalearner and base model importance.

```
> h2o.varimp(metalearner_10mod)
```

	variable	relative_importance	scaled_importance	percentage
1	GBM_5_AutoML_20210504_081342	1.306821172	1.000000000	0.786606487
2	GBM_1_AutoML_20210504_081342	0.124469783	0.095246225	0.074921299
3	GBM_3_AutoML_20210504_081342	0.074830223	0.057261257	0.045042076
4	XRT_1_AutoML_20210504_081342	0.073860588	0.056519277	0.044458430
5	GBM_2_AutoML_20210504_081342	0.073162883	0.055985383	0.044038465
6	GLM_1_AutoML_20210504_081342	0.008195795	0.006271551	0.004933242
7	GBM_4_AutoML_20210504_081342	0.000000000	0.000000000	0.000000000
8	GBM_grid_1_AutoML_20210504_081342_model_1	0.000000000	0.000000000	0.000000000
9	DRF_1_AutoML_20210504_081342	0.000000000	0.000000000	0.000000000
10	DeepLearning_1_AutoML_20210504_081342	0.000000000	0.000000000	0.000000000

Source: own elaboration with R

Annex A.14. 10-model AutoML Metalearner

Figure 36: 10-model AutoML Metalearner.

```
> print(metalearner_10mod)
Model Details:
=====
H2OBinomialModel: glm
Model ID: metalearner_AUTO_StackedEnsemble_AllModels_AutoML_20210504_081342
GLM Model: summary
  family link regularization number_of_predictors_total number_of_active_predictors
1 binomial logit Elastic Net (alpha = 0.5, lambda = 1.65E-4 ) 10 6
  number_of_iterations training_frame
1 4 levelone_training_StackedEnsemble_AllModels_AutoML_20210504_081342

Coefficients: glm coefficients
      names coefficients standardized_coefficients
1 Intercept -3.203161 -1.596996
2 GBM_5_AutoML_20210504_081342 5.217944 1.306821
3 GBM_2_AutoML_20210504_081342 0.240573 0.073163
4 GBM_1_AutoML_20210504_081342 0.399199 0.124470
5 GBM_3_AutoML_20210504_081342 0.245272 0.074830
6 GBM_4_AutoML_20210504_081342 0.000000 0.000000
7 GBM_grid__1_AutoML_20210504_081342_model_1 0.000000 0.000000
8 GLM_1_AutoML_20210504_081342 0.033078 0.008196
9 DRF_1_AutoML_20210504_081342 0.000000 0.000000
10 XRT_1_AutoML_20210504_081342 0.505755 0.073861
11 DeepLearning_1_AutoML_20210504_081342 0.000000 0.000000

H2OBinomialMetrics: glm
** Reported on training data. **

MSE: 0.1104374
RMSE: 0.3323212
LogLoss: 0.3563454
Mean Per-Class Error: 0.1383695
AUC: 0.8714549
AUCPR: 0.6668571
Gini: 0.7429099
R^2: 0.4127369
Residual Deviance: 641.4217
AIC: 655.4217

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
      N Y Error Rate
N 568 106 0.157270 =106/674
Y 27 199 0.119469 =27/226
Totals 595 305 0.147778 =133/900
```

Source: own elaboration with R

Annex A.15. Variable importance of top model

Figure 37: Variable importance of top model.

```
> h2o.varimp(gbm)
Variable Importances:
  variable relative_importance scaled_importance percentage
1 incident_severity 235.154373 1.000000 0.498031
2 insured_hobbies 154.473007 0.656900 0.327157
3 auto_model 44.449188 0.189021 0.094138
4 insured_occupation 11.406408 0.048506 0.024158
5 auto_make 6.118093 0.026017 0.012957

---
  variable relative_importance scaled_importance percentage
29 incident_city 0.000000 0.000000 0.000000
30 property_damage 0.000000 0.000000 0.000000
31 bodily_injuries 0.000000 0.000000 0.000000
32 police_report_available 0.000000 0.000000 0.000000
33 injury_claim 0.000000 0.000000 0.000000
34 auto_year 0.000000 0.000000 0.000000
```

Source: own elaboration with R

Annex A.16. Top model parameters

Figure 38: Top model parameters.

```

> gbm@parameters
$model_id
[1] "GBM_5_AutoML_20210504_081342"

$training_frame
[1] "automl_training_RTMP_sid_8fe6_26"

$n_folds
[1] 5

$keep_cross_validation_models
[1] FALSE

$keep_cross_validation_predictions
[1] TRUE

$score_tree_interval
[1] 5

$fold_assignment
[1] "Modulo"

$n_trees
[1] 46

$max_depth
[1] 15

$min_rows
[1] 100

$stopping_metric
[1] "logloss"

$stopping_tolerance
[1] 0.03333333

$seed
[1] "2823915740711159846"

$distribution
[1] "bernoulli"

$sample_rate
[1] 0.8

$col_sample_rate
[1] 0.8

$col_sample_rate_per_tree
[1] 0.8

$histogram_type
[1] "UniformAdaptive"

$categorical_encoding
[1] "Enum"

$x
 [1] "months_as_customer"      "age"
 [5] "policy_deductable"       "policy_annual_premium"
 [9] "insured_sex"             "insured_education_level"
[13] "insured_relationship"    "capital_gains"
[17] "collision_type"          "incident_severity"
[21] "incident_city"           "incident_hour_of_the_day"
[25] "bodily_injuries"         "witnesses"
[29] "injury_claim"           "property_claim"
[33] "auto_model"              "auto_year"

                                "policy_state"
                                "umbrella_limit"
                                "insured_occupation"
                                "capital_loss"
                                "authorities_contacted"
                                "number_of_vehicles_involved"
                                "police_report_available"
                                "vehicle_claim"

                                "policy_cs1"
                                "insured_zip"
                                "insured_hobbies"
                                "incident_type"
                                "incident_state"
                                "property_damage"
                                "total_claim_amount"
                                "auto_make"

$y
[1] "fraud_reported"

```

Source: own elaboration with R

Annex A.17. Predictions with the GBM fitted model

Figure 39: Predictions with the GBM fitted model.

```

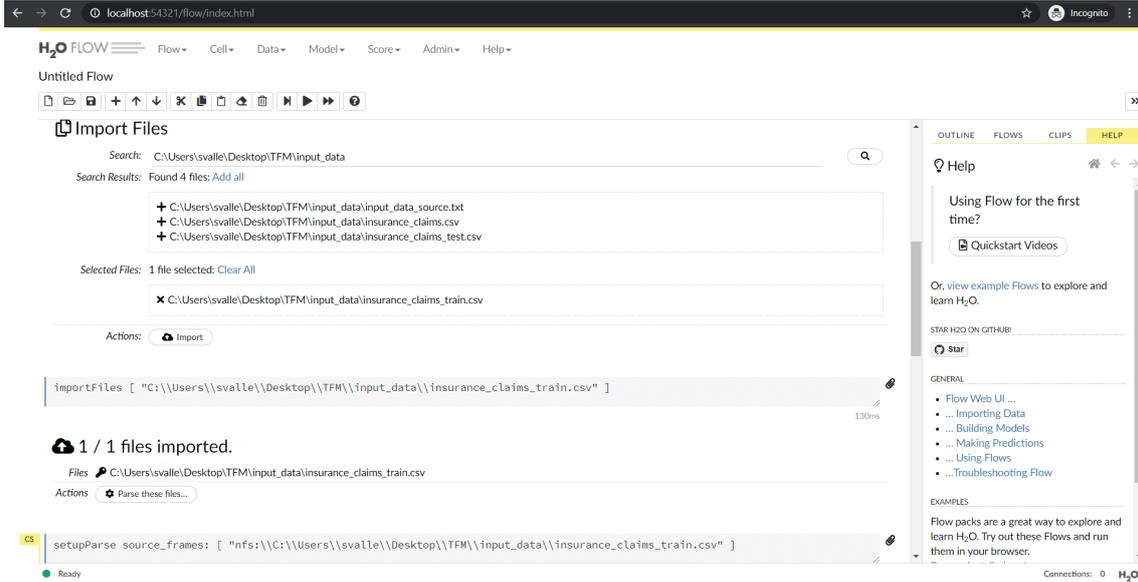
> print(results)
  predict      N      Y      51      52      53      54      55      56      57      58      59      60      61      62      63      64      65      66      67      68      69      70      71      72      73      74      75      76      77      78      79      80      81      82      83      84      85      86      87      88      89      90      91      92      93      94      95      96      97      98      99      100
1      N 0.9532675 0.046732463 51      N 0.9729301 0.027069869
2      N 0.9772417 0.022758300 52      N 0.9507429 0.049257101
3      N 0.9304415 0.069558451 53      N 0.9648447 0.035155309
4      N 0.9630471 0.036952931 54      N 0.9250150 0.074984998
5      N 0.9417953 0.058204713 55      N 0.8581312 0.141868805
6      Y 0.3193648 0.680635218 56      N 0.9749199 0.025080095
7      Y 0.3923041 0.607695944 57      N 0.9560268 0.043973183
8      N 0.9455598 0.054440228 58      N 0.9172357 0.082764275
9      N 0.8084742 0.191525771 59      N 0.6860463 0.313953740
10     N 0.9463868 0.053613236 60      N 0.9629566 0.037043378
11     N 0.9809236 0.019076399 61      N 0.9592435 0.040756535
12     N 0.9784365 0.021563522 62      N 0.9637868 0.036213188
13     N 0.5779127 0.422087319 63      N 0.7986061 0.201393914
14     N 0.9576011 0.042398885 64      N 0.9717902 0.028209788
15     Y 0.1466275 0.853372455 65      Y 0.2763510 0.723649029
16     Y 0.1725790 0.827421030 66      N 0.9847367 0.015263317
17     N 0.8641445 0.135855545 67      N 0.9615845 0.038415544
18     Y 0.1500491 0.849950940 68      N 0.5391671 0.460832943
19     N 0.9782695 0.021730506 69      Y 0.3074722 0.692527768
20     Y 0.2218731 0.778126933 70      N 0.9547100 0.045289988
21     N 0.8849086 0.115091387 71      N 0.9807464 0.019253646
22     N 0.9551487 0.044851291 72      N 0.9582529 0.041747066
23     N 0.9738620 0.026138019 73      N 0.5850954 0.414904598
24     N 0.9549016 0.045098384 74      Y 0.2160706 0.783929435
25     N 0.8962370 0.103762988 75      Y 0.3967149 0.603285111
26     N 0.9622392 0.037760803 76      N 0.9561935 0.043806467
27     N 0.9386895 0.061310487 77      Y 0.4459216 0.554078429
28     N 0.6586450 0.341354960 78      Y 0.2396474 0.760352581
29     N 0.9355462 0.064453753 79      N 0.9292411 0.070758915
30     N 0.4894176 0.510582374 80      N 0.8526204 0.147379603
31     N 0.4869806 0.513019358 81      N 0.9592257 0.040774294
32     N 0.9771656 0.022834407 82      N 0.9784202 0.021579835
33     N 0.7939444 0.206055579 83      Y 0.1926590 0.807340964
34     N 0.9154100 0.084589995 84      N 0.9596335 0.040366525
35     Y 0.4497202 0.550279804 85      N 0.5185385 0.481461521
36     Y 0.3719826 0.628017395 86      N 0.5338602 0.466139772
37     N 0.9679969 0.032003082 87      N 0.9804742 0.019525847
38     N 0.9223446 0.077655423 88      Y 0.2977048 0.702295164
39     N 0.5585569 0.441443059 89      N 0.8929899 0.107010121
40     N 0.6203165 0.379683540 90      N 0.9624318 0.037568211
41     N 0.9642354 0.035764619 91      N 0.9680104 0.031989556
42     Y 0.3727272 0.627272830 92      N 0.9639808 0.036019205
43     N 0.7359237 0.264076297 93      N 0.5710873 0.428912688
44     N 0.9321910 0.067808994 94      N 0.8900570 0.109942986
45     N 0.9683310 0.031668956 95      N 0.9290081 0.070991875
46     N 0.8077944 0.192205632 96      N 0.8811117 0.118888291
47     N 0.4849927 0.515007271 97      Y 0.4176068 0.582393226
48     N 0.9619041 0.038095905 98      N 0.9613040 0.038695954
49     N 0.9602265 0.039773458 99      Y 0.3373016 0.662698378
50     N 0.9901245 0.009875518 100     N 0.9830631 0.016936876

```

Source: own elaboration with R

Annex A.18. Importing data to H2O Flow

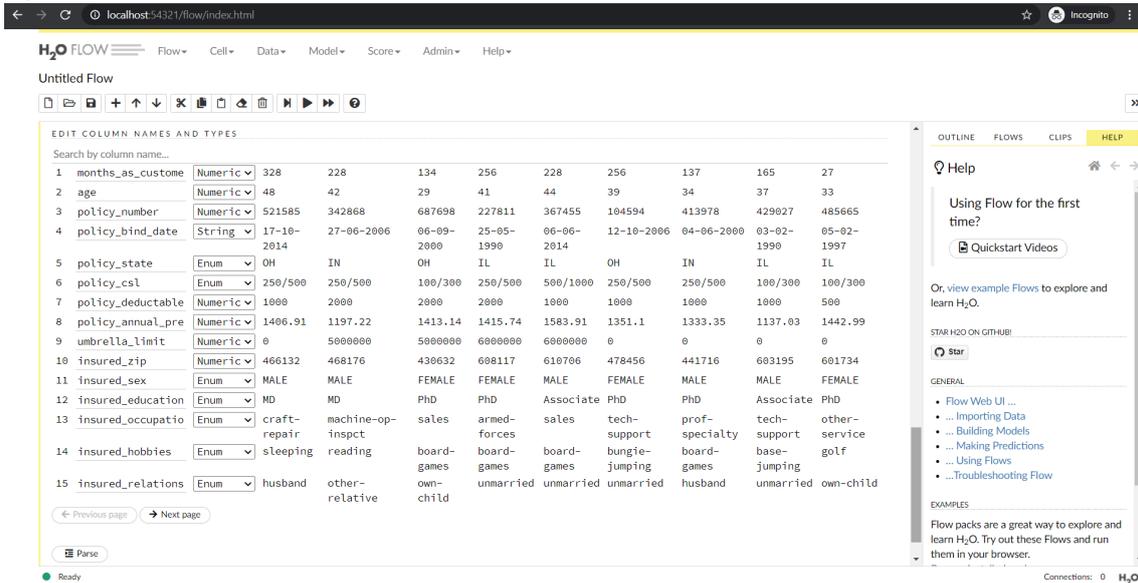
Figure 40: Importing data to H2O Flow.



Source: own elaboration with H2O Flow

Annex A.19. Variables detected with H2O Flow

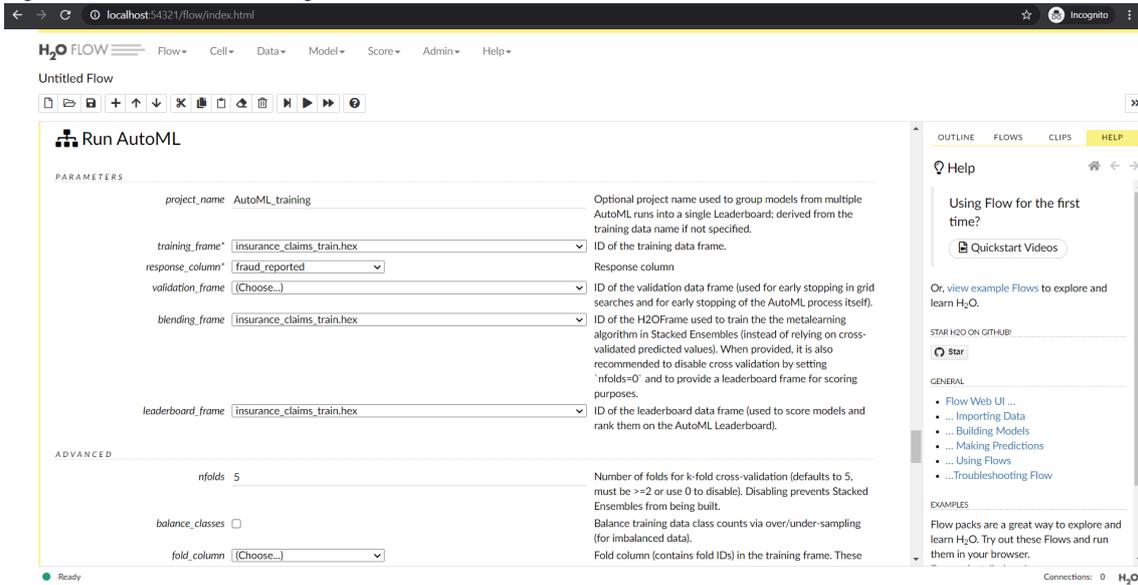
Figure 41: Variables detected with H2O Flow.



Source: own elaboration with H2O Flow

Annex A.20. AutoML function setting with H2O Flow

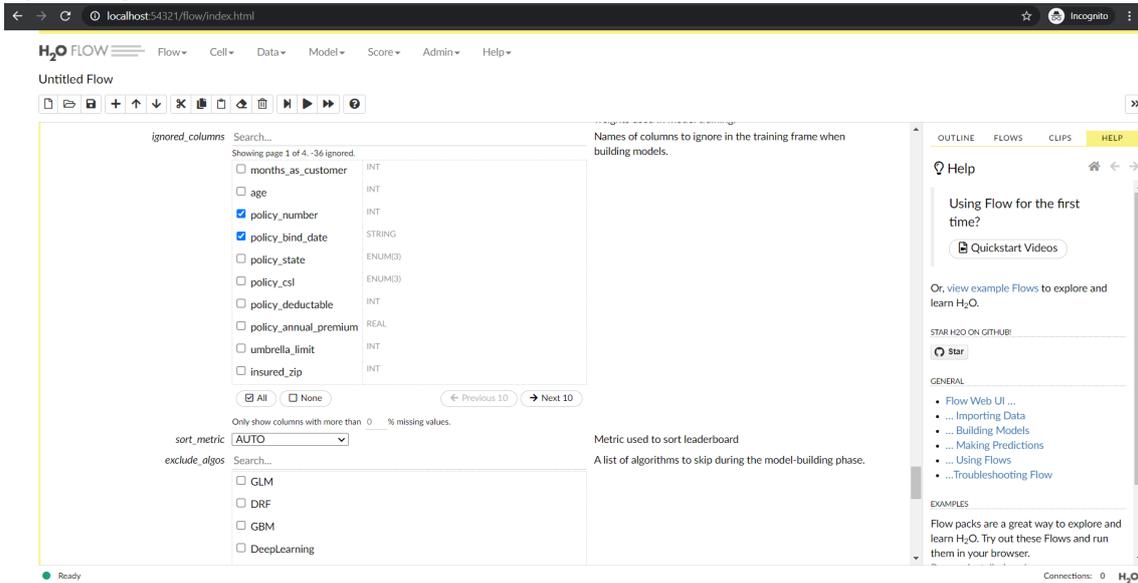
Figure 42: AutoML function setting with H2O Flow.



Source: own elaboration with H2O Flow

Annex A.21. Explanatory variables deselection with H2O Flow

Figure 43: Explanatory variables deselection with H2O Flow.



Source: own elaboration with H2O Flow

Annex A.24. Predictions of GLM fitted model and model accuracy

Figure 46: Predictions of GLM fitted model and model accuracy.

```
> predict.test
901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930
   N   N   N   N   Y   N   N   N   N   N   N   N   N   Y   Y   N   Y   N   Y   N   Y   N   N   N   N   N   N   N   Y
931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960
   N   N   N   N   N   Y   N   N   N   N   N   N   N   N   N   N   N   Y   N   N   N   N   N   N   N   N   N   N   N   N   N
961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990
   N   N   N   N   Y   N   N   Y   N   N   N   N   N   Y   Y   N   Y   Y   N   N   N   N   N   Y   N   Y   Y   N   Y   N   N
991 992 993 994 995 996 997 998 999 1000
   N   N   N   N   N   N   N   N   Y   N

Levels: N Y
> summary(predict.test)
 N Y
80 20
> mean(predict.test == glmtest$fraud_reported)
[1] 0.85
```

Source: own elaboration with R

Annex A.25. Error and weights of the multilayer perceptron

Figure 47: Error and weights of the multilayer perceptron.

```
> mperceptron$result.matrix
              [,1]
error                84.624445748
reached.threshold    0.009109207
steps                44.000000000
Intercept.to.1layhid1 -0.054364739
months_as_customer.to.1layhid1 1.168625322
age.to.1layhid1        0.045682678
policy_state.to.1layhid1 0.959432943
policy_csl.to.1layhid1 0.080815945
policy_deductable.to.1layhid1 0.164780943
policy_annual_premium.to.1layhid1 -0.274322358
umbrella_limit.to.1layhid1 -0.160759392
insured_zip.to.1layhid1 -0.715171609
insured_sex.to.1layhid1 -0.907708776
insured_education_level.to.1layhid1 -1.054342405
insured_occupation.to.1layhid1 -0.369523987
insured_hobbies.to.1layhid1 0.526496716
insured_relationship.to.1layhid1 -0.074596413
capital.gains.to.1layhid1 0.673738076
capital_loss.to.1layhid1 0.080122067
incident_type.to.1layhid1 0.631666205
collision_type.to.1layhid1 0.787669674
incident_severity.to.1layhid1 1.594365250
authorities_contacted.to.1layhid1 -0.574597577
incident_state.to.1layhid1 -1.256291510
incident_city.to.1layhid1 -0.489372805
incident_hour_of_the_day.to.1layhid1 0.323079513
number_of_vehicles_involved.to.1layhid1 0.496308728
property_damage.to.1layhid1 -0.577779865
bodily_injuries.to.1layhid1 -0.318809044
witnesses.to.1layhid1 -1.415051546
police_report_available.to.1layhid1 0.923063493
total_claim_amount.to.1layhid1 -1.783850704
injury_claim.to.1layhid1 1.684882935
property_claim.to.1layhid1 -0.114071054
vehicle_claim.to.1layhid1 -0.693422099
auto_make.to.1layhid1 0.032920928
auto_year.to.1layhid1 -0.213565178
Intercept.to.fraud_reported -1.092408972
1layhid1.to.fraud_reported -0.927746736
```

Source: own elaboration with R

Annex A.26. Predictions of Multilayer perceptron with test data set

Figure 48: Predictions of Multilayer perceptron with test data set.

```

> print(result.test)
      actual prediction
901      0 0.2392616
902      0 0.2997641
903      0 0.2392616
904      0 0.2392616
905      0 0.2392616
906      1 0.2392616
907      0 0.2392616
908      0 0.2392616
909      0 0.2392616
910      0 0.2392616
911      0 0.2392616
912      0 0.2392616
913      1 0.2392616
914      0 0.2392616
915      0 0.2392616
916      1 0.2392616
917      0 0.2392616
918      0 0.2997641
919      0 0.2392616
920      1 0.2392616
921      0 0.2392616
922      0 0.2392616
923      0 0.2997641
924      0 0.2392616
925      0 0.2392616
926      0 0.2392616
927      1 0.2392616
928      1 0.2392616
929      0 0.2392616
930      1 0.2392616
931      1 0.2392616
932      0 0.2392616
933      0 0.2392616
934      0 0.2392616
935      0 0.2997641
936      1 0.2392616
937      0 0.2997641
938      0 0.2392616
939      1 0.2997641
940      1 0.2997641
941      0 0.2997641
942      0 0.2392616
943      0 0.2997641
944      0 0.2392616
945      0 0.2392616
946      0 0.2392616
947      1 0.2997641
948      0 0.2997641
949      0 0.2392616
950      0 0.2997641
951      0 0.2997641
952      0 0.2997641
953      0 0.2392616
954      0 0.2392616
955      0 0.2392616
956      0 0.2392616
957      0 0.2392616
958      0 0.2997641
959      0 0.2392616
960      0 0.2392616
961      0 0.2392616
962      0 0.2392616
963      0 0.2392616
964      0 0.2392616
965      1 0.2997641
966      0 0.2392616
967      0 0.2997641
968      1 0.2997641
969      0 0.2392616
970      0 0.2392616
971      0 0.2392616
972      0 0.2392616
973      1 0.2392616
974      1 0.2392616
975      1 0.2392616
976      0 0.2997641
977      0 0.2392616
978      1 0.2997641
979      0 0.2392616
980      0 0.2392616
981      0 0.2392616
982      0 0.2392616
983      1 0.2392616
984      0 0.2392616
985      0 0.2392616
986      0 0.2392616
987      1 0.2392616
988      1 0.2997641
989      0 0.2392616
990      0 0.2392616
991      0 0.2997641
992      0 0.2392616
993      0 0.2392616
994      0 0.2392616
995      0 0.2392616
996      0 0.2392616
997      0 0.2392616
998      0 0.2997641
999      0 0.2997641
1000     0 0.2392616

```

Source: own elaboration with R

Annex A.27. Support Vector Machine model result

Figure 49: Support Vector Machine model result.

```
> summary(svm)

Call:
svm(formula = fraud_reported ~ ., data = svmtrain, type = "c-classification")

Parameters:
  SVM-Type: C-classification
 SVM-Kernel: radial
      cost: 1

Number of Support Vectors: 519

( 226 293 )

Number of Classes: 2

Levels:
0 1

> summary(svmlin)

Call:
svm(formula = fraud_reported ~ ., data = svmtrain, type = "c-classification", kernel = "linear")

Parameters:
  SVM-Type: C-classification
 SVM-Kernel: linear
      cost: 1

Number of Support Vectors: 479

( 193 286 )

Number of Classes: 2

Levels:
0 1

> summary(svmpol)

Call:
svm(formula = fraud_reported ~ ., data = svmtrain, type = "c-classification", kernel = "polynomial")

Parameters:
  SVM-Type: C-classification
 SVM-Kernel: polynomial
      cost: 1
 degree: 3
coef.0: 0

Number of Support Vectors: 452

( 226 226 )

Number of Classes: 2

Levels:
0 1

> summary(svmsig)

Call:
svm(formula = fraud_reported ~ ., data = svmtrain, type = "c-classification", kernel = "sigmoid")

Parameters:
  SVM-Type: C-classification
 SVM-Kernel: sigmoid
      cost: 1
coef.0: 0

Number of Support Vectors: 510

( 226 284 )

Number of Classes: 2

Levels:
0 1
```

Source: own elaboration with R

9. Bibliography

- Accenture (2018) Remaining insurance processes with intelligent solutions, Source: https://www.accenture.com/_acnmedia/pdf-86/accenture-insurance-intelligent-solutions.pdf
- Accenture (2021) Insurance Revenue Landscape 2025: Innovate for resilience, Accenture, Page 13, Source: <https://www.accenture.com/ca-en/insights/insurance/revenue-landscape-2025-innovate-for-resilience>
- Agresti, A. (2015) Foundations of linear and generalized linear models. John Wiley & Sons
- Amir Hejazi, S. and Jackson, K. R. (2016), A Neural Network Approach to Efficient Valuation of Large Portfolios of Variable Annuities, University of Toronto, Page 2-4, Source: <https://arxiv.org/pdf/1606.07831.pdf>
- Annette, D. (1990) An Introduction to Generalized Linear Models. London: Chapman and Hall.
- Anyoha, R (2017) Artificial Intelligence Timeline, The history of Artificial Intelligence, Blog special edition on artificial intelligence, Harvard University, Source: <https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/>
- AutoGluon (2021) AutoGluon: AutoML for text, image and tabular data, Source: <https://auto.gluon.ai/stable/index.html>
- AXA (2021) La tasa de fraude al seguro en España crece hasta el 2,2%, Actualidad AXA, El blog de AXA, Source: <https://www.axa.es/-/la-tasa-de-fraude-al-seguro-en-espana-crece-hasta-el-2-1#:~:text=La%20tasa%20de%20fraude%20al%20seguro%20detectado%20en%20Espa%C3%B1a%20crece%20hasta%20el%202,2%,AXA%20de%20Fraude%20al%20Seguro.&text=En%202020%20este%20importe%20fue,con%20respecto%20al%20a%C3%B1o%20anterior.>
- Ballings, M. and Van den Poel, D. (2015) Package AUC, Threshold independent performance measures for probabilistic classifiers, Source: <https://cran.r-project.org/web/packages/AUC/AUC.pdf>
- Boral, S. (2019) What is AutoML and Why is it important?, Iottechrends, Source: <https://www.iottechrends.com/what-is-automl/>

- Breiman, L (1996) Stacked regressions, Machine Learning. Source: <https://link.springer.com/article/10.1007/BF00117832>
- Burger, S.V. (2018) Introduction to Machine Learning with R, Rigorous Mathematical Analysis, Page 19, 71, 109 and 135. ISBN: 978-1-491-97644-9
- Chakure, A. (2019) Decision Tree Classification. An introduction to Decision Tree Classifier. Source: <https://medium.com/swlh/decision-tree-classification-de64fc4d5aac#:~:text=Advantages%20of%20Classification%20with%20Decision%20Trees%3A&text=Extremely%20fast%20at%20classifying%20unknown,Excludes%20unimportant%20features.>
- European Commission (2021), A European approach to Artificial Intelligence, Shaping Europe's digital future, Source: <https://digital-strategy.ec.europa.eu/en/policies/european-approach-artificial-intelligence>
- Fernandez-Arjona, L. (2019), A neural network model for solvency calculations in life insurance, University of Zurich, Page 4. Source: <https://arxiv.org/pdf/2005.02318.pdf>
- Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M. and Hutter, F. (2019) Efficient and Robust Automated Machine Learning, University of Freiburg, Germany, Source: <https://ml.informatik.uni-freiburg.de/~staeglis/deploy/papers/15-NIPS-auto-sklearn-preprint.pdf>
- Fritsch, S. Guenther, F., Wright, M. N., Suling, M and Mueller, S. M. (2019), Package Neuralnet, Training of neural networks, Source: <https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf>
- Gartner (2016) Gartner's 2016 hype cycle for emerging technologies identifies three key trends that organizations must track to gain competitive advantage, Gartner, Source: <https://www.gartner.com/en/newsroom/press-releases/2016-08-16-gartners-2016-hype-cycle-for-emerging-technologies-identifies-three-key-trends-that-organizations-must-track-to-gain-competitive-advantage>
- Globe Newswire (2021), Beam Raises \$80 Million in Series E Funding to Further Modernize Dental Benefits. Intrado GlobeNewswire. Source: <https://www.globenewswire.com/news-release/2021/03/02/2185004/0/en/Beam-Raises-80-Million-in-Series-E-Funding-to-Further-Modernize-Dental-Benefits.html>
- Google (2021) AutoML Tables, Source: <https://cloud.google.com/automl-tables/docs?hl=es-419>

- Google Trends (2021) Insurance AI, Source: <https://trends.google.com/trends/explore?date=2012-01-07%202021-01-07&geo=US&q=insurance%20ai>
- H2O (2019) Scalable AutoML in H2O. Source: <https://www.h2o.ai/blog/scalable-automl-in-h2o/>
- H2O (2021) H2O Algorithms, Source: <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science.html#>
- H2O (2021) H2O Architecture, Source: <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/architecture.html>
- H2O (2021) H2O AutoML, Source: <https://www.h2o.ai/products/h2o-automl/>
- H2O (2021) Using Flow – H2O’s Web UI, Source: <https://docs.h2o.ai/h2o/latest-stable/h2o-docs/flow.html>
- Hainaut, D. (2018) A self-organizing predictive map for non-life insurance, discussion paper, Source: https://dial.uclouvain.be/pr/boreal/object/boreal%3A199020/datastream/PDF_01/view
- Heaton, J. (2017) Heaton Research, the number of hidden layers, Source: <https://www.heatonresearch.com/2017/06/01/hidden-layers.html#:~:text=The%20number%20of%20hidden%20neurons,size%20of%20the%20output%20layer>
- Hutter, F., Kotthoff, L. and Vanschoren, J. (2019) Automated Machine Learning. Methods, Systems, Challenges. Springer
- ICEA (2021), Eficacia de la lucha contra el fraude en seguros en 2020. Source: <https://www.icea.es/es-ES/noticias/Noticias/Noticias0521/Dia-12/fraude.aspx>
- INESE (2021) MAPFRE presenta verbatims, una solución tecnológica para escuchar al cliente, FUTURE LATAM. Source: <https://futurelatam.inese.es/mapfre-presenta-verbatims-una-solucion-tecnologica-para-escuchar-al-cliente/>
- Juniper Research (2018), Insurtech-based premiums to exceed \$400 billion by 2023, representing 7% of global insurance market, Source: <https://www.juniperresearch.com/press/insurtech-based-premiums-exceed-400-billion-2023#:~:text=Juniper%20forecasts%20that%20across%20property,five%2Dfold%20increase%20over%202018.>

- Kapetanvasileiou, G. (2019), Combating Insurance Fraud With Machine Learning, SAS, Source: <https://blogs.sas.com/content/hiddeninsights/2019/08/12/combating-insurance-fraud-with-machine-learning/>
- Kharel, S. (2020), Perceptron vs SVM: a quick comparison, Source: <https://medium.com/@subashkharel/perceptron-vs-svm-a-quick-comparison-6b5d6b5d64f>
- Knighton, J., Buchanan, B., Guzman, C., Elliott, R., White, E. and Rahm, B. (2020) Predicting flood insurance claims with hydrologic and socioeconomic demographics via machine learning: Exploring the roles of topography, minority populations, and political dissimilarity, Journal of Environmental Management, Source: <https://www.sciencedirect.com/science/article/pii/S0301479720309798>
- Krashennikova, E., Garcia, J., Maestre, R. and Fernández, F. (2019), Reinforcement learning for pricing strategy optimization in the insurance industry, Engineering Applications of Artificial Intelligence, Page 9, Source: <https://www.sciencedirect.com/science/article/abs/pii/S0952197619300107>
- Kunickaitė, R., Zdanaviciūtė, M and Krilavicius, T. (2020) Fraud Detection in Health Insurance Using Ensemble Learning Methods, Source: <http://ceur-ws.org/Vol-2698/p11.pdf>
- LeDell, E., Gill, N., Aiello, S., Fu, A., Candel, A., Click, C., Kraljevic, T, Nykodym, T., Aboyoun, P., Kurka, M., Malohlava, M. Rehak, L., Eckstrand, E., Hill, B., Vidrio, S., Jadhawani, S., Wang, A., Peck, R., Wong, W., Gorecki, J., Dowle, M., Tang, Y., DiPerna, L., Fryda, T. and H2O.ai. (2020), Package h2o, R Interface for the 'H2O' Scalable Machine Learning Platform, <https://cran.r-project.org/web/packages/h2o/h2o.pdf>
- Malhotra, R and Sharma, S (2018) Machine learning in insurance, Enabling insurers to become AI-driven enterprises powered by automated machine learning, FS Perspective, Accenture, Page 5, Source: https://www.accenture.com/_acnmedia/PDF-84/Accenture-Machine-Leaning-Insurance.pdf
- Meyer, D., Dimitriadou, E., Hornik, K, Weingessel, A., Leisch, F., Chang, C. and Lin, C. (2021) Package 'e1071', Misc Functions of the Department of Statistics, Probability Theory Group, Source: <https://cran.r-project.org/web/packages/e1071/e1071.pdf>

- Microsoft (2021) Aprendizaje automático, Microsoft Azure, Source:
<https://azure.microsoft.com/es-es/services/machine-learning/automatedml/>
- Nilsson, N. J. (1998) Introduction to Machine Learning, Stanford University
- Pal, R (2020) Deep Learning: The fastest growing tech. in the world, Source:
<https://rajtilakls2510.medium.com/deep-learning-the-fastest-growing-tech-in-the-world-6cd2636a68bf>
- Prakash, G. and Jyothi, R. N. (2019), A Deep Learning-Based Stacked Generalization Method to Design Smart Healthcare Solution, Springer Nature Singapore, Source:
https://www.academia.edu/43946090/A_Deep_Learning_Based_Stacked_Generalization_Method_to_Design_Smart_Healthcare_Solution
- PureAI Editors (2020), Comparing 4 ML Classification Techniques: Logistic Regression, Perceptron, Support Vector Machine, and Neural Networks. Source:
<https://pureai.com/articles/2020/04/10/ml-techniques.aspx>
- Rdocumentation (2021) GLM: Fitting Generalised Linear Models. Source:
<https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/glm>
- Research and Markets (2021), Insights on the Telematics Insurance Global Market to 2025 - Prevention of False Insurance Claims is Driving Growth, Cision PR Newswire, Source: <https://www.prnewswire.com/news-releases/insights-on-the-telematics-insurance-global-market-to-2025---prevention-of-false-insurance-claims-is-driving-growth-301267878.html>
- Ripley, B. (2019) Package ‘tree’, Classification and Regression Trees, Source:
<https://cran.r-project.org/web/packages/tree/tree.pdf>
- Sharma, R. (2019), Insurance Claim, Kaggle, Source:
https://www.kaggle.com/roshansharma/fraud-detection-in-insurance-claims/data?select=insurance_claims.csv
- Su, X. and Bai, M (2020), Stochastic gradient boosting frequency-severity model of insurance claims, Source:
<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0238000>
- Sutton, R. S. and Barto, A. G. (2018), Reinforcement Learning, An introduction, Source: ISBN: 978-0-262-19398-6
- Swiss Re Institute (2020) Sigma, Machine intelligence in insurance: insights for end-to-end enterprise transformation, No 5/2020, Source:

https://www.swissre.com/dam/jcr:dbf95c92-4f1a-496a-aefc-0b9807178d59/sigma5_2020_en.pdf

TechBullion PR (2021) How is AI Revolutionizing Insurance?, Insurtech, Source:
<https://techbullion.com/how-is-ai-revolutionizing-insurance/>

Thornton, C., Kotthoff, L., Hoos, H. H., Hutter, F. and Leyton-Brown, K. (2016), Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA, University of British Columbia, Canada, Source:
<https://www.cs.ubc.ca/labs/beta/Projects/autoweeka/papers/16-599.pdf>

Wakefield, K. (2021), A guide to the types of machine learning algorithms and their applications, SAS, Source:
https://www.sas.com/en_gb/insights/articles/analytics/machine-learning-algorithms.html

Wickham, H. and Rstudio (2021), Package tidyverse, Easily Install and Load the 'Tidyverse', <https://cran.r-project.org/web/packages/tidyverse/tidyverse.pdf>

Wolpert, D. (1992) Stacked Generalization, Neural Network, Source:
https://www.researchgate.net/publication/222467943_Stacked_Generalization

Zhang, X. and Mahadevan, S. (2018) Ensemble machine learning models for aviation incident risk prediction. Decision Support systems, Source:
<https://www.researchgate.net/requests/r88851238>

10. Annex B - Project's R code

```
#install packages
install.packages("psych")
install.packages("ggplot2")
install.packages("grid")
install.packages("gridExtra")
install.packages("h2o")
install.packages("tidyverse")
install.packages("neuralnet")
install.packages("ROCR")
install.packages("caret")
install.packages("e1071")
install.packages("AUC")
install.packages("pROC")
install.packages("MLmetrics")
install.packages("PRROC")
install.packages("tree")

#load packages
library(psych)
library(ggplot2)
library(grid)
library(gridExtra)
library(h2o)
library(tidyverse)
library(neuralnet)
library(ROCR)
library(caret)
library(e1071)
library(AUC)
library(pROC)
library(MLmetrics)
library(PRROC)
library(tree)

#####Data Analysis

#Importing data
rawdata <- read.table("C:/Users/svalle/Desktop/TFM/input_data/insurance_claims.csv",
header=TRUE,sep=",")

#Data structure
str(rawdata)
```

```

#Data summary
summary(rawdata)

#Umbrella insurance policy error
data<-rawdata
data$umbrella_limit[data$umbrella_limit == "-1000000"] <- 0

#Data description
numeric_cols <- unlist(lapply(data, is.numeric))
describe(data[numeric_cols])

# Checking if Normally distributed
barplot(prop.table(table(as.factor(data$fraud_reported))))
hist(data$injury_claim)
hist(data$property_claim)
hist(data$vehicle_claim)

#descriptive analysis segmenting by the categorical variable fraud reported
describeBy(data[numeric_cols], group=data$fraud_reported)

boxplot(split(data$total_claim_amount,data$fraud_reported),main="Total claim amount
by non-/fraudulent claims")

#Analysis between categorical variables
ggplot(data = data)+geom_bar(aes(x=fraud_reported, fill=incident_type))+ggtitle(label
= "non-/fraudulent claims per incident type")+theme_bw()
ggplot(data = data)+geom_bar(aes(x=fraud_reported,
fill=authorities_contacted))+ggtitle(label = "non-/fraudulent claims if authorities
contacted")+theme_bw()
ggplot(data = data)+geom_bar(aes(x=fraud_reported,
fill=incident_severity))+ggtitle(label = "non-/fraudulent per incident severity
type")+theme_bw()

#####AutoML with H2O

#initiate the H2O cluster
localH2O = h2o.init()

#Importing data to H2O frame
h2odata <-
h2o.importFile("C:/Users/svalle/Desktop/TFM/input_data/insurance_claims.csv",
header=TRUE,sep=",")

```

```

#Umbrella insurance policy error
h2odata[h2odata[, "umbrella_limit"] < 0, "umbrella_limit"] <- 0

#Variables description
h2o.describe(h2odata)

#Train and Test datasets
h2otrain <- h2odata[1:900,]
h2otest <- h2odata[901:1000,]

#Identifying the response column
y<-"fraud_reported"

#Identifying the predictor columns to use
x<-setdiff(names(h2odata),c(y,"incident_date","policy_number","policy_bind_date"))

#Find models
aml10mod<-h2o.automl(y=y, x=x, training_frame = h2otrain, max_models = 10)
aml600sec<-h2o.automl(y=y, x=x, training_frame = h2otrain, max_runtime_secs = 600)

# Leaderboard
lb10mod<-aml10mod@leaderboard
lb600sec<-aml600sec@leaderboard

print(lb10mod, n=nrow(lb10mod))
print(lb600sec, n=nrow(lb10mod))

#Analyse "All models" stacked ensemble model
model_ids_10mod<-as.data.frame(aml10mod@leaderboard$model_id)[,1] #Model ids
for all models in the AutoML leaderboard
se_10mod<-h2o.getModel(grep("StackedEnsemble_AllModels",model_ids_10mod,
value = TRUE)[1]) #get the "All models" stacked ensemble model
metalearner_10mod<-h2o.getModel(se_10mod@model$metalearner$name) #get the
stacked ensemble metalearner model.
print(metalearner_10mod)

#how important each of the base learners is to the ensemble and plot
h2o.varimp(metalearner_10mod)
h2o.varimp_plot(metalearner_10mod)

#Analysis and plot of the regular variable importance of the top model on the original
training data set
gbm<-h2o.getModel(grep("GBM", model_ids_10mod, value = TRUE)[1])
h2o.varimp(gbm)

```

```

h2o.varimp_plot(gbm)

#Parameters of the top model
gbm@parameters

#Confusion matrix of the top model
h2o.confusionMatrix(gbm)

#prediction with test data
results <- as.data.frame(h2o.predict(gbm, h2otest))
pred<-h2o.predict(gbm,h2otest)
print(pred)
print(results)
summary(pred)
summary(results)

#####Perceptron

#Obtaining data with error fixed and the same previous conditions
datavar<-setdiff(names(data),c("incident_date","policy_number","policy_bind_date"))
data1<-data[datavar]

#Adapting output values to avoid perceptron programming issues
data1$fraud_reported[data1$fraud_reported == "Y"] <- 1
data1$fraud_reported[data1$fraud_reported == "N"] <- 0
data1$fraud_reported <- as.integer(data1$fraud_reported)

data1$insured_sex[data1$insured_sex == "MALE"] <- 1
data1$insured_sex[data1$insured_sex == "FEMALE"] <- 2
data1$insured_sex <- as.integer(data1$insured_sex)

data1$policy_state[data1$policy_state == "IN"] <- 1
data1$policy_state[data1$policy_state == "OH"] <- 2
data1$policy_state[data1$policy_state == "IL"] <- 3
data1$policy_state <- as.integer(data1$policy_state)

data1$policy_csl[data1$policy_csl == "100/300"] <- 1
data1$policy_csl[data1$policy_csl == "250/500"] <- 2
data1$policy_csl[data1$policy_csl == "500/1000"] <- 3
data1$policy_csl <- as.integer(data1$policy_csl)

data1$incident_type[data1$incident_type == "Single Vehicle Collision"] <- 1
data1$incident_type[data1$incident_type == "Vehicle Theft"] <- 2
data1$incident_type[data1$incident_type == "Multi-vehicle Collision"] <- 3

```

```
data1$incident_type[data1$incident_type == "Parked Car"] <- 4
data1$incident_type <- as.integer(data1$incident_type)
```

```
data1$collision_type[data1$collision_type == "?"] <- 1
data1$collision_type[data1$collision_type == "Side Collision"] <- 2
data1$collision_type[data1$collision_type == "Rear Collision"] <- 3
data1$collision_type[data1$collision_type == "Front Collision"] <- 4
data1$collision_type <- as.integer(data1$collision_type)
```

```
data1$incident_severity[data1$incident_severity == "Major Damage"] <- 1
data1$incident_severity[data1$incident_severity == "Minor Damage"] <- 2
data1$incident_severity[data1$incident_severity == "Total Loss"] <- 3
data1$incident_severity[data1$incident_severity == "Trivial Damage"] <- 4
data1$incident_severity <- as.integer(data1$incident_severity)
```

```
data1$authorities_contacted[data1$authorities_contacted == "Police"] <- 1
data1$authorities_contacted[data1$authorities_contacted == "None"] <- 2
data1$authorities_contacted[data1$authorities_contacted == "Fire"] <- 3
data1$authorities_contacted[data1$authorities_contacted == "Other"] <- 4
data1$authorities_contacted[data1$authorities_contacted == "Ambulance"] <- 5
data1$authorities_contacted <- as.integer(data1$authorities_contacted)
```

```
data1$property_damage[data1$property_damage == "YES"] <- 1
data1$property_damage[data1$property_damage == "NO"] <- 2
data1$property_damage[data1$property_damage == "?"] <- 3
data1$property_damage <- as.integer(data1$property_damage)
```

```
data1$police_report_available[data1$police_report_available == "YES"] <- 1
data1$police_report_available[data1$police_report_available == "NO"] <- 2
data1$police_report_available[data1$police_report_available == "?"] <- 3
data1$police_report_available <- as.integer(data1$police_report_available)
```

```
data1$insured_education_level[data1$insured_education_level == "Associate"] <- 1
data1$insured_education_level[data1$insured_education_level == "College"] <- 2
data1$insured_education_level[data1$insured_education_level == "High School"] <- 3
data1$insured_education_level[data1$insured_education_level == "JD"] <- 4
data1$insured_education_level[data1$insured_education_level == "Masters"] <- 5
data1$insured_education_level[data1$insured_education_level == "MD"] <- 6
data1$insured_education_level[data1$insured_education_level == "PhD"] <- 7
data1$insured_education_level <- as.integer(data1$insured_education_level)
```

```
data1$insured_occupation[data1$insured_occupation == "adm-clerical"] <- 1
data1$insured_occupation[data1$insured_occupation == "armed-forces"] <- 2
data1$insured_occupation[data1$insured_occupation == "craft-repair"] <- 3
```

```

data1$insured_occupation[data1$insured_occupation == "exec-managerial"] <- 4
data1$insured_occupation[data1$insured_occupation == "farming-fishing"] <- 5
data1$insured_occupation[data1$insured_occupation == "handlers-cleaners"] <- 6
data1$insured_occupation[data1$insured_occupation == "machine-op-inspct"] <- 7
data1$insured_occupation[data1$insured_occupation == "other-service"] <- 8
data1$insured_occupation[data1$insured_occupation == "priv-house-serv"] <- 9
data1$insured_occupation[data1$insured_occupation == "prof-specialty"] <- 10
data1$insured_occupation[data1$insured_occupation == "protective-serv"] <- 11
data1$insured_occupation[data1$insured_occupation == "sales"] <- 12
data1$insured_occupation[data1$insured_occupation == "tech-support"] <- 13
data1$insured_occupation[data1$insured_occupation == "transport-moving"] <- 14
data1$insured_occupation <- as.integer(data1$insured_occupation)

```

```

data1$insured_hobbies[data1$insured_hobbies == "base-jumping"] <- 1
data1$insured_hobbies[data1$insured_hobbies == "basketball"] <- 2
data1$insured_hobbies[data1$insured_hobbies == "board-games"] <- 3
data1$insured_hobbies[data1$insured_hobbies == "bungee-jumping"] <- 4
data1$insured_hobbies[data1$insured_hobbies == "camping"] <- 5
data1$insured_hobbies[data1$insured_hobbies == "chess"] <- 6
data1$insured_hobbies[data1$insured_hobbies == "cross-fit"] <- 7
data1$insured_hobbies[data1$insured_hobbies == "dancing"] <- 8
data1$insured_hobbies[data1$insured_hobbies == "exercise"] <- 9
data1$insured_hobbies[data1$insured_hobbies == "golf"] <- 10
data1$insured_hobbies[data1$insured_hobbies == "hiking"] <- 11
data1$insured_hobbies[data1$insured_hobbies == "kayaking"] <- 12
data1$insured_hobbies[data1$insured_hobbies == "movies"] <- 13
data1$insured_hobbies[data1$insured_hobbies == "paintball"] <- 14
data1$insured_hobbies[data1$insured_hobbies == "polo"] <- 15
data1$insured_hobbies[data1$insured_hobbies == "reading"] <- 16
data1$insured_hobbies[data1$insured_hobbies == "skydiving"] <- 17
data1$insured_hobbies[data1$insured_hobbies == "sleeping"] <- 18
data1$insured_hobbies[data1$insured_hobbies == "video-games"] <- 19
data1$insured_hobbies[data1$insured_hobbies == "yachting"] <- 20
data1$insured_hobbies <- as.integer(data1$insured_hobbies)

```

```

data1$insured_relationship[data1$insured_relationship == "husband"] <- 1
data1$insured_relationship[data1$insured_relationship == "not-in-family"] <- 2
data1$insured_relationship[data1$insured_relationship == "other-relative"] <- 3
data1$insured_relationship[data1$insured_relationship == "own-child"] <- 4
data1$insured_relationship[data1$insured_relationship == "unmarried"] <- 5
data1$insured_relationship[data1$insured_relationship == "wife"] <- 6
data1$insured_relationship <- as.integer(data1$insured_relationship)

```

```

data1$incident_state[data1$incident_state == "NC"] <- 1

```

```

data1$incident_state[data1$incident_state == "NY"] <- 2
data1$incident_state[data1$incident_state == "OH"] <- 3
data1$incident_state[data1$incident_state == "PA"] <- 4
data1$incident_state[data1$incident_state == "SC"] <- 5
data1$incident_state[data1$incident_state == "VA"] <- 6
data1$incident_state[data1$incident_state == "WV"] <- 7
data1$incident_state <- as.integer(data1$incident_state)

```

```

data1$incident_city[data1$incident_city == "Arlington"] <- 1
data1$incident_city[data1$incident_city == "Columbus"] <- 2
data1$incident_city[data1$incident_city == "Hillsdale"] <- 3
data1$incident_city[data1$incident_city == "Northbend"] <- 4
data1$incident_city[data1$incident_city == "Northbrook"] <- 5
data1$incident_city[data1$incident_city == "Riverwood"] <- 6
data1$incident_city[data1$incident_city == "Springfield"] <- 7
data1$incident_city <- as.integer(data1$incident_city)

```

```

data1$auto_make[data1$auto_make == "Accura"] <- 1
data1$auto_make[data1$auto_make == "Audi"] <- 2
data1$auto_make[data1$auto_make == "BMW"] <- 3
data1$auto_make[data1$auto_make == "Chevrolet"] <- 4
data1$auto_make[data1$auto_make == "Dodge"] <- 5
data1$auto_make[data1$auto_make == "Ford"] <- 6
data1$auto_make[data1$auto_make == "Honda"] <- 7
data1$auto_make[data1$auto_make == "Jeep"] <- 8
data1$auto_make[data1$auto_make == "Mercedes"] <- 9
data1$auto_make[data1$auto_make == "Nissan"] <- 10
data1$auto_make[data1$auto_make == "Saab"] <- 11
data1$auto_make[data1$auto_make == "Suburu"] <- 12
data1$auto_make[data1$auto_make == "Toyota"] <- 13
data1$auto_make[data1$auto_make == "Volkswagen"] <- 14
data1$auto_make <- as.integer(data1$auto_make)

```

```

#Removing variables with too many levels
datavar1<-setdiff(names(data1),c("auto_model","incident_location"))
data2<-data1[datavar1]

```

```

#Train and Test datasets
pertrain<-data2[1:900,]
pertest<-data2[901:1000,]

```

```

#Perceptron model
perceptron<-
neuralnet(fraud_reported~months_as_customer+age+policy_state+policy_csl+policy_d

```

```

eductable+policy_annual_premium+umbrella_limit+insured_zip+insured_sex+insured_
education_level+insured_occupation+insured_hobbies+insured_relationship+capital.gai
ns+capital.loss+incident_type+collision_type+incident_severity+authorities_contacted+
incident_state+incident_city+incident_hour_of_the_day+number_of_vehicles_involved
+property_damage+bodily_injuries+witnesses+police_report_available+total_claim_am
ount+injury_claim+property_claim+vehicle_claim+auto_make+auto_year,
data=pertrain,rep = 3, hidden=0,threshold=0.5, stepmax = 10000000)

```

```
#####GLM
```

```

#Obtaining data with error fixed and the same previous conditions
datavar<-setdiff(names(data),c("incident_date","policy_number","policy_bind_date"))
data1<-data[datavar]

```

```

#Adapting independent variable to make GLM function to work
data1$fraud_reported <- as.factor(data1$fraud_reported)

```

```

#Train and Test datasets
glmtrain<-data1[1:900,]
glmtest<-data1[901:1000,]

```

```

#GLM model
glm <- glm( fraud_reported ~., data = glmtrain, family = binomial, maxit = 100)
summary(glm)

```

```

glm1 <- glm(fraud_reported ~ insured_hobbies+incident_severity, data = glmtrain,
family = binomial)
summary(glm1)

```

```

# Predictions and confusion matrix with train data set
probtrain <- glm1 %>% predict(glmtrain, type = "response")
predict.train <- ifelse(probtrain > 0.5, "Y", "N")
predict.train<-as.factor(predict.train)
confusionMatrix(glmtrain$fraud_reported,predict.train)

```

```

#AUC
predict.train<-as.integer(predict.train)
glmtrain$fraud_reported<-as.integer(glmtrain$fraud_reported)
roc_obj <- roc(glmtrain$fraud_reported, predict.train)
auc(roc_obj)

```

```

#Logloss
predict.train[predict.train == "1"] <- 0
predict.train[predict.train == "2"] <- 1

```

```

glmtrain$fraud_reported[glmtrain$fraud_reported == "1"] <- 0
glmtrain$fraud_reported[glmtrain$fraud_reported == "2"] <- 1

LogLoss(predict.train,glmtrain$fraud_reported)

#AUCPR
PRAUC(predict.train,glmtrain$fraud_reported)

#prediction with test data and model accuracy
probttest <- glm1 %>% predict(glmtest, type = "response")
predict.test <- ifelse(probttest > 0.5, "Y", "N")
predict.test<-as.factor(predict.test)
summary(predict.test)
mean(predict.test == glmtest$fraud_reported)

#####Multilayer perceptron

#Obtaining data with error fixed and the same previous conditions
datavar<-setdiff(names(data),c("incident_date","policy_number","policy_bind_date"))
data1<-data[datavar]

#Adapting output values to avoid perceptron programming issues
data1$fraud_reported[data1$fraud_reported == "Y"] <- 1
data1$fraud_reported[data1$fraud_reported == "N"] <- 0
data1$fraud_reported <- as.integer(data1$fraud_reported)

data1$insured_sex[data1$insured_sex == "MALE"] <- 1
data1$insured_sex[data1$insured_sex == "FEMALE"] <- 2
data1$insured_sex <- as.integer(data1$insured_sex)

data1$policy_state[data1$policy_state == "IN"] <- 1
data1$policy_state[data1$policy_state == "OH"] <- 2
data1$policy_state[data1$policy_state == "IL"] <- 3
data1$policy_state <- as.integer(data1$policy_state)

data1$policy_csl[data1$policy_csl == "100/300"] <- 1
data1$policy_csl[data1$policy_csl == "250/500"] <- 2
data1$policy_csl[data1$policy_csl == "500/1000"] <- 3
data1$policy_csl <- as.integer(data1$policy_csl)

data1$incident_type[data1$incident_type == "Single Vehicle Collision"] <- 1
data1$incident_type[data1$incident_type == "Vehicle Theft"] <- 2
data1$incident_type[data1$incident_type == "Multi-vehicle Collision"] <- 3

```

```
data1$incident_type[data1$incident_type == "Parked Car"] <- 4
data1$incident_type <- as.integer(data1$incident_type)
```

```
data1$collision_type[data1$collision_type == "?"] <- 1
data1$collision_type[data1$collision_type == "Side Collision"] <- 2
data1$collision_type[data1$collision_type == "Rear Collision"] <- 3
data1$collision_type[data1$collision_type == "Front Collision"] <- 4
data1$collision_type <- as.integer(data1$collision_type)
```

```
data1$incident_severity[data1$incident_severity == "Major Damage"] <- 1
data1$incident_severity[data1$incident_severity == "Minor Damage"] <- 2
data1$incident_severity[data1$incident_severity == "Total Loss"] <- 3
data1$incident_severity[data1$incident_severity == "Trivial Damage"] <- 4
data1$incident_severity <- as.integer(data1$incident_severity)
```

```
data1$authorities_contacted[data1$authorities_contacted == "Police"] <- 1
data1$authorities_contacted[data1$authorities_contacted == "None"] <- 2
data1$authorities_contacted[data1$authorities_contacted == "Fire"] <- 3
data1$authorities_contacted[data1$authorities_contacted == "Other"] <- 4
data1$authorities_contacted[data1$authorities_contacted == "Ambulance"] <- 5
data1$authorities_contacted <- as.integer(data1$authorities_contacted)
```

```
data1$property_damage[data1$property_damage == "YES"] <- 1
data1$property_damage[data1$property_damage == "NO"] <- 2
data1$property_damage[data1$property_damage == "?"] <- 3
data1$property_damage <- as.integer(data1$property_damage)
```

```
data1$police_report_available[data1$police_report_available == "YES"] <- 1
data1$police_report_available[data1$police_report_available == "NO"] <- 2
data1$police_report_available[data1$police_report_available == "?"] <- 3
data1$police_report_available <- as.integer(data1$police_report_available)
```

```
data1$insured_education_level[data1$insured_education_level == "Associate"] <- 1
data1$insured_education_level[data1$insured_education_level == "College"] <- 2
data1$insured_education_level[data1$insured_education_level == "High School"] <- 3
data1$insured_education_level[data1$insured_education_level == "JD"] <- 4
data1$insured_education_level[data1$insured_education_level == "Masters"] <- 5
data1$insured_education_level[data1$insured_education_level == "MD"] <- 6
data1$insured_education_level[data1$insured_education_level == "PhD"] <- 7
data1$insured_education_level <- as.integer(data1$insured_education_level)
```

```
data1$insured_occupation[data1$insured_occupation == "adm-clerical"] <- 1
data1$insured_occupation[data1$insured_occupation == "armed-forces"] <- 2
data1$insured_occupation[data1$insured_occupation == "craft-repair"] <- 3
```

```

data1$insured_occupation[data1$insured_occupation == "exec-managerial"] <- 4
data1$insured_occupation[data1$insured_occupation == "farming-fishing"] <- 5
data1$insured_occupation[data1$insured_occupation == "handlers-cleaners"] <- 6
data1$insured_occupation[data1$insured_occupation == "machine-op-inspct"] <- 7
data1$insured_occupation[data1$insured_occupation == "other-service"] <- 8
data1$insured_occupation[data1$insured_occupation == "priv-house-serv"] <- 9
data1$insured_occupation[data1$insured_occupation == "prof-specialty"] <- 10
data1$insured_occupation[data1$insured_occupation == "protective-serv"] <- 11
data1$insured_occupation[data1$insured_occupation == "sales"] <- 12
data1$insured_occupation[data1$insured_occupation == "tech-support"] <- 13
data1$insured_occupation[data1$insured_occupation == "transport-moving"] <- 14
data1$insured_occupation <- as.integer(data1$insured_occupation)

```

```

data1$insured_hobbies[data1$insured_hobbies == "base-jumping"] <- 1
data1$insured_hobbies[data1$insured_hobbies == "basketball"] <- 2
data1$insured_hobbies[data1$insured_hobbies == "board-games"] <- 3
data1$insured_hobbies[data1$insured_hobbies == "bungee-jumping"] <- 4
data1$insured_hobbies[data1$insured_hobbies == "camping"] <- 5
data1$insured_hobbies[data1$insured_hobbies == "chess"] <- 6
data1$insured_hobbies[data1$insured_hobbies == "cross-fit"] <- 7
data1$insured_hobbies[data1$insured_hobbies == "dancing"] <- 8
data1$insured_hobbies[data1$insured_hobbies == "exercise"] <- 9
data1$insured_hobbies[data1$insured_hobbies == "golf"] <- 10
data1$insured_hobbies[data1$insured_hobbies == "hiking"] <- 11
data1$insured_hobbies[data1$insured_hobbies == "kayaking"] <- 12
data1$insured_hobbies[data1$insured_hobbies == "movies"] <- 13
data1$insured_hobbies[data1$insured_hobbies == "paintball"] <- 14
data1$insured_hobbies[data1$insured_hobbies == "polo"] <- 15
data1$insured_hobbies[data1$insured_hobbies == "reading"] <- 16
data1$insured_hobbies[data1$insured_hobbies == "skydiving"] <- 17
data1$insured_hobbies[data1$insured_hobbies == "sleeping"] <- 18
data1$insured_hobbies[data1$insured_hobbies == "video-games"] <- 19
data1$insured_hobbies[data1$insured_hobbies == "yachting"] <- 20
data1$insured_hobbies <- as.integer(data1$insured_hobbies)

```

```

data1$insured_relationship[data1$insured_relationship == "husband"] <- 1
data1$insured_relationship[data1$insured_relationship == "not-in-family"] <- 2
data1$insured_relationship[data1$insured_relationship == "other-relative"] <- 3
data1$insured_relationship[data1$insured_relationship == "own-child"] <- 4
data1$insured_relationship[data1$insured_relationship == "unmarried"] <- 5
data1$insured_relationship[data1$insured_relationship == "wife"] <- 6
data1$insured_relationship <- as.integer(data1$insured_relationship)

```

```

data1$incident_state[data1$incident_state == "NC"] <- 1

```

```

data1$incident_state[data1$incident_state == "NY"] <- 2
data1$incident_state[data1$incident_state == "OH"] <- 3
data1$incident_state[data1$incident_state == "PA"] <- 4
data1$incident_state[data1$incident_state == "SC"] <- 5
data1$incident_state[data1$incident_state == "VA"] <- 6
data1$incident_state[data1$incident_state == "WV"] <- 7
data1$incident_state <- as.integer(data1$incident_state)

```

```

data1$incident_city[data1$incident_city == "Arlington"] <- 1
data1$incident_city[data1$incident_city == "Columbus"] <- 2
data1$incident_city[data1$incident_city == "Hillsdale"] <- 3
data1$incident_city[data1$incident_city == "Northbend"] <- 4
data1$incident_city[data1$incident_city == "Northbrook"] <- 5
data1$incident_city[data1$incident_city == "Riverwood"] <- 6
data1$incident_city[data1$incident_city == "Springfield"] <- 7
data1$incident_city <- as.integer(data1$incident_city)

```

```

data1$auto_make[data1$auto_make == "Accura"] <- 1
data1$auto_make[data1$auto_make == "Audi"] <- 2
data1$auto_make[data1$auto_make == "BMW"] <- 3
data1$auto_make[data1$auto_make == "Chevrolet"] <- 4
data1$auto_make[data1$auto_make == "Dodge"] <- 5
data1$auto_make[data1$auto_make == "Ford"] <- 6
data1$auto_make[data1$auto_make == "Honda"] <- 7
data1$auto_make[data1$auto_make == "Jeep"] <- 8
data1$auto_make[data1$auto_make == "Mercedes"] <- 9
data1$auto_make[data1$auto_make == "Nissan"] <- 10
data1$auto_make[data1$auto_make == "Saab"] <- 11
data1$auto_make[data1$auto_make == "Suburu"] <- 12
data1$auto_make[data1$auto_make == "Toyota"] <- 13
data1$auto_make[data1$auto_make == "Volkswagen"] <- 14
data1$auto_make <- as.integer(data1$auto_make)

```

```

#Removing variables with too many levels
datavar1<-setdiff(names(data1),c("auto_model","incident_location"))
data2<-data1[datavar1]

```

```

#Train and Test datasets
mpertrain<-data2[1:900,]
mpertest<-data2[901:1000,]

```

```

#Perceptron model
mperceptron<-
neuralnet(fraud_reported~months_as_customer+age+policy_state+policy_csl+policy_d

```

```

eductable+policy_annual_premium+umbrella_limit+insured_zip+insured_sex+insured_
education_level+insured_occupation+insured_hobbies+insured_relationship+capital.gai
ns+capital.loss+incident_type+collision_type+incident_severity+authorities_contacted+
incident_state+incident_city+incident_hour_of_the_day+number_of_vehicles_involved
+property_damage+bodily_injuries+witnesses+police_report_available+total_claim_am
ount+injury_claim+property_claim+vehicle_claim+auto_make+auto_year,
data=mpertrain, linear.output=FALSE)
plot(mperceptron)
mperceptron$result.matrix

```

```

#confusion matrix with train data

```

```

mpertrain1 <- subset(mpertrain, select =
c("months_as_customer","age","policy_state","policy_csl","policy_deductable","policy
_annual_premium","umbrella_limit","insured_zip","insured_sex","insured_education_l
evel","insured_occupation","insured_hobbies","insured_relationship","capital.gains","c
apital.loss","incident_type","collision_type","incident_severity","authorities_contacted"
,"incident_state","incident_city","incident_hour_of_the_day","number_of_vehicles_inv
olved","property_damage","bodily_injuries","witnesses","police_report_available","tota
l_claim_amount","injury_claim","property_claim","vehicle_claim","auto_make","auto_
year"))
mperceptrontrain.result <- compute(mperceptron, mpertrain1)
result.train <- data.frame(actual = mpertrain$fraud_reported, prediction =
mperceptrontrain.result$net.result)
roundedresult<-sapply(result.train,round,digits=0)
roundedresultdf=data.frame(roundedresult)
attach(roundedresultdf)
table(actual,prediction)

```

```

#AUC

```

```

mpertrain$fraud_reported<-as.integer(mpertrain$fraud_reported)
roc_objMLP <- roc(mpertrain$fraud_reported, mperceptrontrain.result$net.result)
auc(roc_objMLP)

```

```

#Logloss

```

```

LogLoss(mperceptrontrain.result$net.result,mpertrain$fraud_reported)

```

```

#AUCPR

```

```

PRAUC(mperceptrontrain.result$net.result,mpertrain$fraud_reported)

```

```

#Prediction with test data

```

```

mpertest1 <- subset(mptest, select =
c("months_as_customer","age","policy_state","policy_csl","policy_deductable","policy
_annual_premium","umbrella_limit","insured_zip","insured_sex","insured_education_l
evel","insured_occupation","insured_hobbies","insured_relationship","capital.gains","c

```

```

apital.loss","incident_type","collision_type","incident_severity","authorities_contacted"
,"incident_state","incident_city","incident_hour_of_the_day","number_of_vehicles_inv
olved","property_damage","bodily_injuries","witnesses","police_report_available","tota
l_claim_amount","injury_claim","property_claim","vehicle_claim","auto_make","auto_
year"))
mperceptrontest.result <- compute(mperceptron, mpertest1)
result.test <- data.frame(actual = mpertest$fraud_reported, prediction =
mperceptrontest.result$net.result)
print(result.test)

```

```

#####Support Vector Machine

```

```

#Obtaining data with error fixed and the same previous conditions
datavar<-setdiff(names(data),c("incident_date","policy_number","policy_bind_date"))
data1<-data[datavar]

```

```

#Adapting output values to avoid SVM programming issues
data1$fraud_reported[data1$fraud_reported == "Y"] <- 1
data1$fraud_reported[data1$fraud_reported == "N"] <- 0
data1$fraud_reported <- as.integer(data1$fraud_reported)

```

```

#Train and Test datasets
svmtrain<-data1[1:900,]
svmtest<-data1[901:1000,]

```

```

#SVM model
svm <- svm(fraud_reported ~ ., data=svmtrain,type='C-classification')
svmlin <- svm(fraud_reported ~ ., data=svmtrain,type='C-classification',
kernel="linear")
svmpol <- svm(fraud_reported ~ ., data=svmtrain,type='C-classification',
kernel="polynomial")
svmsig <- svm(fraud_reported ~ ., data=svmtrain,type='C-classification',
kernel="sigmoid")
summary(svm)
summary(svmlin)
summary(svmpol)
summary(svmsig)

```

```

#confusion matrix with train data (default radial kernel)
pred<-predict(svm,svmtrain)
confmatrix = table(Predicted=pred, Actual = svmtrain$fraud_reported)
print(confmatrix)

```

```

#confusion matrix with train data (linear Kernel)

```

```

predlin<-predict(svmlin,svmtrain)
confmatrixlin = table(Predicted=predlin, Actual = svmtrain$fraud_reported)
print(confmatrixlin)

#confusion matrix with train data (polynomial kernel)
predpol<-predict(svmpol,svmtrain)
confmatrixpol = table(Predicted=predpol, Actual = svmtrain$fraud_reported)
print(confmatrixpol)

#confusion matrix with train data (sigmoid kernel)
predsig<-predict(svmsig,svmtrain)
confmatrixsig = table(Predicted=predsig, Actual = svmtrain$fraud_reported)
print(confmatrixsig)

#AUC
predlin<-as.integer(predlin)
svmtrain$fraud_reported<-as.integer(svmtrain$fraud_reported)
roc_objSVM <- roc(svmtrain$fraud_reported, predlin)
auc(roc_objSVM)

#Logloss
predlin[predlin == "1"] <- 0
predlin[predlin == "2"] <- 1

LogLoss(predlin,svmtrain$fraud_reported)

#AUCPR
PRAUC(predlin,svmtrain$fraud_reported)

#Prediction with test data not possible due to difference levels in auto model variable
predlintest<-predict(svmlin,newdata=svmtest)
svmtrain$auto_model <- as.factor(svmtrain$auto_model)
svmtest$auto_model <- as.factor(svmtest$auto_model)
str(svmtrain$auto_model)
str(svmtest$auto_model)

#####Decision Tree

#Obtaining data with error fixed and the same previous conditions
datavar<-
setdiff(names(data),c("incident_date","policy_number","policy_bind_date","auto_mode
l","incident_location"))
data1<-data[datavar]

```

```

data1$fraud_reported<-as.factor(data1$fraud_reported)
data1$policy_state <- as.factor(data1$policy_state)
data1$policy_csl <- as.factor(data1$policy_csl)
data1$insured_sex <- as.factor(data1$insured_sex)
data1$insured_education_level <- as.factor(data1$insured_education_level)
data1$insured_occupation <- as.factor(data1$insured_occupation)
data1$insured_hobbies <- as.factor(data1$insured_hobbies)
data1$insured_relationship <- as.factor(data1$insured_relationship)
data1$incident_type <- as.factor(data1$incident_type)
data1$incident_severity <- as.factor(data1$incident_severity)
data1$authorities_contacted <- as.factor(data1$authorities_contacted)
data1$incident_state <- as.factor(data1$incident_state)
data1$incident_city <- as.factor(data1$incident_city)
data1$incident_location <- as.factor(data1$incident_location)
data1$property_damage <- as.factor(data1$property_damage)
data1$police_report_available <- as.factor(data1$police_report_available)
data1$auto_make <- as.factor(data1$auto_make)
data1$auto_model <- as.factor(data1$auto_model)
data1$collision_type <- as.factor(data1$collision_type)

```

```
#Train and Test datasets
```

```

dttrain<-data1[1:900,]
dttest<-data1[901:1000,]

```

```
#Decision Tree model
```

```

dtree<-tree(fraud_reported ~ ., data = dttrain)
summary(dtree)
plot(dtree)
text(dtree ,pretty = 1)

```

```
#confusion matrix with train data
```

```

dtrepred<-predict(dtree, newdata = dttrain,type = "class")
with(dttrain,table(dtrepred,fraud_reported))

```

```
#AUC
```

```

dtrepred<-as.integer(dtrepred)
roc_objDT <- roc(dttrain$fraud_reported, dtrepred)
auc(roc_objDT)

```

```
#Logloss
```

```

dtrepred[dtrepred == "1"] <- 0
dtrepred[dtrepred == "2"] <- 1

```

```
dttrain$fraud_reported<-as.integer(dttrain$fraud_reported)
```

```
dttrain$fraud_reported[dttrain$fraud_reported == "1"] <- 0
dttrain$fraud_reported[dttrain$fraud_reported == "2"] <- 1
```

```
LogLoss(dtrepred,dttrain$fraud_reported)
```

```
#AUCPR
```

```
PRAUC(dtrepred,dttrain$fraud_reported)
```

```
#Prediction with test data set
```

```
dtrepredtest<-predict(dtrees, newdata = dttest,type = "class")
with(dttest,table(dtrepredtest,fraud_reported))
```